FAKULTET ORGANIZACIJE I INFORMATIKE

ZBORNIK RADOVA

RAČUNALNE IGRE 2019

STRUČNA KONFERENCIJA

UREDNICI

doc. dr. sc. Mario Konecki
dr. sc. Mladen Konecki
Andrej Kovačević

ISSN 2623-7679

IMPRESUM

IZDAVAČ Konferencija Računalne igre 2019

NAKLADNIK Fakultet organizacije i informatike
Pavlinska 2, 42000 Varaždin

KONTAKT racunalne-igre@foi.hr

UREDNICI doc. dr. sc. Mario Konecki
Sveučilište u Zagrebu

dr. sc. Mladen Konecki
Sveučilište u Zagrebu

Andrej Kovačević
Exordium Games

GLAVNI UREDNIK doc. dr. sc. Mario Konecki
Sveučilište u Zagrebu

DATUM ODRŽAVANJA KONFERENCIJE 4. listopada 2019.

MJESTO ODRŽAVANJA KONFERENCIJE Fakultet organizacije i informatike
Pavlinska 2, 42000 Varaždin

ISSN 2623-7679

STRUČNA KONFERENCIJA

RAČUNALNE IGRE 2019

4. listopada 2019.

ORGANIZATORI I SUORGANIZATORI

foi
SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

CECIIS
Central European Conference on Information and Intelligent Systems

Universitas Studiorum Zagrabiensis MDCLXIX

Sveučilište u Rijeci

Sveučilište u Splitu

Sveučilište Josipa Jurja Strossmayera u Osijeku

Sveučilište u Zadru
1396
Universitas Studiorum Jadertina
2002

Sveučilište u Dubrovniku
Universitas Studiorum Ragusina

Universitas Studiorum Polensis Georgii Dobrila

University North
Sveučilište Sjever

ALGEBRA
VISOKO UČILIŠTE

MACHINA
GAME DEV ACADEMY

VERN'

Grad Varaždin

Grad Bjelovar

Grad Daruvar

GENERALNI POKROVITELJ

SISAČKO-MOSLAVAČKA ŽUPANIJA

STRUČNA KONFERENCIJA

RAČUNALNE IGRE 2019

4. listopada 2019.

ORGANIZATORI I SUORGANIZATORI

foi
SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

CECIIS
Central European Conference on Information and Intelligent Systems

ALGEBRA
VISOKO UČILIŠTE

MACHINA
GAME DEV ACADEMY

VERN'

Grad Varaždin

Grad Bjelovar

Grad Daruvar

GENERALNI POKROVITELJ

SISAČKO-MOSLAVAČKA ŽUPANIJA

SADRŽAJ

1 **Izrada jednostavne dječje edukativne igre za web okruženje u grafičkom sučelju alata Construct 2**
Tomislav Novak

9 **Stanje industrije proizvodnje videoigara u Republici Hrvatskoj**
Mario Konecki, Davor Švaić i Andrej Kovačević

19 **Trip the Ark Fantastic kao Gesamtkunstwerk**
Aleksandar Gavrilović i Matija Vigato

27 **Razvoj sustava za interakciju unutar višekorisničke umrežene igre korištenjem sustava Unreal**
Dobrila Šunde i Mirko Sužnjević

41 **Vrednovanje korisničkog iskustva 2D platformske igre**
Domagoj Bakić i Dijana Plantak Vukovac

51 **ICT-AAC aplikacije kao medij za komunikaciju i učenje**
Ivana Rašan, Ivan Slivar, Matea Žilak, Željka Car i Jasmina Ivšac Pavliša

59 **Izrada multiplatformske igre pomoću programskog jezika OpenGL i Jave**
Marijana Borovec i Marko Prosen

70 **Web tehnologije za prikaz 3D objekata**
Aleksandar Trajkov i Mario Konecki

77 **Izrada trodimenzionalne platformske igre u programskom alatu Unity**
Denis Huber, Danijel Radošević i Mladen Konecki

83 **Izrada računalne igre u tri dimenzije sa zagonetkama**
Hrvoje Hodak, Robert Kudelić i Mladen Konecki

89 **Korištenje stabla ponašanja za implementaciju umjetne inteligencije**
Patrik Dolovski i Mladen Konecki

95 **Izrada igre obrane tornjevima u programskom alatu Unity**
Aldin Alagić i Mladen Konecki

102 **Proceduralno generiranje razina 2D igre u programskom alatu Unity**
Domagoj Ćurko, Danijel Radošević i Mladen Konecki

108 **Izrada akcijske računalne igre iz prvog lica u programskom alatu Unreal Engine 4**
Josip Petanjek, Danijel Radošević i Mladen Konecki

114 **Izrada računalne igre u programskom alatu Construct 3**
Luka Perić i Mladen Konecki

119 **Izrada igre borbene arene za više igrača u programskom alatu Unity**
Dario Poje i Mladen Konecki

124 **Izrada igre obrane tornjeva u programskom jeziku C++ i SDL-u**
Tomislav Vugrinec, Danijel Radošević i Mladen Konecki

130 **Implementacija osnovnih mehanika igre uloga u programskom alatu Unreal Engine 4**
Ariana-Lea Zuber i Mladen Konecki

137 **Izrada strateške igre u programskom alatu GameMaker Studio 2**
Jakov Štulić i Mladen Konecki

143 **Izrada igre uloga u programskom alatu Unity**
Karlo Vuljanko i Mladen Konecki

149 **Izrada inkrementalne igre za Android operacijski sustav pomoću programskog alata Unity**
Stiven Drvoderić, Danijel Radošević i Mladen Konecki

156 **Izrada strateške igre na poteze u programskom alatu Unity**
Tomislav Jukica i Mladen Konecki

161 **Žanrovi računalnih igara**
Antonio Brnada i Mario Konecki

168 **E-sport: usporedba s tradicionalnim sportom i mogućnost razvoja ove vrste sporta u studentskoj populaciji**
Sanja Ćurković, Mario Konecki i Damir Vučić

176 **Kombinatorne igre**
Zvonimir Galić, Mirko Čubrilo i Mario Konecki

182 **Izrada igre "Potapanje brodova" u C++-u**
Mislav Matijević

Izrada jednostavne dječje edukativne igre za web okruženje u grafičkom sučelju alata Construct 2

Tomislav Novak
Fakultet organizacije i informatike, Sveučilište u Zagrebu
fritexvz@gmail.com

Sažetak

U ovom radu opisan je postupak izrade jednostavne edukativne računalne igre 'Razvrstaj otpad' za djecu putem grafičkog sučelja alata Construct 2. Igra je namijenjena za web platformu, a temeljena je na razvojnoj okolini HTML5 (engl. HyperText Markup Language version 5) i JavaScriptu. Igru je moguće instalirati putem internetskog preglednika i pokrenuti kao odvojenu 'Progresivnu web aplikaciju' (PWA) s početnog zaslona korisnikova uređaja. Također, izrađena je i inačica u obliku Android aplikacije koju pokreće 'WebViewClient'.

Ključne riječi: HTML5 Canvas, Progresivne web aplikacije (PWA), Construct 2

Uvod

Svijet se brzo mijenja, a inovativne i nove tehnologije tržištu se predstavljaju iz godine u godinu. Nove tehnologije dostupnije su nego ikad, a njihovu važnost i primjenu sve više uviđamo u raznim područjima, od medicine do poljoprivrede. Sve prisutnije su mobilne aplikacije za operativne sustave poput Android OS ili iOS. Kako bi se izradila aplikacija za određeni mobilni operativni sustav, potrebno je mnogo tehničkog znanja i praktičnog rada. Jedna od novih tehnologija je i 'HTML5 Canvas' [1] razvojna okolina kojom je omogućeno napredno prikazivanje uglađenih dvodimenzionalnih i trodimenzionalnih animacija te iscrtavanje grafičkih elemenata i objekata unutar sučelja internetskog preglednika, a sve to koristeći se prezentacijskim HTML [2] jezikom za izradu web stranica te skriptnim programskim jezikom JavaScript [3]. Cilj ovog rada je kroz praktični izvedeni rad prikazati razinu na kojoj se može izraditi jedna jednostavna edukativna računalna igra za djecu u alatu Construct 2. U igrici je najvažnija edukativna komponenta.

Tehnologija

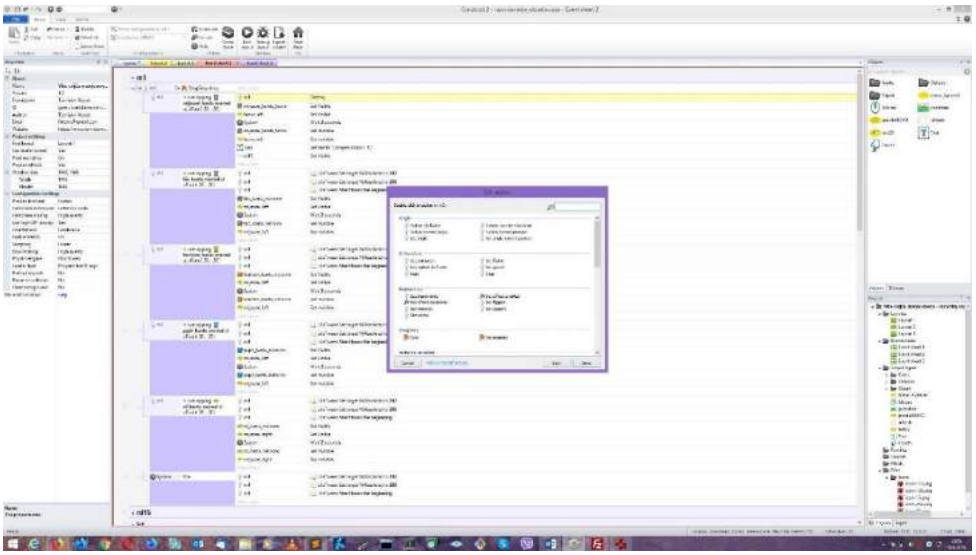
Za izradu ove računalne igre korišten je jednostavan i brz alat Construct 2 [4]. Alat Construct 2 korisniku omogućuje da u grafičkom sučelju mišem povlači, odvlači i spušta elemente i objekte na scensko platno po željenom rasporedu,

da kreira različite scene, uređuje događaje i stvara vizualne efekte nad objektima na temelju interakcije korisnika unutar vremenskog razdoblja, omogućuje izvođenje ispitivanja funkcionalnosti igre unutar zadanog internetskog preglednika, omogućuje izvoz izrađene igre za različite platforme poput web platforme, Xbox 360, Android OS, iOS, Windows Store, Linux, Mac OS i drugih. Također, alat je proširiv uz mnoštvo dostupnih gotovih grafičkih i ostalih audio-vizualnih elemenata koji se mogu i kupiti putem službene elektroničke trgovine na Internetu.

Construct 2 ne ograničava kreativnost korisnika samo na osnovne radnje mišem i kreiranje događaja nad objektima, već nudi trenutačni pristup desecima postojećih vrsti predložaka igara poput jednostavnih puzli, pucačina iz prvog lica i sl. Čak i sami možemo stvarati likove te ih spremati kao objekte za kasniju upotrebu.

Tehnologije i alati koji su upotrjebljeni za razvoju igre su: HTML5 , CSS [5], JavaScript, jQuery [6], Adobe Photoshop, Adobe Illustrator te smještaj na web poslužitelju (server), Android Studio (Java) [7].

Izvorni kod izvezen je u HTML obliku i pisan u skriptnom JavaScript programskom jeziku. Grafički elementi izrađeni su u alatu Adobe Photoshop te Adobe Illustratoru. Animacije objekata za igru poput vibracije objekata na pogrešno odabrani spremnik za otpad kreirane su putem liste događaja za pojedine objekte na sceni. Osim scenskih elemenata igre, izvoz iz alata Construct 2 je i datoteka manifest u 'JSON' (engl. JavaScript Object Notation) [8] obliku podataka koja sadržava informacije o ikonama, pozdravnom zaslonu, putanji pokretanja i dr. parametrima obaveznim za funkcionalnost progresivne web aplikacije (PWA) [9]. Od vanjskih biblioteka za dinamičnost elemenata upotrjebljen je skup biblioteka jQuery.



Slika br. 1: Uređivanje događaja za objekte na sceni u alatu Construct 2

Progresivna web aplikacija (PWA)

Kako bi igru bilo moguće pokrenuti u internetskom pregledniku, tijekom postupka izvoza igre za web platformu igra je prilagođena i mora ispunjavati uvjete za 'Progresivne web aplikacije (PWA)'. Progresivne web aplikacije stvaraju korisnički doživljaj čiji je dohvat usporediv s webom, a odlikuju ih pouzdanost, brzina i privlačnost. Također progresivne web aplikacije se trenutno učitavaju, brzo reagiraju na interakciju korisnika, kvalitetnog su prikaza, a važan čimbenik je upravo taj što pružaju prirodan dojam aplikacije na instaliranom uređaju s potpunim korisničkim doživljajem.

Progresivne web aplikacije mogu se instalirati putem 'Web Service Worker' [10] API (engl. Application programming interface) [11] sučelja te smjestiti na korisnikovu početnom zaslonu i to bez potrebe za trgovinom aplikacija. Zahvaljujući 'manifest' datoteci oblikovanoj u tekstualnom 'JSON' (engl. JavaScript Object Notation) obliku podataka, pružaju potpuni doživljaj upotrebe na cijelom zaslonu, dok uz pomoć 'Web Push API' [12] (engl. Application programming interface) sučelja mogu korisnika obavijestiti o aktualnostima u stvarnom vremenu za neki događaj, pa čak i ponovo privući korisnike ili im ponuditi nešto drugačije.

Pri samom pokretanju s korisnikova početnog zaslona, uslužne skripte omogućuju trenutno pokretanje progresivne web aplikacije bez obzira na stanje mreže. Manifest progresivne web aplikacije omogućuje kontrolu nad izgledom aplikacije i njezinim pokretanjem. U manifestu određujemo ikone za početni zaslon i fotografiju pozdravnog zaslona koji se prikazuje tijekom pokretanja i učitavanja aplikacije, kodni jezik, stranicu koja će se učitati kad se aplikacija pokrene, stranicu koja će se učitati u slučaju izvan mrežnog rada, orijentaciju zaslona, boju naslovne trake i boju teme te sami način pokretanja [13].

Za komunikaciju između korisnika, usluga i poslužitelja na kojemu je smještena progresivna web aplikacija potrebna je sigurna internetska veza putem šifriranog 'HTTPS' (engl. HyperText Transfer Protocol Secure) [14] prometa koristeći 'TLS' (engl. Transport Layer Security) [15] kriptografsku tehnologiju, manifest datoteka sa potrebnim grafičkim ikonama i parametrima pokretanja web aplikacije, uslužna skripta za instalaciju te moderan internetski preglednik koji podržava 'Service Worker API' sučelje poput internetskog preglednika Google Chrome, Mozilla FireFox i drugih.

Slika br. 2: Zaslon instalirane igre kao PWA na računalu putem internetskog preglednika Google Chrome

Slika br. 3: Zaslon instalirane igre kao PWA na mobilnom uređaju putem internetskog preglednika Google Chrome

Slika br. 4: Početni zaslon Android WebView aplikacije igre

Ideja i razvoj

Na vijestima nerijetko slušamo ili čitamo o prirodnim katastrofama i nekako nam se doima da nas se ne tiču ili da mi nismo krivci. Nažalost to baš i nije tako. Potaknuti činjenicom i situacijom u svijetu po pitanju gospodarenja otpadom, pri čemu se nepravilnim zbrinjavanjem otpada onečišćuje ne samo okoliš, već se dovodi u pitanje zdravlje nas ljudi i životinjski svijet, kako bismo krenuli i naučili pravilno gospodariti otpadom od najmlađih dana, izrađena je jednostavna edukativna igra za djecu o pravilnom razvrstavanju i odlaganju otpada. Važno je da usvojimo naviku odvajanja otpada od malih nogu kako bismo očuvali prirodu. Scenski dio ovog rada predstavlja dvodimenzionalna ploha na kojoj se nalazi pozadina za vizualni dojam, zatim područje prikazane vrste otpada za odlaganje u odgovarajući spremnik otpada te spremnici za odlaganje otpada.

Glavna ideja ove igre je naučiti igrača koju vrstu otpada odlagati u koji spremnik za otpad. Na samom početku spremnici otpada imaju zadane oblike, dok se na njima kasnije, ovisno o radnji, pojavljuje emocija i gesta. Igra naizmjenično prikazuje otpad u gornjem lijevom obojanom krugu. Korisnik mišem ili prstom prikazani otpad mora odvući i pridružiti (spustiti) u odgovarajući spremnik za otpad za koji smatra da taj otpad pripada.

Ako je korisnik otpad spustio u točan spremnik, pojavljuje se obavijest u obliku oblačića s porukom 'Odlično' te spremnik poprima emociju 'sretan', a

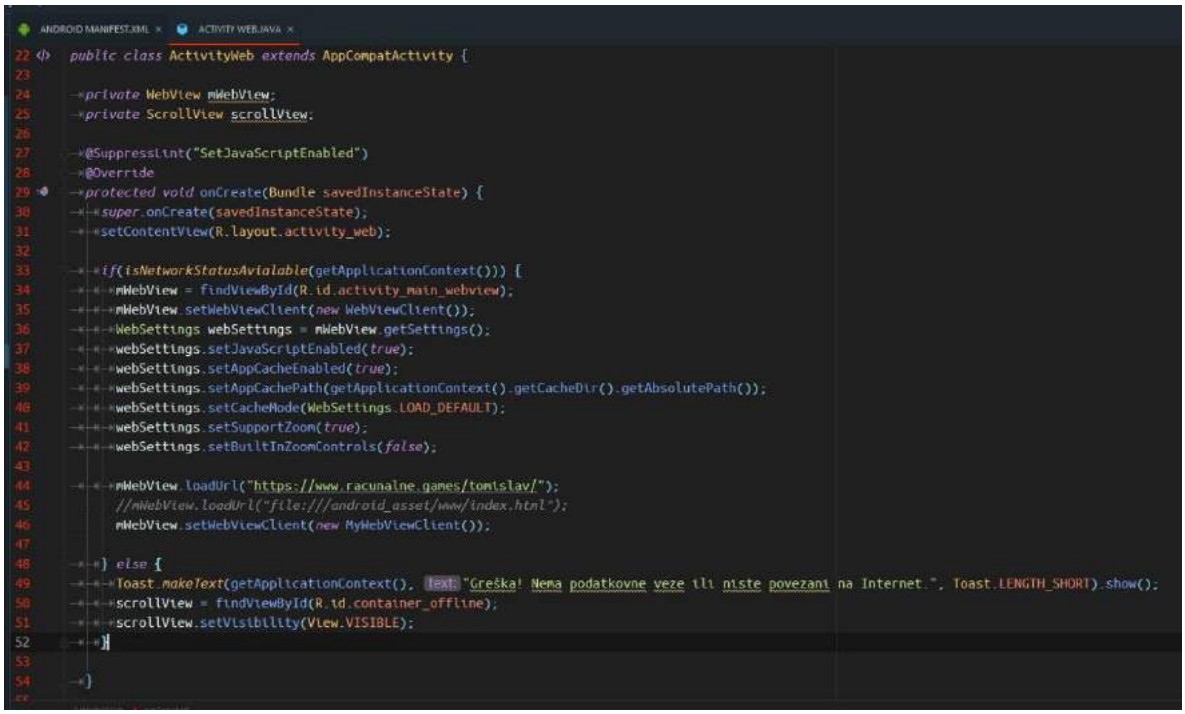
ako je radnja pogrešna tada se pojavljuje obavijest u obliku oblačića s porukom 'Pogrešno' te navedenim točnim spremnikom za otpad, a spremnik poprima emociju 'ljutito'. Otpad spušten u pogrešan spremnik animacijom vibracije vraća se na početni položaj. Igrač ponovno mora otpad pridružiti odgovarajućem spremniku otpada. Igrač otpad mora pridružiti točnom spremniku otpada kako bi mu se prikazao sljedeći otpad te kako bi nastavio s igrom.

Početno bodovno stanje iznosi 0. Po točno odloženom otpadu, igraču se pribraja 10 bodova. Negativnih bodova za pogrešno odloženi otpad nema. Maksimalan iznos prikupljenih bodova tijekom igre iznosi 340 bodova. Po završetku igre, ispisuje se obavijest u obliku oblačića s porukom 'Bravo' te se navodi rečenica o naučenom odlaganju otpada i primjeni stečenog znanja igranja ove igre u svakodnevnom životu.

Slika br. 5: Sučelje za izradu igre u alatu Construct 2

Mogućnosti za više platformi

Igru je moguće odigrati i instalirati kao progresivnu web aplikaciju na vlastitom uređaju putem internetske stranice igre [16]. Također, igru je moguće instalirati kao 'native' aplikaciju na mobilne uređaje s Android OS-om putem poveznice [17], dok se sama igra otvara putem 'WebViewClient' [18] komponente. Video snimke igre na različitim platformama može se pogledati putem poveznice [19].



Slika br. 6: Dio koda Android aplikacije (WebViewClient)

Zaključak

Općenito, razvoj računalnih igara zahtjeva mnogo znanja te je vrlo izazovan posao koji obuhvaća znanja iz različitih područja, od psihologije, dizajna i programiranja, pa sve do animiranja i osmišljavanja priče na zadovoljstvo igrača. Međutim, to tako i ne mora biti ako je riječ o znatiželji, entuzijazmu te se uz malo dobre volje može naučiti i isprobati nešto novo i drugačije.

U ovom radu opisan je razvoj računalne igre koja predstavlja spoj mobilne aplikacije u prirodnom 'native' okruženju i mobilnog weba – a upravo to progresivne web aplikacije (PWA) predstavljaju. Zahvaljujući današnjim modernim web tehnologijama, krajnji korisnici (igrači) imaju priliku web aplikacije i igre pokretati čak i u internetskom pregledniku koji se nalazi na njihovim mobilnim uređajima, računalima i drugim uređajima koji su povezani na Internet.

Najveća prednost progresivne web aplikacije je upravo to što neće zauzeti mnogo diskovnog prostora na vašem Android ili iOS mobilnom uređaju, a radit će gotovo identično kao i obična aplikacija.

U svega nekoliko klikova mišem te uz što manje troškova i uz što manje uloženog vremena za učenje i razvoj, izraditi jednostavnu računalnu igru za web platformu danas je brže i lakše pomoću alata Construct 2.

Reference

[1] Canvas API – Web APIs | MDN (18. rujna 2019.). Dostupno na: https://www.developer.mozilla.org/en-US/docs/Web/API/Canvas_API
[2] HTML Standard (18. rujna 2019.). Dostupno na: https://html.spec.whatwg.org/multipage/
[3] JavaScript Tutorial (18. rujna 2019.). Dostupno na: https://www.w3schools.com/js/
[4] Make Your Own 2d Games With Construct 2 (18. rujna 2019.). Dostupno na: https://www.scirra.com/construct2/
[5] Cascading Style Sheets (18. rujna 2019.). Dostupno na: https://www.w3.org/Style/CSS/
[6] jQuery (18. rujna 2019.). Dostupno na: https://jquery.com/
[7] Download Android Studio and SDK tools | Android Developers (18. rujna 2019.). Dostupno na: https://developer.android.com/studio
[8] JSON (18. rujna 2019.). Dostupno na: https://www.json.org/
[9] 4.2 Uvod u Progressive Web Apps (18. rujna 2019.). Dostupno na: https://support.google.com/google-ads/answer/7336531?hl=hr
[10] Service Workers: an introduction | Web Fundamentals | Google Developers (18. rujna 2019.). Dostupno na: https://developers.google.com/web/fundamentals/primers/service-workers/
[11] Application programming interface - Wikipedia? (18. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Application_programming_interface
[12] Push API – Web APIs | MDN (18. rujna 2019.). Dostupno na: https://developer.mozilla.org/en-US/docs/Web/API/Push_API
[13] 4.2.1. Zašto razvijati Progressive Web Apps? (18. rujna 2019.). Dostupno na: https://support.google.com/google-ads/answer/7336531?hl=hr
[14] Secure your site with HTTPS (18. rujna 2019.). Dostupno na: https://support.google.com/webmasters/answer/6073543?hl=en
[15] CARNet CERT - TLS protokol CCERT-PUBDOC-2009-03-257 (18. rujna 2019.). Dostupno na: https://www.cert.hr/wp-content/uploads/2009/03/CCERT-PUBDOC-2009-03-257.pdf
[16] Web igra autora (18. rujna 2019.). Dostupno na: https://www.racunalne.games/tomislav/
[17] Android igra – preuzimanje (18. rujna 2019.). Dostupno na: https://www.racunalne.games/tomislav/apk/igra.apk
[18] WebViewClient | Android Developers (18. rujna 2019.). Dostupno na: https://developer.android.com/reference/android/webkit/WebViewClient
[19] Video snimke autora igrice (18. rujna 2019.). Dostupno na: https://drive.google.com/open?id=1-HFiKp6KsRywlumsZWMYkb2RpnpNcZI0

Stanje industrije proizvodnje videoigara u Republici Hrvatskoj

Mario Konecki, Davor Švaić i Andrej Kovačević
Fakultet organizacije i informatike, Sveučilište u Zagrebu
Akademija dramske umjetnosti, Sveučilište u Zagrebu
Exordium Games d.o.o.
mario.konecki@foi.hr, dsvaic@adu.hr, andrej@exordiumgames.com

Sažetak

Industrija proizvodnje videoigara postala je iznimno važna i jaka industrija na globalnoj razini. Međutim, iako su mnogi svjesni ove činjenice, manje je onih koji znaju da i industrija proizvodnje videoigara u Hrvatskoj raste i ostvaruje značajne uspjehe na svjetskoj razini.

U ovom radu bit će dan osvrt na stanje industrije proizvodnje videoigara u Hrvatskoj i svijetu, kao i osvrt na obrazovnu podršku ovoj industriji u kojoj se svake godine traže novi zaposlenici koji će omogućiti nastavak sve brže ekspanzije niza hrvatskih pouzeća koja se bave razvojem videoigara.

Ključne riječi: videoigre, proizvodnja videoigara, industrija videoigara, Klaster hrvatskih proizvođača računalnih igara

Uvod

Proizvodnjom videoigara u Hrvatskoj pojedinci i poduzeća bave se već 30-ak godina. Veteran hrvatskih videoigara je studio Croteam koji je svjetski renome izgradio na franšizi Serious Sam. Od tada je domaća scena razvoja videoigara narasla na oko 500 djelatnika, 60-ak poduzeća i prihode od preko 400 milijuna kuna godišnje.

Tijekom zadnjih 5 godina ostvaren je veliki napredak unutar djelatnosti proizvodnje videoigara te je ostvaren svjetski značajan uspjeh kroz videoigre poduzeća Nanobit, Croteam, Gamepires, Pine Studio i drugih hrvatskih studija za proizvodnju videoigara. Radi što bržeg razvoja djelatnosti, više poduzeća se udružilo u Klaster hrvatskih proizvođača računalnih igara (engl. CGDA - Croatian Game Developers Association). Putem Klastera se jača vidljivost i radi se na poboljšanju uvjeta djelatnosti razvoja videoigara.

Djelatnost razvoja videoigara je nastavila svoj rast i njezin značaj postaje sve veći kako u Hrvatskoj tako i u Europi, ali i na globalnoj razini [1; 2; 3].

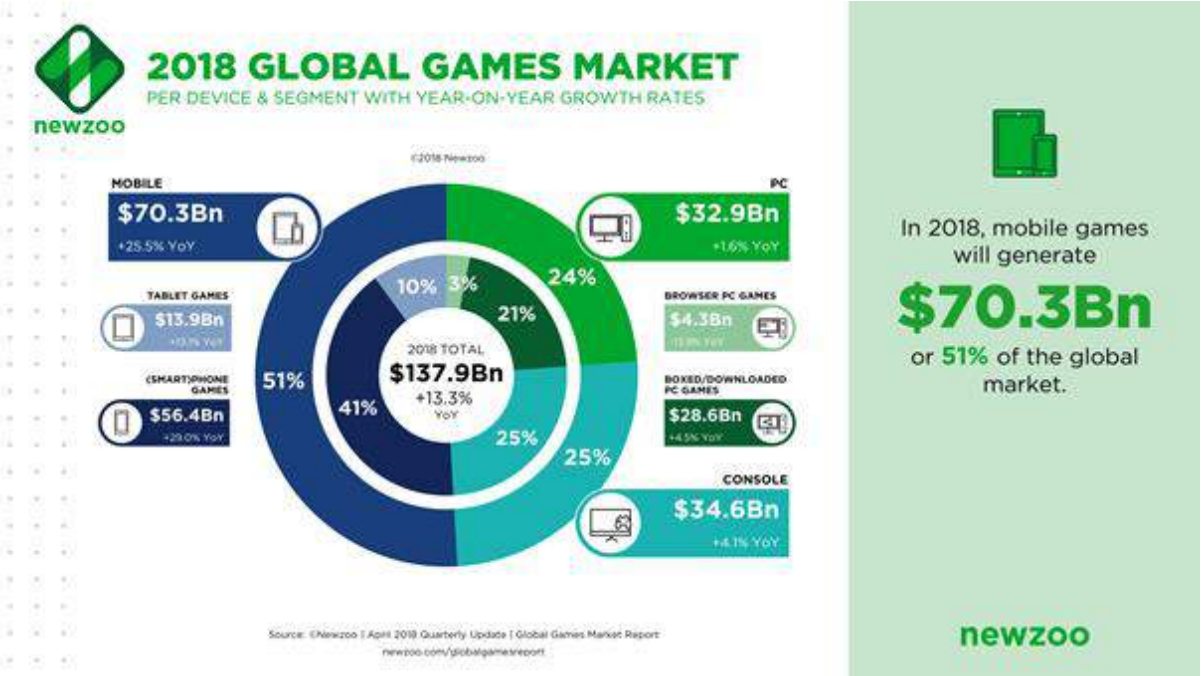
Videoigre su uvrštene u Zakon o audiovizualnim djelatnostima [4], a Hrvatski audiovizualni centar (HAVC) je najavio javne pozive za potporu razvoju i proizvodnji videoigara.

Neki od izazova djelatnosti koji su i dalje aktualni su dvostruko oporezivanje prihoda ostvarenih u SAD-u, manjak poreznih olakšica kako za djelatnost tako i za vanjske investicije te sustavna podrška daljnjem razvoju djelatnosti [5].

Pokazatelji stanja globalnog i hrvatskog tržišta

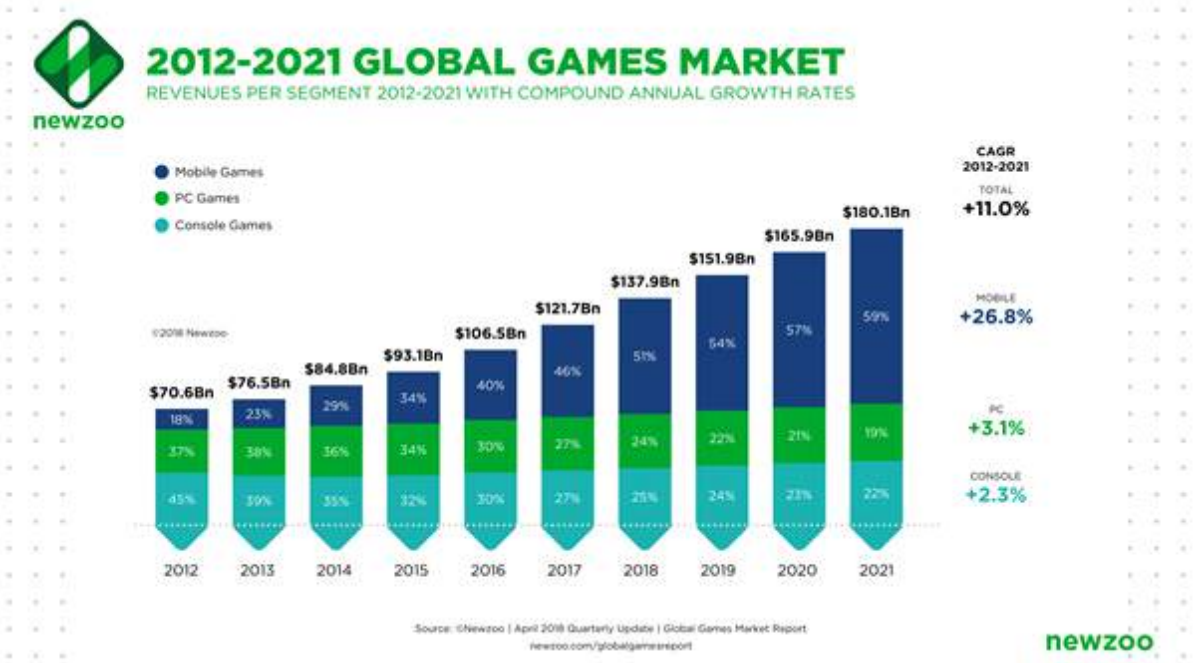
Osnovni pokazatelji globalne industrije videoigara, koja je pretekla filmsku i glazbenu zbirno, navedeni su u nastavku [1]:

- globalni prihodi za 2018. se procjenjuju na 137,9 milijardi USD,
- godišnja stopa rasta od 11% nije bila pod utjecajem globalne krize,
- projekcija globalnih prohoda do 2020. je 165,9 milijardi USD.



Slika br. 1: Pregled prihoda po segmentima za 2018.

Navedeni pokazatelji govore o doista velikom potencijalu industrije proizvodnje videoigara i mogućnosti utjecaja ove industrije na gospodarstvo Republike Hrvatske u kojem su informatičke usluge već sada važan dio sveukupnog izvoza. Podrška industriji proizvodnje videoigara u smislu odgovarajuće legislative, mogućnosti financiranja, kao i mogućnosti obrazovanja i isticanja važnosti ove industrije neophodna je za njezin daljnji razvoj.



Slika br. 2: Pregled prihoda kroz godine

Prvi sistematizirani uvid u stanje industrije videoigara omogućila je projektna studija "Mapiranje kreativnih i kulturnih industrija u Republici Hrvatskoj", koju je 2015. izradio Ekonomski institut u Zagrebu [6]. Mapiranje je pokazalo da je u Hrvatskoj tada poslovalo desetak trgovačkih društava specijaliziranih za proizvodnju računalnih igara i videoigara, koja su većinom osnovana kao mikro i mala poduzeća.

Tablica 5. Pokazatelji poslovanja vodećih proizvođača računalnih igara, 2009., 2011. i 2013. godina

	2009.	2011.	2013.	% promjena 2013./2009.	Prosječna godišnja stopa rasta 2009.- 2013.
Broj zaposlenih	19	30	57	200,0%	31,6%
Prosječni broj zaposlenih	2,11	3,3	6,3	200,0%	31,6%
Ukupni prihodi (tis. kn)	4.103	8.529	20.970	311,1%	50,4%
Proizvodnost rada (tis. kn)	215,9	284,3	367,9	70,4%	14,2%
Dobit prije oporezivanja (tis. kn)	1.133	-123	6.581	380,7%	53,7%
Izvoz (tis. kn)	2.790	6.637	17.577	430,1%	58,4%
Udio izvoza u prihodima	68,0%	77,8%	83,8%	22,1%	5,1%
Saldo izvoza i uvoza (tis. kn)	2.790	6.637	17.577	530,1%	58,7%

Napomena: dani su izračuni na temelju podataka devet vodećih tvrtki proizvođača računalnih igara.
Izvor: izračun autorica prema podacima Poslovne Hrvatske.

Slika br. 3: Pokazatelji poslovanja dostupni u okviru studije "Mapiranje kreativnih i kulturnih industrija u Republici Hrvatskoj"

Mapiranje je utvrdilo i pokazatelje prosječnih stopa rasta u razdoblju od 2009. do 2013. (prema Poslovnoj Hrvatskoj), a prema kojima je [6]:

- vidljiv rast broja zaposlenih od 31,6% godišnje,
- vidljiv prosječni rast ukupnih godišnjih prihoda od 50,4% godišnje,
- izvoz rastao za prosječno 58,4% godišnje.

S obzirom na to da je mapiranje rađeno na vrlom malom broju poslovnih subjekata apsolutni iznosi predstavljaju ponešto manjkave pokazatelje, dok relativne stope pružaju bolji uvid u trendove i kretanja. Tako je Hrvatska već u razdoblju 2009. - 2013. imala višestruko veće rezultate u odnosu na svjetske stope rasta.

Podrška industriji proizvodnje videoigara u području obrazovanja

Povijest i tradicija videoigara sežu otprilike 40 godina unazad. Industrija videoigara je u tih 40 godina narasla do takvih razmjera da danas po ostvarenom prometu premašuje ostvareni promet industrije filmova i glazbe zajedno [7]. Po broju uključenih osoba – u poduzećima proizvođačima videoigara i po broju publike (igrača), predstavlja najveću zajednicu u području industrije zabave.

Razvoj ove industrije stvorio je specifičan kulturni i ekonomski kontekst, koji je u posljednjih 20 godina postao predmetom interesa mnogih obrazovnih i znanstvenih institucija koje su ili specifično obrazovale studente i polaznike za područje videoigara, ili su započele s promatranjem područja videoigara kroz znanstveno-analitički teorijski diskurs.

Prva funkcija obrazovnih institucija, ona koja ima za svrhu prenošenje specifičnih znanja i vještina u području videoigara te služi kao potpora rastu i razvoju industrije pripremajući novi kadar za buduću profesiju, započela je početkom 2000-tih na sjevernoameričkim fakultetima, a gotovo deset godina kasnije i na europskim [8].

Bavljenje područjem videoigara kroz znanstveni i istraživački rad započelo je nešto ranije – krajem 1990-tih godina, angažmanom istraživača i znanstvenika drugih područja u različitim aspektima područja videoigara, čime su odgovarali na širi profesionalni i javni interes, odnosno potrebu da se fenomen igranja videoigara znanstveno obradi [9].

Obrazovanje i industrija videoigara u Republici Hrvatskoj tek su nedavno prepoznali potrebu i nužnost povezivanja kako bi se promislio modalitet stvaranja kapaciteta za provođenje obrazovnih programa koji bi podržali i održali trendove rasta ove industrije.

U Hrvatskoj se je donedavno bilo moguće obrazovati u području razvoja videoigara putem specijalističkih tečajeva u privatnoj akademiji Machina i na visokom učilištu Algebra koje provodi stručni specijalistički diplomski studij Razvoj računalnih igara. No, tijekom 2018. više inicijatora počelo je razmišljati o potrebi uvođenja dodatnih mogućnosti obrazovanja u području razvoja videoigara.

Slijedom navedenog, tijekom 2018. počeo je rad na srednjoškolskom četverogodišnjem obrazovnom programu "Tehničar za razvoj videoigara". Inicijativu za izradu ovog srednjoškolskog programa dala je Sisačko-moslavačka županija i Razvojna agencija Sisačko-moslavačke županije – SiMoRa, a projekt izrade samog obrazovnog programa vodio je doc. dr. sc. Mario Konecki. Obrazovni program "Tehničar za razvoj videoigara" izrađen je u okviru Laboratorija za dizajn programskih sučelja, internetske servise i videoigre koji djeluje na Fakultetu organizacije i informatike Sveučilišta u Zagrebu, u suradnji sa stručnjacima iz javnog i privatnog sektora.

Srednjoškolski obrazovni program "Tehničar za razvoj videoigara" dovršen je tijekom 2019. i odobren od strane Ministarstva, znanosti i obrazovanja kao eksperimentalni program 2. svibnja 2019.

Eksperimentalni srednjoškolski program počeo se provoditi u školskoj godini 2019./2020. u Tehničkoj školi Sisak, a interes za upis je bio višestruko veći od broja učenika koje je bilo moguće primiti.

U prvoj polovini 2019. započeo je rad na izradi studijskog programa za razvoj videoigara u okviru projekta Hrvatskog kvalifikacijskog okvira (HKO) naziva Edu4Games – Izrada standarda zanimanja i kvalifikacija te novih studijskih programa za područje dizajna i razvoja videoigara. Projekt je financiran je iz Europskog socijalnog fonda (ESF).

Nositelj projekta je Akademija dramske umjetnosti, a partneri na projektu su Akademija likovnih umjetnosti, Arhitektonski fakultet – Studij dizajna, Fakultet elektrotehnike i računarstva i Fakultet organizacije i informatike. Voditelj projekta je prodekan za međunarodnu, međufakultetsku i međusveučilišnu suradnju Akademije dramske umjetnosti izv. prof. art. Davor Švaić.

Cilj projekta Edu4Games je razvoj novog diplomskog studijskog programa naziva "Dizajn i razvoj videoigara" s četiri usmjerenja i četiri programa cjeloživotnog učenja: dizajner videoigara, programer videoigara, producent videoigara i umjetnik videoigara. Projekt okuplja mnoge nastavnike i renomirane stručnjake iz područja razvoja videoigara koji će biti zaduženi za pojedina područja novog studijskog programa. Predviđeno trajanje projekta je

36 mjeseci, nakon čega bi studentima trebalo biti omogućeno i visokoškolsko obrazovanje u području videoigara.

U svrhu podrške poduzetničkim aktivnostima u području razvoja videoigara osnovan je i poduzetnički inkubator PISMO u Novskoj [10] koji mladim poduzetnicima uz uredske i poslovne prostore nudi i financijsku potporu, kao i kompletnu tehnološku infrastrukturu i opremu potrebnu za razvoj novih i konkurentnih proizvoda u području videoigara.

Trendovi obrazovanja u području razvoja videoigara

Nakon gotovo 20 godina provođenja različitih modaliteta obrazovanja, obrazovnih formata i programa, prilagođavanih brzom rastu i razvoju industrije videoigara koja je izašla iz okvira industrije zabave i ušla u najrazličitije grane ljudskog djelovanja, obrazovanje u području videoigara danas zahtjeva interdisciplinarnost i tematsku širinu u znanstvenom, kulturnom i ekonomskom kontekstu.

Uska specijalizacija u smislu obrazovanja za specifične poslove unutar područja videoigara, uz sve izazove koje tržište rada danas stavlja pred studente koji završe svoje visokoškolsko obrazovanje, više ne daje dobre rezultate ni s gledišta studenata ni s gledišta poslodavaca [11].

Alumni s usko specijaliziranim znanjima i vještinama teško mijenjaju radna mjesta u trenutku profesionalnog zasićenja, dok poslodavci često imaju potrebu za djelatnicima različitih znanja, uz pomoć kojih mogu obavljati različite poslove unutar poduzeća i kvalitetnije komunicirati unutar projektnog tima.

Današnji studijski programi u području videoigara moraju obrazovati buduće profesionalce koji će biti pripremljeni za rad kod poslodavaca, ali koji će isto tako biti sposobni i osnivati vlastita poduzeća, voditi vlastite projekte i zapošljavati te sudjelovati u poslovima i projektima u drugim područjima.

Istraživanje HEVGA-e (Higher Education Video Game Alliance - udruženje visokoškolskih ustanova koje izvode studijske programe za područje videoigara) iz 2019. [12] ukazuje na činjenicu kako više od trećine diplomiranih studenata koji su završili studijske programe iz područja industrije videoigara radno mjesto pronalazi u drugim industrijama.

APPENDIX

TABLE A3: *Industry of employment reported by respondents in non-video game industries*

Current Position	% of Respondents
Advertising	1%
Business and Finance	1%
Construction	1%
Consulting	2%
Customer Service	1%
Education	31%
Energy	1%
Entertainment	4%
Food Industry	6%
Government, Security and Defence	9%
Health Care	3%
Insurance	1%
Leisure	1%
Lounge	1%
Marketing	1%
Media and Entertainment	3%
Non Profit	1%
Sales	1%
Technology	30%
Veterinary	1%

Slika br. 4: Druge industrije u kojima studenti studijskih programa u području videoigara pronalaze posao

Zanimljiv podatak koji se ističe u tablici prikazanoj na slici 4 je da 30% studenata koji su završili studijski program u području videoigara posao pronalazi u obrazovanju - što je s obzirom na svakim danom sve veći broj novih obrazovnih institucija i programa razumljivo, no isto tako upućuje na nužnost pripreme studenata i za tu vrstu poslova.

Anketa HEVGA-e iz 2014. – 2015. [13] daje pregled ukupno 240 kolegija koji se izvode u sklopu 29 studijskih programa iz područja videoigara na fakultetima članovima HEVGA-e [14], dok ranije spomenuto istraživanje iz 2019. navodi kolegije prema broju završenih studenata koji su iste odslušali tijekom studiranja.

Popis kolegija u tablici jasno upućuje na područja u kojima su studenti stekli dodatna znanja i vještine za rad, uz specijalistička znanja vezana za područje videoigara - Igre i društvo, Istraživanje u području videoigara, Interaktivno pripovijedanje / Kreativno pisanje, Igre i učenje, Serious gaming, Igrifikacija itd.

APPENDIX

TABLE A1: *Total number of courses taken by respondents arranged by frequency*

Course Name	Total
Game Design	284
Game Production	215
Game Programming	208
3D Modeling	188
Animation	164
Level Design	151
Project Management	146
Games and Society	132
Game Research	131
Interactive Storytelling/Creative Writing	129
Critical Game Studies	128
Project-Based Learning	127
Games and Learning	124
Business of Gaming	123
Graphics	115
Game Engine Scripting	113
Serious Games	102
Audio Design	97
Gamification	92
Game AI	90
Visual Design	88
Virtual Reality/ Augmented Reality	83
Quality Assessment	82
Concept Art	79
Technical Writing	67
Fine Art	53
Music	50
Data Analytics	42
Game Platform Hardware Architecture	36

Slika br. 5: Kolegiji studijskih programa koje su slušali studenti studijskih programa u području videoigara, prema broju anketiranih studenata koji su pojedini kolegij odslušali

Aktualni trend obrazovanja u području videoigara su studijski programi u okviru kojih se videoigre ne promatraju samo kao oblik zabave, već kao medij koji svoju primjenu i funkciju može imati u najrazličitijim područjima i aktivnostima društva.

Zaključak

Industrija proizvodnje računalnih igara predstavlja važan segment svjetskog gospodarstva, ali isto tako i hrvatskog gospodarstva. U Republici Hrvatskoj industrija proizvodnje videoigara raste i ima potencijal postati značajan dio sveukupnog izvoza Republike Hrvatske.

Da bi se navedeno moglo ostvariti potrebno je stvoriti adekvatan zakonski okvir i osigurati odgovarajuću potporu razvoju kapaciteta, a važno je ponuditi i adekvatne modele financiranja daljnjeg razvoja područja. Dio navedenih preduvjeta daljnjeg razvoja industrije videoigara je i razvoj kvalitetnog sustava obrazovanja utemeljenog na potrebama industrije i na aktualnim globalnim trendovima.

Trendovi i prakse u međunarodnom obrazovnom kontekstu za područje videoigara ukazuju na nužnost promišljanja koje će dovesti do toga da studijski programi budu kreirani na način da ispunjavaju sve tri misije visokog obrazovanja i da cjelovito obrazuju buduće profesionalce u sektoru na način da oni mogu doprinositi razvoju područja i biti osviješteni i obrazovani članovi društva koji će svojim znanjima i vještinama doprinositi javnom interesu i napretku društva.

Reference

[1] 2018 Global Games Market Report, https://cdn2.hubspot.net/hubfs/700740/Reports/Newzoo_2018_Global_Games_Market_Report_Light.pdf, posjećeno 10.9.2019.

[2] Games Funding Guide 2016, Creative Europe Desk Denmark, https://www.creative-europe-desk.de/downloads/Games%20Funding%20Guide%202016.pdf, posjećeno 10.9.2019.

[3] Enabling Digital Growth with EU Funding, EGDF, http://www.egdf.eu/wp-content/uploads/2018/04/201801-EuropeanFundingInstruments.pdf, posjećeno 10.9.2019.

[4] Zakon o audiovizualnim djelatnostima, https://www.zakon.hr/z/489/Zakon-o-audiovizualnim-djelatnostima, posjećeno 10.9.2019.

[5] Tax Incentives for the Audio Visual Industry, https://link.springer.com/chapter/10.1007%2F978-981-287-832-8_8

[6] Mapiranje kulturnih i kreativnih industrija u RH, Ekonomski Institut, http://hkkkki.eu/dokumenti/mapiranje.pdf, posjećeno 10.9.2019.

[7] The Video Games' Industry is Bigger Than Hollywood, https://lpesports.com/e-sports-news/the-video-games-industry-is-bigger-than- hollywood, posjećeno 10.9.2019.

[8] Freyermuth, Gundolf S., (2015.), *Games | Game Design | Game Studies: An Introduction, (str. 235 – 263. Academization and Aestetic Production), CCR Writers Inc.*

[9] Mäyrä, Frans, (2009.), *Getting into the Game: Doing Multi – Disciplinary Game Studies*, (str. 3 – 7.), authors version: https://people.uta.fi/~frans.mayra/Mayra_Multidisciplinary_Game_Studies_2009.pdf, posjećeno 10.9.2019.

[10] Poduzetnički inkubator PISMO, http://inkubator-pismo.eu, posjećeno 10.9.2019.
[11] Czauderna, Andre, (2018.), Academic Game Design Education: A Comparative Perspective, JCSG 2018. (str. 9 – 12.), Springer Nature Switzerland
[12] 2019 Survey Of Program Graduates, HEVGA, https://hevga.org/wp-content/uploads/2019/03/HEVGA_2019_Survey_of_Program_Graduates.pdf, posjećeno 10.9.2019.
[13] Our State of Play, HEVGA, https://hevga.org/wp-content/themes/hevga2wp/assets/our-state-of-play-2014-15%20.pdf, posjećeno 10.9.2019.
[14] HEVGA, https://hevga.org, posjećeno 10.9.2019.

Trip the Ark Fantastic kao Gesamtkunstwerk

Aleksandar Gavrilović i Matija Vigato
Gamechuck
Visoko učilište Algebra
gamechuckdev@gmail.com, matijavigato@gmail.com

Sažetak

Gesamtkunstwerk je pojam u estetici kojim se označava umjetničko djelo koje teži sintezi više oblika umjetnosti. Videoigre kao suvremeni oblik umjetnosti imaju potencijal za sintezu, pored igre kao snažnog distinktivnog elementa, i elemenata književnosti, glazbe, igranog filma, animiranog filma, videa, stripa, arhitekture, skulpture, slikarstva i dr. Međutim, njihov sintetički potencijal često je zanemaren u kritici videoigara u kojoj se često prednost daje elementu igre, te u estetskoj procjeni gdje se najčešće odabire metodološki pristup teorije filma. Isto tako, sintetičnost videoigara često je zanemarena u njihovoj izradi. Studio za razvoj videoigara Gamechuck u svojoj igri *Trip the Ark Fantastic* nastoji primijeniti principe *Gesamtkunstwerka*, a ovaj rad će predstaviti plan istog, analizirajući pojavnost prisutnih oblika umjetnosti u sintezi djela.

Ključne riječi: *Trip the Ark Fantastic, Gesamtkunstwerk*, umjetnost, videoigre, estetika, razvoj videoigara

Uvod

U ovom radu objasnit će se koncept *Gesamtkunstwerka*, razmotriti posebnost i problemi u ostvarenju tog koncepta u videoigrama kao obliku umjetnosti, a u konačnici predstaviti nastojanja u postizanju istog u izradi videoigre *Trip the Ark Fantastic*. Tri principa *Gesamtkunstwerka* koja će se predstaviti su 1. poklapanje narativa s audiovizualnim elementima, 2. lajtmotivi i 3. poklapanje narativa s mehanikom igre. Na kraju će se ukazati i na neke izazove u izradi *Gesamtkunstwerk* videoigara.

Gesamtkunstwerk

Gesamtkunstwerk (njem. cjelovita umjetnost) je "umjetničko djelo koje nastaje ravnopravnom i naglašenom suradnjom više umjetnosti" (*Hrvatski jezični portal*: s.v. "Gesamtkunstwerk"). Termin je prvi put upotrijebio Karl Friedrich Eusebius Trahndorff 1827. u eseju *Estetika ili nauka o svjetonazoru i*

umjetnosti[1] (*Ästhetik oder Lehre von Weltanschauung und Kunst*). Pojam je popularizirao skladatelj Richard Wagner u svojim teorijskim djelima *Umjetnost i religija* (*Die Kunst und die Religion*), *Umjetnost budućnosti* (*Das Kunstwerk der Zukunft*) i *Opera i drama* (*Oper und Drama*)[2]. Wagner tvrdi da su u grčkoj tragediji (primjerice kod Eshila) riječ, glazba i ples bili u savršenoj harmoniji. prema Wagneru, padom atenskog polisa umjetnosti su se počele razilaziti (*Zersplitterung der Kunste*, njem. fragmentacija umjetnosti) stoga ih on nastoji ponovno ujediniti u svojoj operi. Pritom mu je jedna od inspiracija Baudelaireov koncept sinestezije prema kojem harmonično djelovanje osjetila vodi intenzivnijem iskustvu djela (Wolfman, 2013).

Ideje *Gesamtkunstwerka* provlačile su se kroz mnogo umjetničkih škola i pokreta 19. i 20. stoljeća, kao što su Arts and Crafts pokret, škola Bauhaus (osnivač Walter Gropius i dr.), radionica Wiener Werkstätte, skupina *Die Brücke* i dr. Svih njih povezuje težnja ka umjetničkom okruženju u kojemu se sve umjetnosti udružuju.

Videoigre kao Gesamtkunstwerk

Videoigre kao oblik umjetnosti[3] u sebi sintetiziraju igru zajedno s cijelim nizom elemenata raznih umjetničkih oblika: slikarstva, glazbe, književnosti, skulpture, video-umjetnosti[4], igranog filma[5], animiranog filma, stripa[6], arhitekture i dr. Film, također novi oblik umjetnosti dvadesetoga stoljeća, već je razmatran kao *Gesamtkunstwerk* (Birdsall, 2012). U manifestu *Rođenje šeste umjetnosti* Ricciotto Canudo proglasio je film novom umjetnošću kao sinteze pet antičkih umjetnosti izdvojenih kod Hegela, a to su: 1. arhitektura, 2. skulptura, 3. slikarstvo, 4. glazba i 5. poezija (Hegel, 2011). Videoigre, uz uključenje igre, mogu se smatrati nastavkom sinteze umjetnosti započete u filmu, i to u novom, još kompleksnijem obliku.

Međutim, najveći izazov videoigrama nije sinteza elemenata raznih oblika umjetnosti međusobno, već njih u odnosu spram igre. Igra se na prvi pogled može činiti uljezom u strukturi videoigara kao *Gesamtkunstwerka*. Međutim, ni djela primijenjene umjetnosti Bauhausa i ranije spomenutih škola i pokreta nemaju isključivo umjetničke elemente, a opet zadovoljavaju *Gesamtkunstwerk*

[1] Prev. M.V.
[2] Naslove prev. M.V.
[3] Više o videoigrama kao obliku umjetnosti u: (Vigato, 2019).
[4] Neke videoigre (npr. *Her story*) nisu primarno sastavljene od animacija već od filmskih snimki.
[5] Česta je, primjerice, upotreba tzv. *lens flare*-a, fenomena u kojem se svjetlo raspršuje u sistemu leća, često kao reakcija na snažno svjetlo, proizvodeći artefakte unutar slike, a koji je ovisan o materijalnoj leći i karakterističan za film.
[6] Studio Gamechuck izdao je nekoliko videoigara u formi interaktivnog stripa, kao što su *vApe Escape* i *All You Can Eat*.

principe. Iako oblici umjetnosti čiji su elementi uključeni u pojedine videoigre variraju, igra je nužan element svake igre. Na primjer, *Spider in Web* je isključivo tekstualna igra, dok je *Limbo* gotovo u potpunosti bez teksta, ali obje su videoigre, dok bilo koja videoigra koja isključuje element igre automatski ispada iz vlastite kategorije, primjerice u kategoriju interaktivnog filma ili interaktivne drame. Videoigrama se često upravo na temelju elementa igre spočitava umjetnička legitimacija na što Grant Tavinor odgovara usporedbom videoigara s čudnovatim kljunašem (Tavinor, 2009). On liježe jaja, a nakon toga prelazi na dojenje, čime pokazuje do tada teško zamisliv spoj sisavca i oviparne životinje. Isto tako, videoigre u područje umjetnosti uvode igru kao do tada teško zamislivi gradivni element. Stoga, u kontekstu *Gesamtkunstwerka* igra jednako sudjeluje u sintezi kao i umjetnički elementi videoigre.

Prožimanje igre i narativa jedna je od burnih tema u teoriji videoigara. Mogu li se ta dva dijela prožimati ili su nužno izolirana? Tzv. ludolozi (lat. *ludus* - igra) i esencijalisti, smatraju da se videoigre ni ne mogu smatrati narativima. Jesper Juul tvrdio je da se videoigre mogu prepričati i da imaju neke narativne elemente kao što su *cutscenes*, ali da same po sebi nisu narativ. S druge strane, tzv. naratolozi smatraju da, iako narativ nije *raison d'être* videoigara, da je njihov važan element[7]. Ovaj rad zauzima naratološku perspektivu, tj. perspektivu prema kojoj se igra i narativ u videoigrama mogu, ali i trebaju prožimati.

Primjena *Gesamtkunstwerk* principa na videoigru *Trip the Ark Fantastic*

Trip the Ark Fantastic je videoigra koju producira studio Gamechuck i koja je osmišljena s idejama *Gesamtkunstwerka*, a ujedno je i prva, a i jedina hrvatska videoigra selektirana za financiranje od strane programa Europske unije za razvoj europskih videoigara u sklopu MEDIA fonda. Kao "totalno umjetničko djelo" ili *Gesamtkunstwerk*, Trip the Ark Fantastic pristupa svakoj vrsti umjetnosti koju koristi kao jednako važnoj za proučavanje tema kojima se igra bavi: od narativnih elemenata i elemenata igre do audiovizualnih elemenata, pa čak i elemenata dizajna sučelja.

Priča je basnolika alegorija te prati ježa znanstvenika u životinjskom carstvu koje nalikuje našem društvu u 19. stoljeću. Jež se silom prilika pronađe u situaciji da njegove odluke imaju mogućnost promjene čitavog životinjskog društva iz monarhije u demokraciju ili anarhiju putem otkrivanja laži u mitologiji životinjskog carstva (arke u koju vjeruju) te teorije evolucije, što u životinjskom svijetu može pokrenuti revoluciju. Filozofske teme koje se

[7] Žarište debate naratologa i ludologa bila je konferencija Digital Games Research Association (DiGRA) 2003. godine.

istražuju su odnos mitološke i znanstvene istine i njihov utjecaj na društvo, te pitanje najboljeg oblika društva: je li to monarhija, oligarhija, demokracija ili pak anarhija su teme o kojima likovi u priči često razgovaraju, dok se igraču ne nameće nijedna opcija već mu se daje mogućnost samostalne odluke.

Slika br. 1: Slika na zidu prikazuje društvenu hijerarhiju koja vlada u životinjskom carstvu igre

Poklapanje narativa s audiovizualnim elementima

Životinje nisu suviše antropomorfizirane i više podsjećaju na prave životinje nego na ljude, osim u slučajevima kada je to nužno (npr. šape služe i kao ruke da bi životinja mogla pisati knjigu). Takvo suzdržavanje od antropomorfizma koristi se prvensteno da igrač ne izgubi svijest o tome da se radi o životinjama, tj. da alegorija ne postane previše apstraktna, jer je evolucija bitan dio narativa igre, a njeno otkrivanje od strane znanstvenika njen narativni zaključak.

Vizualni identitet igre kao jedan od uzora uzima stare Disney crtiće, što se očituje u ručno crtanoj *frame-by-frame* animaciji i pejzažima punim boja i detalja u visokoj 4K rezoluciji. U dizajnu životinja nastoji se zadržati ozbiljnost kako bi se zadržala ozbiljnost samih tema kojima se igra bavi.

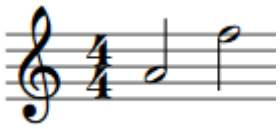
Slika br. 2: André, lik u igri, kao vepar je samo djelomično antropomorfiziran

Pošto društvo predstavljeno u igri nalikuje stvarnom društvu 19. stoljeća, neki elementi tog doba uključeni su u dizajn igre. Primjerice, u arhitekturi dominiraju drvo i kamen, a stil dizajna podsjeća na kombinaciju fantastičnih priča s viktorijanskom engleskom, dok je glazba je u stilu neoromantizma.

Lajtmotivi

Lajtmotiv (njem. *Leitmotiv*: vodeći motiv) je "glazbeni termin kojim se označava tema ili neka glazbena ideja koja kao sklop tonova predstavlja ili simbolizira neku osobu, predmet, mjesto, ideju, stanje duha ili nadnaravnu silu." (*Hrvatska enciklopedija*: s.v. "lajtmotiv"). Pojam se najviše veže za Wagnera koji je lajtmotiv koristio kao tehniku u ujedinjavanju djela. Primjerice, u svojoj operi *Tristan i Izolda* kao lajtmotiv za lik Tristana koristi tzv. "Tristanov akord" koji je bitonalan, tj. koji se sastoji od dva zvuka različitog tonaliteta, u ovom slučaju jedan u duru, a drugi u molu, čime se dočarava Tristanova podvojenost i unutrašnja promjenjivost (Wolfman, 2013). U videoigri *Trip the Ark Fantastic* isplanirano je oko 65 lajtmotiva vezanih za određene likove, mjesta, ali i događaje (npr. pretjerivanje lika u govoru ili otkrivanje nove informacije). Također, glazbena mjera popratne glazbe likova određena je njihovom pripadnošću određenoj kasti. Tako primjerice pojavu aristokracije i plemstva (lavovi i ostale mačke) prati tročetvrtinska mjera (3/4), građana (ptice) šesteroosminska (6/8), kmetova i *default* likova (sitne životinje) četveročetvrtinska (4/4), dok robove (štakori, gušteri, žabe i dr.) prati slobodna mjera, tj. ona koja nije unaprijed određena, što je primjerice karakteristično za jazz glazbu. Time glazba u igri nije samo atmosferična pratnja narativu već i samostalni narativni element jer ponekad daje

informacije van okvira dijaloga. Primjerice, lajtmotiv anarhije koji se može čuti u glazbenoj temi lika nagovješćuje igraču da taj lik pripada anarhističkoj društvenoj frakciji, iako to nije rečeno ni u dijalogu niti je igraču to saznanje u igri još dostupno.



Slika br. 3: Motiv za "Ark Fantastic" mit; instrumenti za izvedbu su gudači, limena glazba ili oboje

U videoigri *Trip the Ark Fantastic* lajtmotivi shvaćeni u širem smislu pojavljuju se i u grafičkom korisničkom sučelju pa tako sučelja s više uglova predstavljaju višu kastu. Inspirirano romanom Flatland, kvadratni okviri predstavljaju običan puk, okviri s više uglova predstavljaju višu kastu, dok su lavovi kao najviša kasta predstavljeni portretima u kružnom okviru.

Poklapanje narativa s mehanikom igre

U izradi videoigre *Trip the Ark Fantastic* također se velika pažnja posvetila prožimanju igre i narativa. To se posebno očituje u mehanici igre koja simulira različite znanstvene metode, što je u skladu s narativom prema kojemu je glavni lik znanstvenik koji razotkriva mitološke istine. Glavni lik po svijetu traži dokaze, bilo eksperimentalnim metodama (korištenje mikroskopa, kamere i slično), bilo društvenim metodama (razgovor s lokalnim stanovništvom pri čemu igra zahtijeva i učenje novih životinjskih jezika[8]), bilo istraživanjem prijašnjih radova (iščitavanje dokaza iz knjiga razasutim po kraljevstvu). Svi skupljeni dokazi koriste se kako bi se napisao znanstveni 'članak' koji se potom objavljuje u carstvu, a čija razine snage dokaza utječe na to koliko će društvo vjerovati u ono napisano članku te koliko će reputacija igrača kao znanstvenika rasti.

Izazovi izrade *Gesamtkunstwerk* videoigre

Kako bi se svi elementi videoigre sintetizirali na takav način potrebna je visoka razina povezanosti članova razvojnog tima, što je pomalo drugačije od tradicionalnog načina razvoja video igara gdje komunikacija između sektora nije toliko česta (obično nema potrebe za dnevnim dogovorima između programera, kompozitora i pisca). Zbog toga, *Gesamtkunstwerk* pristup izradi videoigara težak je i na produkcijskoj i organizacijskoj razini što objašnjava zašto nije toliko čest. Iako ima dosta primjera korištenja lajtmotiva za

[8] Međusobna sličnost određenih životinjskih jezika u igri odgovara bliskosti životinja koje ih govore na evolucijskom stablu, što je još jedan primjer sinteze mehanike igre i narativa.

mehaniku igre, za što je najjednostavniji primjer zvuk novčića u *Super Mariju* kada Mario pokupi novčić, ne nalazi se mnogo primjera koji su lajtmotive doveli do razine do koje ih je doveo Wagner u svojim operama ili Howard Shore u *Gospodaru prstenova*.

Najveći izazov predstavlja vođenje kreativnog tima, u kojemu se treba brinuti o sintezi elemenata videoigre pazeći da ju određeni elementi ne narušavaju. Primjerice, tokom izrade *Trip the Ark Fantastic* animatorici je odgovarao veći omjer likova spram ostatka scene kako mogla bolje prikazati njihove emocije, ali likovi su ipak prikazani maleni na velikoj pozadini kako bi se sačuvala dobra prostorna orijentacija u igri.

Unatoč tome, iskustvo članova tima u umjetničkom stvaranju van sfere videoigara je poželjno jer donosi nove metode rada koje dovode do inovativnijeg konačnog rezultata. Tako u timu koji izrađuje *Trip the Ark Fantastic* tako sudjeluju pisac koji je već izdao zbirku kratkih priča, te scenografkinja koja je radila na mnoštvu crtića i filmova.

Zaključak

U ovom radu iznijeli su se odabrani principi *Gesamtkunstwerka* u videoigrama: poklapanje narativa s audiovizualnim elementima, lajtmotivi i poklapanje narativa s mehanikom igre, na primjeru videoigre *Trip the Ark Fantastic.* Pritom se kao pretpostavka uzela da su videoigre oblik umjetnosti, a u borbi ludologije i naratologije zauzeo se potonji pristup prema kojemu se narativ i mehanika igre mogu i trebaju povezivati. Igra se shvatila kao element koji jednako sudjeluje u videoigrama kao *Gesamtkunstwerku* kao i elementi različitih oblika umjetnosti. Pri izradi, ali i kritici i konzumaciji videoigara, prednost bi se trebala dati sintezi svih odabranih gradivnih elemenata kako bi videoigre kao izrazito kompleksan oblik umjetnosti ostvarile svoj potencijal. Porastom *indie* i umjetničke scene u svijetu izrade videoigara za očekivati je da će s vremenom biti puno više primjera *Gesamtkunstwerk* igara.

Reference

[1] Birdsall, C. (2012) Cinema as a Gesamtkunstwerk?. Amsterdam University Press.
[2] Hegel, Georg Wilhelm Friedrich. (2011). Predavanja iz estetike. I. Zagreb: Demetra.
[3] Hrvatska enciklopedija. Dostupno na: http://www.enciklopedija.hr/ (10.9.2019.)
[4] Hrvatski jezični portal. Dostupno na: http://hjp.znanje.hr/index.php?show=search (10.9.2019.)
[5] Tavinor, Grant. (2009). The Art of Videogames. MA: Wiley-Blackwell.

[6] Vigato, M. (2019). Jesu li videoigre umjetnost?. Dostupno na: https://apps.unizg.hr/rektorova-nagrada/javno/akademske-godine/2018/nagradeni-radovi (10.9.2019)
[7] Wolfman, U. R. (2013). Richard Wagner's Concept of the 'Gesamtkunstwerk'. Interlude. Dostupno na: https://interlude.hk/richard-wagners-concept-of-the-gesamtkunstwerk/ (10.9.2019.)

Razvoj sustava za interakciju unutar višekorisničke umrežene igre korištenjem sustava Unreal

Dobrila Šunde i Mirko Sužnjević
Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu
{mirko.suznjevic@fer.hr, dobrila.sunde@gmail.com}

Sažetak

U ovom radu se opisuje kreiranje sustava za univerzalnu interakciju sa svim postojećim tipovima objekata u sustavu Unreal na nekoj razini u višekorisničkim igrama. Objekti koji nasljeđuju Aktora unutar sustava Unreal mogu biti kandidati za slanje i replikaciju preko mreže, no svi ostali zahtijevaju prilagođeno rješenje [1]. Cilj je s dobrim početnim dizajnom sustava eliminirati posebno baratanje za svaku novo dodanu interakciju. Uspješno je implementirano da korisnik ne mora razmišljati o tipu objekta koji se nalazi u sceni i kako se on šalje preko mreže, nego može jednostavno na svaki nadodati željene interakcije i implementirati njihovu funkcionalnost.

Ključne riječi: Unreal Engine, interakcije, višekorisnička umrežena igra

Uvod

C++ je programski jezik koji se koristio za izgradnju većine velikih konzolnih i Windows igara. Brz je, kompajleri i optimizatori su solidni i programeri imaju veliku kontrolu nad upravljanjem memorijom. Ima opsežne knjižnice koje se mogu koristiti za dizajniranje složenih grafika. Video igre kreirane s C++ programskim jezikom se također lako distribuiraju nad raznim platformama. Najpopularnije razvojne okoline za razvoj video igara su Unity i Unreal Engine. Dok Unity koristi C# programski jezik, Unreal Engine koristi samo C++ programski jezik. Kako bi ovaj rad bio što bliži najvišim grafičkim standardima industrije video igara odlučeno je kreirati ovaj projekt u Unreal Engine razvojnoj okolini. Unreal Engine također ima dobru arhitekturu i podršku za višekorisničke igre koje su bile fokus ovog rada.

Interakcije čine jednu od najosnovnijih funkcionalnosti i mehanika u igrama. Sustav Unreal podržava različite tipove objekata u svojim scenama. Pravila replikacije i slanja preko mreže za svaki od tih tipova mogu biti vrlo različiti. Objekti su instance klase koje nasljeđuju Unrealovu UObject klasu. UObject klasa je bazna klasa za sve objekte u Unreal sustavu, uključujući Aktore. Znači sve instance u Unreal sustavu su zapravo Objekti, no pojam Aktor se koristi za instance klase koje nasljeđuju od klase AActor u hijerarhiji. Da bi se objekt

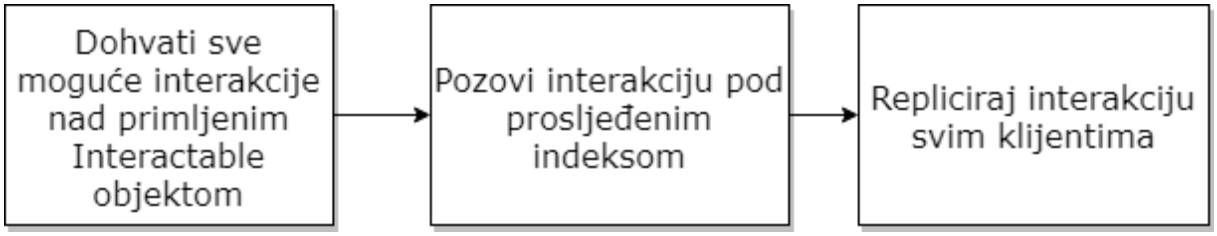
mogao postaviti na razinu mora nasljeđivati barem od Aktora ili neke niže klase [2]. Bilo koji objekt koji nasljeđuje Aktora unutar sustava Unreal može biti kandidat za slanje i replikaciju preko mreže. Problem nastaje kada se želi poslati neki objekt koji ne nasljeđuje Aktora [1]. Motivacija ovog rada je dobrim početnim dizajnom sustava spriječiti da korisnik svakom novo dodanom interakcijom treba posebno razmišljati o njenom slanju preko mreže i omogućiti mu jednostavno implementiranje funkcionalnosti nad objektima u sceni. Npr. za Aktora vrata koji se sami repliciraju preko mreže podržavat će se interakcije otvaranja i zatvaranja preko mreže. Za vegetaciju npr. drvo koje ne podržava replikaciju jer nije Aktor, podržavat će se interakcija rezanja drva preko mreže. Dodatno za prikaz da radi na bilo kakvoj vegetaciji koja također nisu Aktori implementirat će se jednostavni Blueprint za ispisivanje teksta na ekranu prilikom interakcije. Bilo kakva naknadno dodana interakcija neće trebati paziti na tip objekta na kojeg je dodana i njegov način slanja preko mreže.

Vrste interakcija u sustavu Unreal

Projekt bi trebao podržavati sve moguće tipove interakcija u virtualnom svijetu. U skladu s njima će se postavljati i scena kako bi se testiralo da li sustav radi u svim mogućim slučajevima. Razlikujemo tri osnovna tipa elemenata u virtualnoj sceni te će se za svaki od njih morati drukčije baratati interakcijama.

Interakcija sa statičnom mrežom poligona

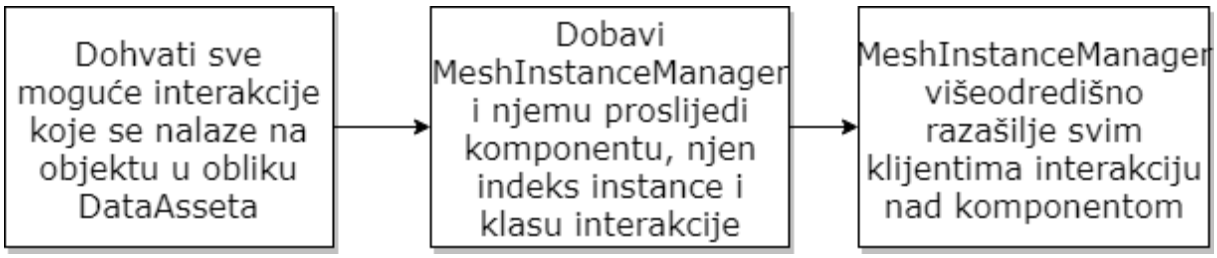
Statične mreže poligona su jedan od temeljnih tipova renderabilne geometrije u Unreal sustavu. Koriste se za kreiranje arhitekture i prikaz objekata u igrama kao npr. metaka, oružja i sl. [2]. Prilikom pokretanja bacanja zrake rezultat udara vraća komponentu Aktora kojeg je udario. Ta komponenta može biti ili tipa klase UMeshComponent ili UInstancedStaticMeshComponent. Instancirane statične mreže poligona su efikasan način prikazivanja mnogo kopija iste mreže poligona [3]. Na slici 1 se može vidjeti što se događa na poslužiteljskoj strani nakon primanja Interactable sučelja nekakve statične mreže poligona u sceni.



Slika br. 1: Dijagram toka interakcije sa statičnom mrežom poligona na poslužitelju

Interakcija s instanciranom mrežom poligona vegetacije

Sustav instancirane mreže poligona vegetacije omogućava brzo crtanje ili brisanje niza statičnih mreža poligona na Aktorima okoline, drugim Aktorima statičnih mreža poligona i BSP (Binary Space Partitioning) geometriji. Koriste se kao hijerarhijske instancirane statične mreže poligona unutar sustava Unreal. Funkcioniraju vrlo slično instanciranim statičkim mrežama poligona, osim što podržavaju LOD-ove (Level of Details) i uklanjanje udaljenih poligona (što dolazi s dodatnom cijenom u performansama). Te mreže poligona se automatski grupiraju zajedno u skupine koje se prikazuju korištenjem hardverskog instanciranja što znači da se mnoge instance mogu prikazati samo s jednim prolazom iscrtavanja [4]. Prilikom pokretanja bacanja zrake rezultat udara će vratiti DataAsset, koji može biti potklasa za pohranjivanje prilagođenih podataka na objektima u Unrealu, komponentu instancirane mreže poligona i njen indeks instance. Na slici 2 se može vidjeti dijagram toka interakcije s instanciranom mrežom poligona na poslužiteljskoj strani.



Slika br. 2: Dijagram toka interakcije s instanciranom mrežom poligona na poslužitelju

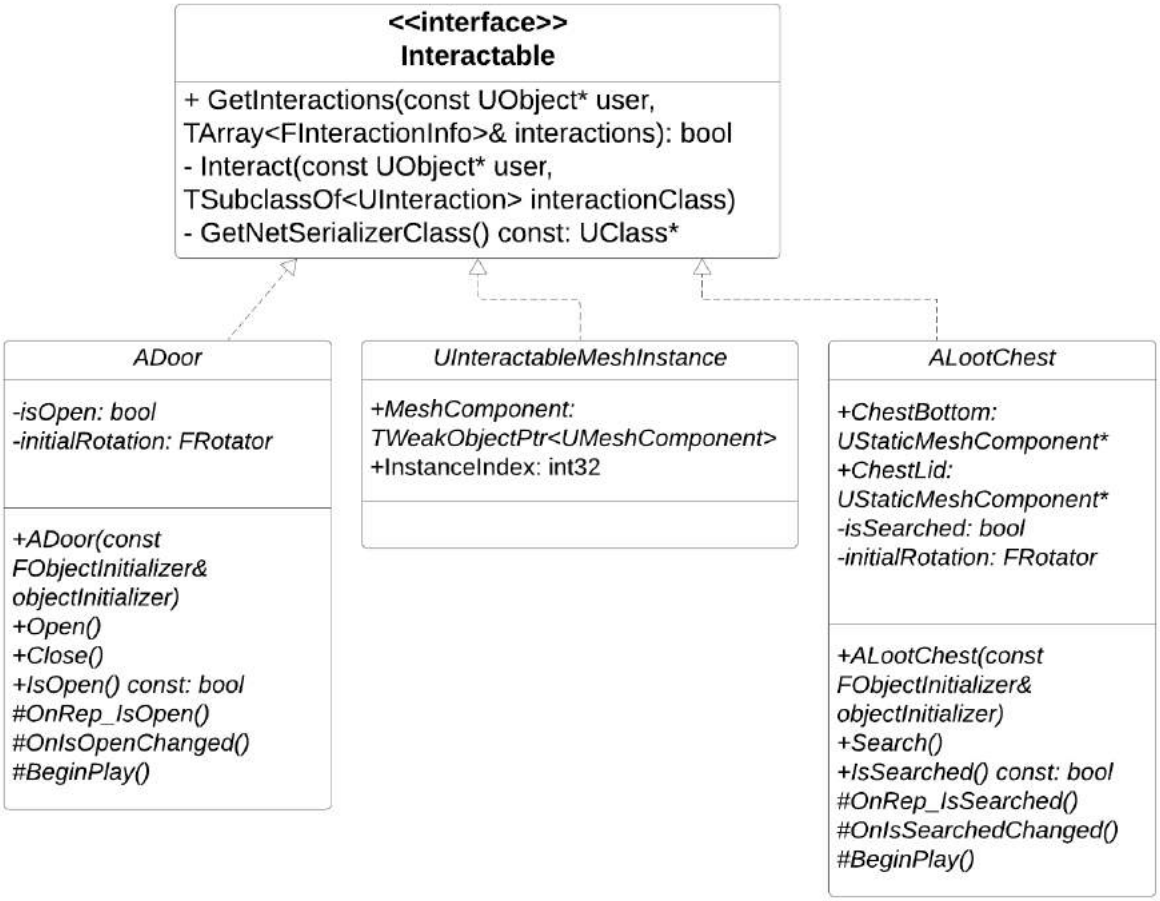
Interakcija s vegetacijom terena

Sve što ne spada u gornje dvije kategorije interakcije zapravo spada u kategoriju interakcije s vegetacijom terena. Teren se sastoji od pločica. Objekt svake pločice terena se zove posrednikom terena u Unreal sustavu. Popločani tereni su zapravo ploče koje pripadaju jednom terenu. Prednost popločanih terena je to što učitavaju ili prazne ploče kako se igrač približava ili udaljava kako bi uštedjeli na performansama [4]. Vrijedi isti dijagram toka na slici 2 na poslužitelju kao i za instanciranu mrežu poligona vegetacije.

Model rješenja

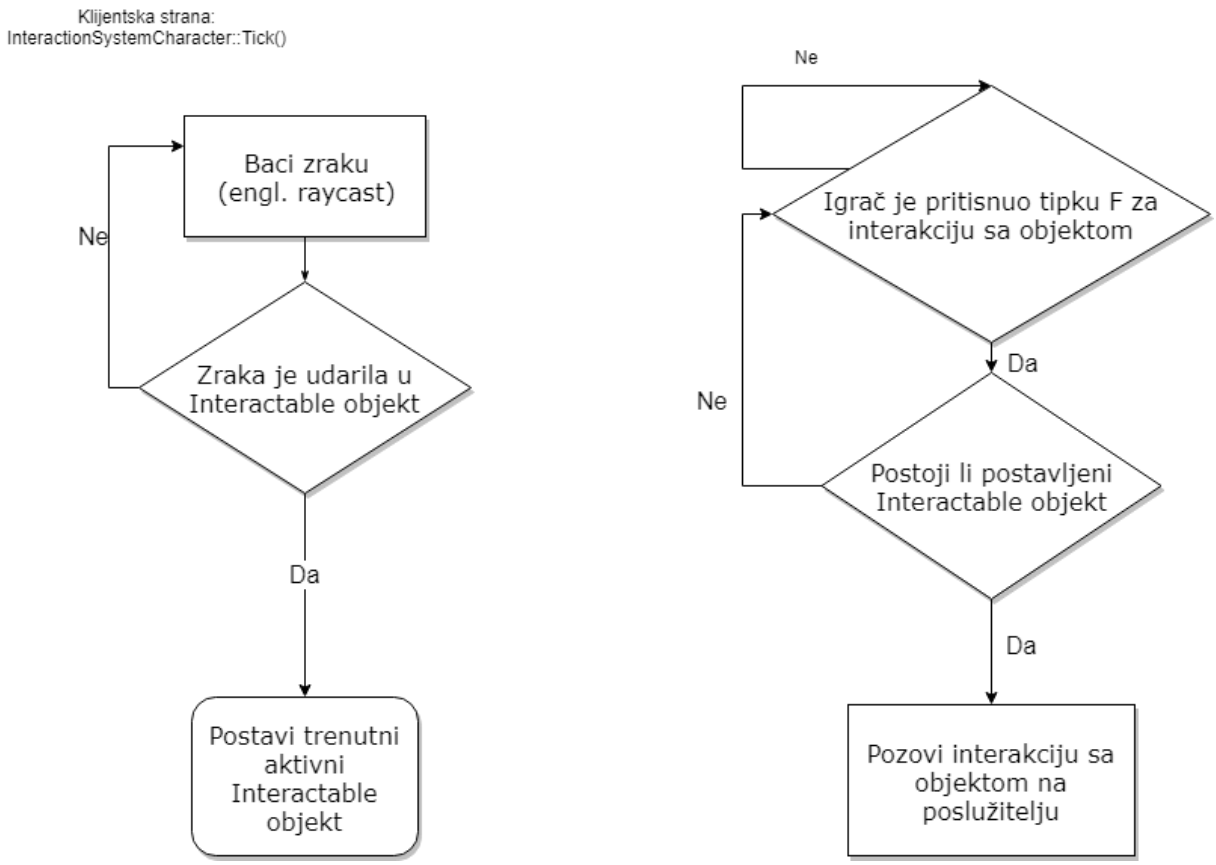
Osnovna ideja dizajna sustava interakcija je da postoji nekakvo univerzalno sučelje, u ovom radu nazvano Interactable sučelje, koje će svi objekti s kojima se može ulaziti u interakciju tada nasljeđivati i implementirati. Problem predstavlja višekorisnički dio. Replikacija se obavlja nad Aktorima, a sučelje Interactable nije Aktor, već nasljeđuje UInterface sustava Unreal Engine. Dakle potrebno je naći način prosljeđivanja Interactable sučelja između klijenta i

poslužitelja. Na slici 3 je prikazan dijagram sa primjerima klasa koje ga nasljeđuju.



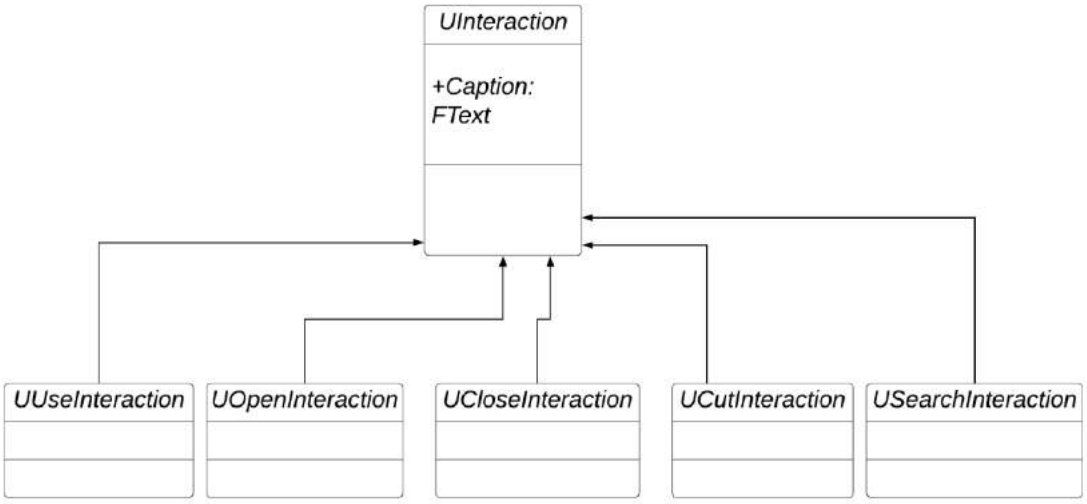
Slika br. 3: Dijagram sučelja Interactable i klasa koje ga nasljeđuju

U svakom osvježavanju AInteractionSystemCharacter klase igrača kojom će igrač moći upravljati vanjskim kontrolama poziva se bacanje zraka. Ako zraka pogodi u neki Interactable objekt postavi varijablu trenutno aktivnog Interactable objekta igračeve klase na taj pogođeni Interactable objekt. Dijagram toka ove funkcionalnosti se može vidjeti na slici 4.



Slika br. 4: Dijagram toka InteractionSystemCharacter klase na klijentu

Ako igrač pritisne tipku F na tipkovnici a postavljen je aktivni Interactable objekt onda će se pozvati interakcija s trenutno aktivnim objektom na poslužitelju. Na slikama 1 i 2 se može vidjeti što se događa na poslužiteljskoj strani. Cijeli proces dodavanja novih interakcija dodatno olakšava klasa Interaction koja predstavlja baznu klasu za sve moguće interakcije. Primjer nekih interakcija koje je nasljeđuju se može vidjeti na slici 5.



Slika br. 5: Dijagram klasa koje nasljeđuju Interaction klasu

Primjer tipičnog procesa dodavanja novog Interactable objekta u svijet:

- Objekt mora naslijediti Interactable sučelje
- Objekt može dodati svoje nove moguće interakcije u programski kod koje nasljeđuju Interaction klasu i odrediti im natpis koji će se prikazivati na zaslonu kada pređu mišem preko tog objekta
- Objekt mora nadglasati funkcije Interactable sučelja za svoje željene interakcije i implementirati željenu funkcionalnost za svaku moguću interakciju
- Kada igrač pritisne tipku F iznad trenutno aktivnog Interactable objekta Interactable objekt se mora poslati preko mreže
- Svaki objekt može sam implementirati svoju NetSerialize klasu za njegovo slanje preko mreže ili koristiti neku od već prethodno implementiranih
- Klasa FInteractableReplicationHelper koja će poslije biti bolje objašnjena služi za prilagođeno slanje objekata preko mreže i njegova NetSerialize klasa poziva željenu NetSerialize klasu proslijeđenog Interactable objekta
- Poslužitelj uspješno primi Interactable objekt, raspakira ga i pozove nad njim njegovu Interact funkciju
- Svaki objekt je sam odgovoran da se unutar njegove Interact funkcije to ponašanje ili replicira ili višesmjerno razašilje klijentima

Serijalizacija prilagođenih struktura za umrežavanje u Unreal sustavu

UStruct je verzija strukture sustava Unreal Engine. Da bi se koristila nije potrebno naslijediti nikakvu klasu već samo treba označiti strukturu za USTRUCT().
Prilikom deklariranja UStruct u Unreal sustavu moguće je dodati NetSerialize metodu. Ako se ta metoda definira, Unreal Engine će je koristiti prilikom serijalizacije i deserijalizacije te strukture za umrežavanje tijekom replikacije svojstava i RPC-a.

Kreira se pomoćna struktura FinteractableReplicationHelper za slanje Interactable sučelja preko mreže. U programskom kodu Isječak koda 1 se može vidjeti kako dodati metodu NetSerialize UStructu koji će sadržavati Interactable sučelje koje je potrebno slati preko mreže i serijalizirati ili deserijalizirati.

```
//-------------------------------------------------------------------------------------
USTRUCT()
struct FInteractableReplicationHelper
{
	GENERATED_USTRUCT_BODY()

public:
	TScriptInterface<IInteractable> Interactable;

public:
	FInteractableReplicationHelper() = default;
	FInteractableReplicationHelper(const TScriptInterface<IInteractable>&
interactable) : Interactable(interactable)
	{

	}

	bool NetSerialize(FArchive& ar, class UPackageMap* Map, bool& bOutSuccess);
};
//-------------------------------------------------------------------------------------
template<>
struct TStructOpsTypeTraits<FInteractableReplicationHelper> : public
TStructOpsTypeTraitsBase2<FInteractableReplicationHelper>
{
	enum
	{
		WithNetSerializer = true,
	};
};
//-------------------------------------------------------------------------------------
```

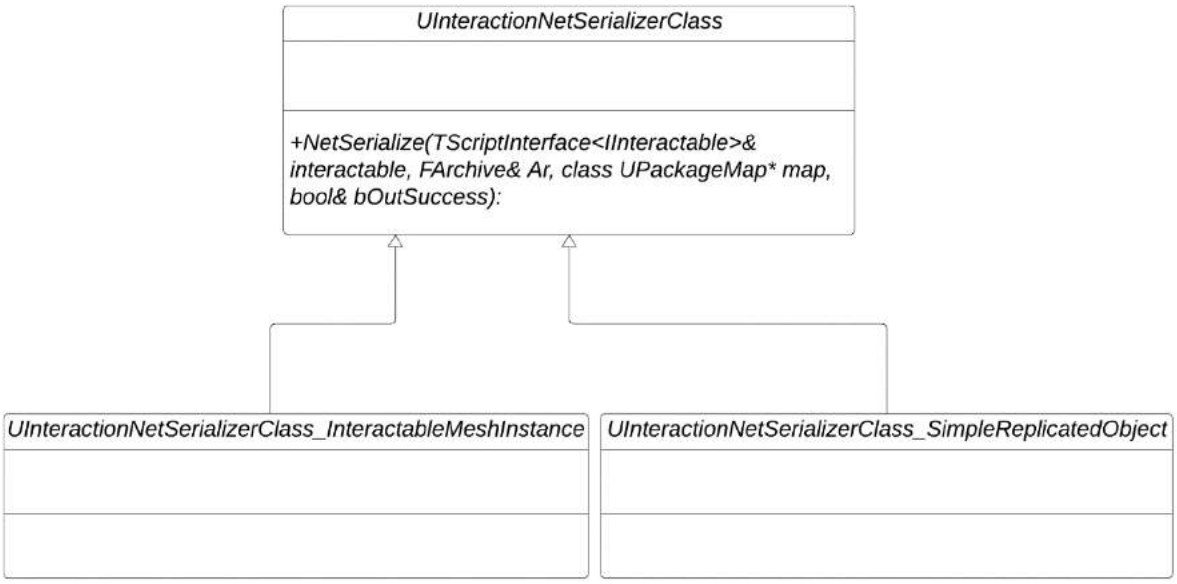
Isječak koda 1: Dodavanje metode NetSerialize u Ustruct strukturu

Potrebno je primijetiti zadnji dio u programskom kodu: kako bi se Unreal sustavu reklo da UStruct definira prilagođenu NetSerializer funkciju, potrebno je postaviti na istinu svojstvo tipa WithNetSerializer za strukturu FInteractableReplicationHelper. Ako se taj dio koda ne doda, NetSerialize metoda se nikad neće pozvati.

Metoda NetSerialize prima FArchive u koji pakira i iz kojeg otpakirava podatke strukture. FArchive je klasa koja implementira česti obrazac za serijalizaciju podataka, omogućavajući pisanje dvosmjernih funkcija. U osnovi, kada je riječ o serijalizaciji, potrebno je osigurati da je način serijalizacije podataka jednak onom koji se koristi za deserijalizaciju. Najbolji način da se to osigura je pisanje jedne funkcije osjetljive na kontekst koja obavlja obje stvari. FArchive preopterećuje << operator koji je osnova stvaranja dvosmjernih funkcija. Njegovo ponašanje je osjetljivo na kontekst: kada je FArchive u načinu pisanja, kopira podatke s desna na lijevo, kada je FArchive u načinu čitanja, kopira podatke s lijeva na desno.

Pristup serijalizaciji i deserijalizaciji će u slučaju rada s Interactable objektima biti drukčiji, ovisno o kojem se objektu radi. Potrebno je kreirati baznu klasu, u ovom radu nazvanu UinteractionNetSerializerClass, za serijalizaciju i deserijalizaciju Interactable objekata koju će onda naknadno nasljeđivati

specijalizirane klase i implementirati njenu NetSerialize metodu u skladu s potrebama. Dijagram klasa koje nasljeđuju UinteractionNetSerializerClass se može vidjeti na slici 6.



Slika br. 6: Dijagram klasa koje nasljeđuju InteractionNetSerializer klasu

Tada implementacija NetSerialize metode pomoćne strukture može izgledati kao što je prikazano u Isječak koda 2.

```
//------------------------------------------------------------------------------------
bool FInteractableReplicationHelper::NetSerialize(FArchive& Ar, class UPackageMap*
Map, bool& bOutSuccess)

{
	if (Ar.IsSaving())
	{
		UClass* netSerializerClass = Interactable->GetNetSerializerClass();
		Ar << netSerializerClass;

		UInteractionNetSerializerClass* serializer =
NewObject<UInteractionNetSerializerClass>(Map, netSerializerClass);
		serializer->NetSerialize(Interactable, Ar, Map, bOutSuccess);
	}
	else if (Ar.IsLoading())
	{
		UClass* netSerializerClass = nullptr;
		Ar << netSerializerClass;

		UInteractionNetSerializerClass* serializer =
NewObject<UInteractionNetSerializerClass>(Map, netSerializerClass);
		serializer->NetSerialize(Interactable, Ar, Map, bOutSuccess);
	}

	return true;
}
//------------------------------------------------------------------------------------
```

Isječak koda 2: Implementacija NetSerialize metode

Osnovna ideja je da struktura FInteractableReplicationHelper ovisno o Interactable objektu kojeg sadržava pozove točnu NetSerialize metodu pa bazna klasa koju će te metode nasljeđivati olakšava taj proces.

Ako se sprema u arhivu tj. šalju objekti preko mreže, struktura FInteractableReplicationHelper će pročitati klasu koju sadržani objekt želi koristiti za serijalizaciju ili deserijalizaciju i spremiti je u arhivu za slanje. Kreirat će tu klasu za serijalizaciju ili deserijalizaciju i pozvati njenu NetSerialize metodu koja će rukovati s ostalim potrebnim akcijama za spremanje tog objekta. Ako se čita iz arhive tj. primaju objekti preko mreže, struktura FInteractableReplicationHelper će pročitati klasu za serijalizaciju ili deserijalizaciju iz arhive i kreirati novi objekt te klase koji će potom pozvati NetSerialize metodu koja će rukovati s ostalim potrebnim akcijama za čitanje tog objekta.

Rezultati

Vrata su u ovom projektu Aktor koji se replicira preko mreže. Koriste jednostavnu serijalizaciju gdje šalju sam Interactable objekt preko mreže prilikom interakcije te poslužitelj nad njime obavlja interakciju i to novo stanje replicira svim klijentima. Igrač dolaskom do vrata vidi mogućnost interakcije otvaranja ili zatvaranja vrata kao na slikama 7 i 8 te pritiskom na tipku F ih može otvoriti ili zatvoriti. Još jedan primjer takve jednostavne serijalizacije Aktora se može vidjeti na slici 9 gdje je prikazana interakcija pretraživanja škrinje.

Slika br. 7: Interakcija otvaranja vrata

Slika br. 8: Interakcija zatvaranja vrata

Slika br. 9: Interakcija zatvaranja vrata

Klasa InteractableMeshInstance služi kao omotač oko instanciranih mreža poligona koje nemaju Interactable sučelje na sebi. Svaka instancirana mreža poligona se sastoji od komponente mreže poligona i indeksa instance. Npr. sva vegetacija na razini je jedna instancirana statična mreža poligona a pojedino drvo je njena komponenta koja ima svoj indeks. Klasa MeshInstanceManager sprema InteractableMeshInstance u mapu.

Svaki put kada igrač pređe mišem preko bilo kakve instancirane mreže poligona i kreira se omotač InteractableMeshInstance spremit će se u tu mapu. Ključ te mape je kombinacija komponente mreže poligona i njenog indeksa. Zato je potrebno kreirati taj par koji će moći naknadno služiti kao ključ mape. Funkcija GetInteractions() dohvaća prethodno spomenuti DataAsset s komponente i iz njega čita interakcije. Moguće je da se DataAsset nalazi na samoj statičnoj mreži poligona umjesto samo na komponenti pa je potrebno i to provjeriti i probati je dohvatiti. Postoje dva moguća tipa mreže poligona: statična mreža poligona i skeletna mreža poligona, koja se često koristi za prikaz likova i drugih animiranih objekata, pa se rade provjere za njih kao što je prikazano u Isječak koda 3.

```
//-------------------------------------------------------------------------------------
bool UInteractableMeshInstance::GetInteractions(const UObject* user,
TArray<FInteractionInfo>& interactions)
{
	if (MeshComponent == nullptr)
	{
		return false;
	}

	UInteractionAssetUserData* interactionUserData = MeshComponent-
>GetAssetUserData<UInteractionAssetUserData>();
	if (interactionUserData == nullptr)
	{
		if (UStaticMeshComponent* staticMeshComponent =
Cast<UStaticMeshComponent>(MeshComponent.Get()))
		{
			if (UStaticMesh* staticMesh = staticMeshComponent-
>GetStaticMesh())
			{
				interactionUserData = staticMesh-
>GetAssetUserData<UInteractionAssetUserData>();
			}
		}
		else if (USkinnedMeshComponent* skinnedMeshComponent =
Cast<USkinnedMeshComponent>(MeshComponent.Get()))
		{
			if (USkeletalMesh* skeletalMesh = skinnedMeshComponent-
>SkeletalMesh)
			{
				interactionUserData = skeletalMesh-
>GetAssetUserData<UInteractionAssetUserData>();
			}
		}
	}

	if (interactionUserData == nullptr)
	{
		return false;
	}

	interactions.Append(interactionUserData->Interactions);
	return true;
}
//-------------------------------------------------------------------------------------
```

Isječak koda 3: Provjere za statičnu mrežu poligona i skeletnu mrežu poligona

Konačno implementacije klase za serijalizaciju tog objekta preko mreže se može vidjeti u Isječak koda 4.

```
//---------------------------------------------------------------------------
bool
UInteractionNetSerializerClass_InteractableMeshInstance::NetSerialize(TScriptInterface
<IInteractable>& interactable, FArchive& Ar, class UPackageMap* map, bool&
bOutSuccess)
{
	if (Ar.IsSaving())
	{
		UInteractableMeshInstance* obj =
Cast<UInteractableMeshInstance>(interactable.GetObject());
		Ar << obj->MeshComponent;
		Ar << obj->InstanceIndex;
	}
	else if (Ar.IsLoading())
	{
		UMeshComponent* meshComponent;
		int32 instanceIndex;
		Ar << meshComponent;
		Ar << instanceIndex;

		if (AMeshInstanceManager* meshInstanceManager =
GetMeshInstanceManager(map))
		{
			interactable = meshInstanceManager-
>FindOrCreateInteractableMeshInstance(meshComponent, instanceIndex);
		}
		else
		{
			interactable = nullptr;
		}
	}

	return true;
}
//---------------------------------------------------------------------------
```

Isječak koda 4: Implementacija klase za serijalizaciju objekta mreže poligona

Korištenje takve serijalizacije omogućilo je interakciju rezanja pojedinog drva na razini, prikazano na slici 10, čiju promjenu stanja, tj nestajanje s razine mogu vidjeti svi klijenti na poslužitelju. Dodatno je implementirana interakcija ispisivanja teksta svim klijentima prikazana na slici 1.

Slika br. 10: Interakcija sječe drva

Slika br. 11: Interakcija ispisivanja teksta svim spojenim klijentima

Zaključak

Unreal Engine je razvojno okruženje za razvoj video igara koje se još uvijek svakodnevno razvija. Njihova mrežna arhitektura omogućava jednostavan razvoj višekorisničkih video igara ali također podržava i implementaciju novih funkcionalnosti ako su korisniku potrebne kao što je bio slučaj u ovom radu. Univerzalni sustav interakcija je kreiran s idejom olakšavanja naknadnog dodavanja funkcionalnosti u igru. Njegova arhitektura je tijekom rada

uspješno osmišljena i implementirana te je postignuta indiferencija nad tipom objekta. Korisnik ne mora razmišljati o tipu objekta koji se nalazi u sceni nego može jednostavno na svaki nadodati željene interakcije i njihovu funkcionalnost implementirati kroz programski kod ili vizualno skriptiranje u Blueprintsu.

Reference

[1] N. C, "'Unreal Engine 4' Network Compendium," [Online]. Available: http://cedric-neukirchen.net/Downloads/Compendium/UE4_Network_Compendium_by_Cedric_eXi_Neukirchen.pdf. Posjećeno 5.9.2019.

[2] "Unreal Engine 4 Documentation Gameplay Framework Quick Reference," [Online]. Available: https://docs.unrealengine.com/en-US/Gameplay/Framework/QuickReference/index.html. Posjećeno 5.9.2019.

[3] "Unreal Engine 4 Documentation Instanced Static Mesh," [Online]. Available: https://docs.unrealengine.com/en-US/BlueprintAPI/Components/InstancedStaticMesh/index.html. Posjećeno 5.9.2019.

[4] "Unreal Engine 4 Documentation Foliage Tool," [Online]. Available: https://docs.unrealengine.com/en-US/Engine/Foliage/index.html. Posjećeno 5.9.2019.

[5] "Unreal Engine 4 Documentation Controller," [Online]. Available: https://docs.unrealengine.com/en-US/Gameplay/Framework/Controller/index.html. Posjećeno 5.9.2019.

Vrednovanje korisničkog iskustva 2D platformske igre

Domagoj Bakić[1] i Dijana Plantak Vukovac[1]
[1]Sveučilište u Zagrebu, Fakultet organizacije i informatike Varaždin
dombakic@foi.hr, dijana.plantak@foi.hr

Sažetak

Korisničko iskustvo (engl. *User Experience*, UX) predstavlja emocije, stavove i ponašanja koji se javljaju kada osoba koristi određeni proizvod, sustav ili uslugu. Korisničko iskustvo u video igrama dugo je vremena bilo zanemareno zbog ograničenog vremena i budžeta u razvoju igre, no pozitivno korisničko iskustvo predstavlja izuzetno važan aspekt razvoja igre koji će omogućiti privlačenje i zadržavanje igrača. U ovom radu prikazani su rezultati vrednovanja korisničkog iskustva igre *Inside* primjenom dvaju upitnika: prvo je vrednovana upotrebljivost igre pomoću upitnika *System Usability Scale* (SUS), a zatim je vrednovano cjelokupno korisničko iskustvo pomoću upitnika *Game Experience Questionnaire* (GEQ). Prema rezultatima, igra Inside ima prosječno ocijenjenu upotrebljivost, a pozitivan utjecaj na korisničko iskustvo kod ispitanika najviše ima kategorija uranjanja ili imerzije igrača.

Ključne riječi: korisničko iskustvo, video igre, vrednovanje, SUS, GEQ

Uvod

Pojam korisničkog iskustva tema je rasprave među HCI (engl. *Human - Computer Interaction*) zajednicom. Može se na njega gledati kao na krovni pojam za poticanje istraživanja HCI zajednice da se usredotoči na aspekte koji nadilaze dizajn korisničkog sučelja i upotrebljivost (engl. *Usability*). Upotrebljivost se odnosi na točno i efikasno izvršavanje zadatka od strane korisnika dok je u interakciji s proizvodima, sustavima ili uslugama, i dio je korisničkog iskustva. S druge strane, korisničko iskustvo povezano je s hedonističkim aspektima interakcije i/ili posjedovanja proizvoda, kao što su estetika, zabava, izazov, stimulacija ili samoizražavanje [4]. McCarthy i Wright [7] navode da je korisničko iskustvo veza između ljudi i interaktivnih tehnologija. Ovom definicijom se širi fokus s računala na široki raspon interaktivnih tehnologija i od radnih zadataka do doživljenog iskustva. Formalizacija izraza *korisničko iskustvo* dana je ISO standardom 9241-110: 2010 i fokusirana je na percepciju osobe i njene odgovore koji proizlaze iz upotrebe ili anticipirane upotrebe proizvoda, sustava ili usluge [6].

Dobro ili loše dizajnirano korisničko iskustvo je obično jednostavno za prepoznati, ali teško za definirati zato što je korisničko iskustvo subjektivno iskustvo pojedinaca. Kao što ime sugerira, korisničko iskustvo je iskustvo koje korisnik doživljava prilikom interakcije s proizvodom, sustavom ili uslugom. Slično kao i kod glazbe, korisničko iskustvo nekog softvera može biti subjektivno dobro ili subjektivno loše dizajnirano. Ne postoji objektivno dobro ili loše korisničko iskustvo nego sve ovisi o samom korisniku [1].

Vrednovanje korisničkog iskustva u video igrama je važno jer daje uvid kako su igrači prihvatili igru. Tako razvojni inženjeri mogu shvatiti gdje su pogriješili i gdje su donijeli dobre odluke u dizajnu što im može pomoći u budućim projektima. U ovom radu opisano je vrednovanje upotrebljivosti pomoću upitnika *System Usability Scale* (SUS) [9] i vrednovanje korisničkog iskustva pomoću upitnika *Game Experience Questionnaire* (GEQ) [5] na primjeru 2D platformske igre *Inside*.

Korisničko iskustvo u kontekstu video igara

Anderson i suradnici [1] navode da je dobro imati za cilj postizanje dobrog korisničkog iskustva u softveru, ali da kvalitetno dizajnirano korisničko iskustvo i nije pravi cilj. Ono vodi nečemu drugom i važnijem, daje smisao softveru. U kontekstu video igre, korisničko iskustvo daje svrhu video igri.

Iskustvo igranja video igara obično se shvaća kao subjektivni odnos korisnika i video igre izvan stvarne implementacije igre. Provedba je vezana brzinom mikroprocesora igraće platforme, ergonomijom kontrolera, upotrebljivim sučeljem te brzinom internet veze ako govorimo o online igrama. Korisničko iskustvo je više od toga, ono se smatra i osobnim odnosom. Shvaćanje ovog odnosa kao osobnog problematično je u znanstvenim okvirima. Osobno i subjektivno znanje ne dopušta generaliziranje ili krivotvorenje teorije (Popper, citirano u [3]).

Korisnici prvo identificiraju igru, a zatim izgrađuju odnos s njom. Vlasništvo igre je veza koje vodi do uživanja. Vlasništvo se postiže kada igrač ima kontrolu nad igrom. Ako je kontrola slaba tada drugi potporni elementi moraju stvoriti osjećaj vlasništva kod igrača. Igrač tada u igri izgrađuje svoju priču. Način na koji igrač čini igru svojom je korištenje radnji koje se mogu upotrijebiti za pobjedu u igri ili za postizanje vlastitih ciljeva. Kako igra napreduje, igrač počinje primati različite vrste nagrada što može biti od pomoći za pobjedu u igri ili samo nešto što igrač uživa raditi. To je ujedno i prilika da igrač može učiniti nešto strano svojoj stvarnosti [3].

Korisničko iskustvo u video igrama opisuje se pojmovima kao što su imerzija, zabava, prisutnost, uključenost, angažiranost i tok. Često su ti pojmovi

definirani prilično široko, primjerice tok je opisan kao optimalno iskustvo. Ovim pojmovima pridaju se različiti psihološki elementi: npr. koncentracija, emocije i kognitivne procjene izazova u igri nazivaju se uranjanjem ili imerzijom [10].

Ciljevi ili zadaci unutar igre usmjeravaju igrače. Unutar pravila igre i izbora, igrači slijede ciljeve, ostvaruju nagrade, donose odluke i suočavaju se s izazovnim situacijama. Igrači svjesno ili nesvjesno za vrijeme igranja ocjenjuju svoju izvedbu u igri. Pitaju se postižu li željene ciljeve i mogu li odgovoriti na izazove koje im igra postavlja. Kad postignu ciljeve nakon svladavanja prepreka, pojavljuju se pozitivni osjećaji i osjećaj kompetencije. Igra pripovijedanja pretvara se u priču u kojoj igrač ima aktivnu ulogu. Zanimljiva mjesta usmjeravaju igračev fokus na svijet igre i pružaju bijeg iz stvarnog svijeta. Igrači postaju angažirani kroz svoju ulogu s događajima koji se događaju u igri [10].

Vrednovanje korisničkog iskustva u video igrama

Novak (citirano u [2]) govori da se vrednovanje korisničkog iskustva u video igrama i drugim sličnim sustavima za zabavu odvija još u ranim fazama razvoja igre. Programeri prvih računalnih sustava počeli su razvijati prve verzije digitalnih igara i uspostavili su veoma primitivnu formu vrednovanja korisničkog iskustva tako da su jednostavno probali igrati igru i shvatili zašto neke stvari treba promijeniti. Pojava igara kao što je Tetris pokazale su kako male promjene u načinu igranja ili priči mogu imati veliki utjecaj na ukupno korisničko iskustvo tijekom igranja igre.

Kod implementacijskih faza i faza testiranja popularne metode vrednovanja korisničkog iskustva su (Novak, citirano u [2]):

- Testiranje igranjem – igranje različitih verzija igre ovisno o stupnju razvoja. Traže se nekonzistentnosti i pogreške.
- Polustrukturirani intervjui – otvorenija vrsta intervjua s krajnjim korisnicima (igračima) koja nema nužno predefinirana pitanja.
- Promatranje – promatranje igrača koji igraju igru. Prate se verbalna i neverbalna ponašanja.
- Kvantitativne usporedbe ponašanja igrača – ponašanja igrača koja se mogu brojčano prikazati, npr. vrijeme potrebno za prelazak jednog nivoa u igri.
- Upitnici fokusirani na stavove i iskustva igrača – upitnici koji ispituju igrače o njihovom iskustvu igranja.
- Heurističko vrednovanje – igru vrednuju HCI stručnjaci pomoću heuristika za igrivost.

Sve navedene metode vrednovanja poznate su i često korištene u razvoju igara. Ipak, neki razvojni studiji ih pritisnuti vremenskim rokovima i smanjenim budžetom zanemaruju i nerijetko se dogodi da igra ne bude dovoljno testirana pa se igračima ne svidi. U današnje vrijeme, kada se razvojem igara bavi velik broj razvojnih studija, igračima nije teško naći drugu igru za igrati i treba biti oprezan prilikom vrednovanja kako bi se na tržište izbacila igra u najboljem mogućem stanju, odnosno nadogradnjom igre ostvarilo bolje korisničko iskustvo.

Primjer je studio *Epic Games*, koji je u svoju megapopularnu igru *Fortnite* odlučio dodati mogućnost korištenja padobrana i nakon inicijalnog skoka s mape pri padu s određenih visina. U početku je ova mehanika bila dio eksperimentalnog *moda* u igri, ali je kasnije dodana i u standardne *modove*. Mnogi igrači nisu s oduševljenjem prihvatili novu mehaniku jer se promijenila brzina i koncepcija igre, koja je sada iziskivala drugačije taktike i strategije za ostvarivanje pobjede. Nakon mnogo rasprava o ovoj odluci, *Epic Games* je objavio da eksperiment nije prošao prema očekivanjima te da će idućom zakrpom u standardnim *modovima* onemogućiti višestruko korištenje padobrana [8].

Vrednovanje korisničkog iskustva 2D platformske igre *Inside* pomoću upitnika

Plan istraživanja

U nastavku rada prikazuje se dio istraživanja provedenog u okviru izrade diplomskog rada. Vrednovanje korisničkog iskustva provedeno je na video igri *Inside* koja je *indie* projekt razvojnog studija *PlayDead*. Riječ je o 2D platformskoj igri s mračnom tematikom preživljavanja i jednostavnim mehanikama kretanja i rješavanja zagonetki. Vrednovanje je provedeno upravo na ovoj igri jer je dovoljno jednostavna za igranje da ju mogu odigrati svi ispitanici, a dovoljno drugačija od ostalih igara na tržištu da nudi nešto drugačije što može dati bolji uvid u emocije ispitanika. Prelazak igre ne traje duže od tri sata, iako ni potpuni prelazak igre nije potreban kako bi se stekao dojam o igri.

Vrednovanje korisničkog iskustva bilo je podijeljeno u dva dijela. U prvom dijelu vrednovanja prelazak igre bio je osobno nadgledan od strane prvog autora kod šest ispitanika. Svi ispitanici bili su spremni odvojiti nekoliko sati svog vremena i prihvatili su biti dio istraživanja. Ispitanici su igranje igre *Inside* odradili na istom računalu gdje su u prostoriji bili samo ispitanik i ispitivač koji je promatrao napredak i reakcije igrača. Nakon odigravanja igre slijedilo je ispitivanje stavova i emocija igrača o igri pomoću upitnika. Prvo je primijenjen upitnik *System Usability Scale* ili skraćeno SUS koji omogućuje mjerenje upotrebljivosti neovisno o tehnologiji, što znači da se može

primijeniti na vrednovanju hardvera, softvera, web stranica itd. [9]. Sastoji se od deset izjava s pet ponuđenih odgovora, od „U potpunosti se ne slažem“ do „U potpunosti se slažem“. Nakon igranja igre ispitanici su popunjavali upitnik SUS koji je terminologijom bio prilagođen kontekstu video igara. Izjave iz upitnika predstavljene su u Prilogu 1.

U drugom dijelu vrednovanja korisničkog iskustva igre *Inside* korišten je upitnik *Game Experience Questionnaire* (GEQ). IJsselsteijn, de Kort i Poels [5] autori su upitnika koji ima modularnu strukturu te je podijeljen u tri dijela:

1. osnovni upitnik
2. modul društvene prisutnosti
3. modul nakon odigravanja igre

U ovom istraživanju koristili su se osnovni upitnik i modul nakon odigravanja igre jer je igra namijenjena jednom igraču (*single-player mode*) te se ne može ispitivati društvena prisutnost igrača. Osnovni upitnik pomoću sedam kategorija procjenjuje kako su se igrači osjećali za vrijeme igranja igre. Sastoji se od 33 izjave, a svaka izjava pripada jednoj od sedam kategorija: osjetilno i maštovito uranjanje ili imerzija, tok igre, kompetencija, pozitivni utjecaji, negativni utjecaji, napetost i izazov. Modul nakon igre procjenjuje kako se igrači osjećaju nakon što su prestali igrati igru, pomoću četiri kategorije: pozitivno iskustvo, negativno iskustvo, umor (prezasićenost) i povratak u stvarnost. Pitanja iz upitnika prikazana su u Prilogu 2.

Uz dosadašnjih šest ispitanika, u vrednovanje korisničkog iskustva uključili su novi ispitanici koji su prikupljeni putem društvenih mreža. Ispitanici su pozvani na ispunjavanje GEQ upitnika uz uvjet da su igru odigrali barem pola sata. Ispitanici su prikupljeni putem društvene mreže *Facebook* tako da je upitnik podijeljen na osobnom profilu autora i slanjem privatnih poruka prijateljima i poznanicima. Uz to, upitnik je podijeljen i na VoIP (eng. *Voice over Internet Protocol*) platformi *Discord* na kojoj autor i širi krug prijatelja i poznanika imaju kanal preko kojeg komuniciraju. U distribuciji upitnika primijenjena je i tzv. *snowbolling* metoda, odnosno bilo je uključeno prosljeđivanje poziva od strane već uključenih ispitanika prema potencijalnim drugim ispitanicima. Upitnik je bio otvoren devet dana za vrijeme kojih ga je ispunilo ukupno 30 ispitanika. Upitnik je izrađen pomoću alata *Google Forms* po uzoru na upitnik IJsselsteijna i suradnika [5] te je uključivao i neka opća pitanja o igranju video igara. Ispitanici su trebali ispuniti osnovni dio upitnika (eng. *GEQ Core*) u kojem su procijenili kako su se osjećali za vrijeme igranja igre i modul nakon igre (engl. *GEQ post-game*) u kojem su procijenili kako su se osjećali nakon igranja igre. Svaka kategorija ima svoju ocjenu, odnosno aritmetičku sredinu odgovora koji mogu biti u rasponu od „nipošto“ (0) do „u potpunosti“ (4).

Rezultati istraživanja

U ovom dijelu rada prikazat će se samo sumarni rezultati SUS i GEQ upitnika provedenih u sklopu istraživanja.

Ako bi se računao prosječni rezultat vrednovanja upotrebljivosti dobiven upitnikom SUS, unatoč malom broju ispitanika, on bi iznosio 67,91 što se može zaokružiti na 68. Kako navodi Sauro (2011), prosječan rezultat za SUS upitnik iznosi 68. To bi značilo da prema ovom istraživanju igra Inside ima prosječno ocijenjenu upotrebljivost. U tablici 1 prikazani su sumarni rezultati SUS upitnika.

Tablica 1: Sumarni rezultati upitnika SUS

Ispitanik	**Ispitanik 1**	**Ispitanik 2**	**Ispitanik 3**	**Ispitanik 4**	**Ispitanik 5**	**Ispitanik 6**
SUS vrijednost	72,5	77,5	50	70	72,5	60

Za četiri od šest ispitanika se može reći da im se igra svidjela, nekom više, nekom manje. Oni su u igri prepoznali više pozitivnih pojava od onih negativnih i zato su rezultati iznad prosjeka. Rezultati SUS upitnika za preostala dva ispitanika su manji i značajno odstupaju od rezultata ostalih ispitanika i od prosjeka za SUS upitnik. Na malom uzorku ispitanika prosječni rezultat može zavarati. Iako se ukupan prosječni rezultat čini nizak, dojam ispitivača je da su ispitanici u igri vidjeli mnogo pozitivnih stvari, čak bi se moglo reći da pozitivne stvari dominiraju. Ipak, ispitanici smatraju da neke stvari treba promijeniti, što je utvrđeno pomoću intervjua nakon igranja igre. Ispitanici su uglavnom zamjerali ponekad nepotrebno otežane zagonetke i ne baš intuitivnu kontrolu za hvatanje objekata. Igra bi se zasigurno mogla popraviti kad bi se neke zagonetke redizajnirale i na početku igre dodale kratke upute za korištenje kontrola ili postavljanje zadane kontrole za hvatanje objekata na lijevi *Ctrl* što je mnogo intuitivnije za korisnike.

U drugom dijelu vrednovanja sudjelovalo je trideset ispitanika koji su ispunjavali GEQ upitnik. Prije ispunjavanja upitnika popunili su i osnovne podatke o sebi i svom načinu igranja video igri. Rezultati govore da je 70% ispitanika muškog spola i da je najviše ispitanika starosti 21 i 22 godine. Najmlađi ispitanik imao je samo 9 godina, a najstariji 30. Što se tiče razine obrazovanja ispitanika, najzastupljenija je završena srednja škola s udjelom od 66,7%. Više od polovice ispitanika ima 7 ili više godina iskustva igranja video igara koje igraju najčešće nekoliko puta tjedno (36,7%) ili svaki dan (30%). Vrijeme koje ispitanici provedu igrajući igre u jednom danu je najčešće 1-2 sata (30%) ili 2-3 sata (26,7%). Igre najčešće igraju sami (46,7%) ili online s prijateljima/poznanicima (33,3%). Najpopularnija platforma među ispitanicima je PC (Mac/Linux) s udjelom od čak 86,7%. Preferirani žanrovi igara su akcijske igre, avanture i sportske igre, a 73,3% ispitanika već ima

iskustvo igranja 2D platformskih igri tako da su znali što mogu očekivati od igre *Inside*. Prije popunjavanja upitnika, igru Inside je 11 ispitanika igralo 3 ili više sati, 7 ispitanika ju je igralo manje od pola sata i 8 ispitanika ju je igralo do 1 sat vremena.

U tablici 2 prikazani su sumarni rezultati upitnika GEQ podijeljeni prema kategorijama.

Tablica 2: Sumarni rezultati upitnika GEQ

Osnovni dio (GEQ Core)	Prosječna ocjena
Kompetencija	2,43
Osjetilno i maštovito uranjanje ili imerzija	2,65
Tok igre	1,81
Napetost	1,24
Izazov	1,72
Negativni utjecaj	1,49
Pozitivni utjecaj	2,48

Nakon igre (GEQ Post-game)	Prosječna ocjena
Pozitivno iskustvo	1,98
Negativno iskustvo	1,41
Umor (prezasićenost)	1,45
Povratak u stvarnost	1,57

U osnovnom dijelu upitnika kategorije s najvećim ocjenama su osjetilno i maštovito uranjanje ili imerzija, pozitivni utjecaj i kompetencija. Prve dvije su očekivano visoko rangirane, dok je kompetencija malo iznenađenje jer se očekivao niži rezultat, ali igra je ostavila takav dojam na igrače. Kategorija s najmanjom ocjenom je napetost što je isto iznenađujuće jer u igri postoji mnogo napetih trenutaka gdje je potrebno pobjeći od neprijatelja, ali očito ispitanici to nisu doživjeli na takav način. U modulu nakon igre, ocjene su slične, ističe se jedino kategorija pozitivno iskustvo koja ima najveću ocjenu što govori da će igrači igru upamtiti više po pozitivnim stvarima nego negativnim.

Zaključak

Pojam korisničkog iskustva nije dovoljno prepoznat u razvoju video igara i u razvoju softvera općenito. Treba znati da se radi o subjektivnoj pojavi i ne može se objektivno reći da neka igra ima dobro ili loše dizajnirano korisničko iskustvo. Ipak, korisničko iskustvo se može vrednovati i u ovom radu prikazano je korištenje upitnika *System Usability Scale* za vrednovanje upotrebljivosti i upitnika *Game Experience Questionnaire* za vrednovanje korisničkog iskustva. Oba upitnika koristila su se za vrednovanje 2D

platformske igre *Inside*. Prema rezultatima vrednovanja, igra Inside ima prosječno ocijenjenu upotrebljivost ponajviše zbog nepotrebno otežanih zagonetki u nekim situacijama i neintuitivnih kontrola za hvatanje objekata u igri. Neki dijelovi igre bi se zasigurno mogli drugačije dizajnirati kako bi igrači bolje shvatili što trebaju napraviti i tako smanjili frustracije prilikom igranja. Što se tiče GEQ upitnika, može se zaključiti da je uranjanje ili imerzija igrača kategorija koja je najviše pridobila igrače, kao i kategorija pozitivno iskustvo u modulu nakon igre. Iznenađujuće malu ocjenu imaju kategorije napetost i izazov što govori da je igra ispitanicima bila dovoljno lagana za igranje i mogla bi biti neizvjesnija u nekim dijelovima.

Reference

[1] Anderson, J., McRee, J. i Wilson, R. (2010). *Effective UI* (1. izd.). O'Reilly Media

[2] Bernhaupt, R. (2010). User Experience Evaluation in Entertainment. U: R. Bernhaupt (ur.), *Evaluating User Experience in Games* (str. 3-7). Springer

[3] Calvillo-Gámez, E. H., Cairns, P. i Cox, A. L. (2010). Assessing the Core Elements of the Gaming Experience. U: R. Bernhaupt (ur.), *Evaluating User Experience in Games* (str. 47-71). Springer.

[4] Hassenzahl, Mark, Law, E. L.-C., & Hvannberg, E. T. (2006). User Experience – Towards a unified view. *User Experience – Towards a Unified View: Second International COST294-MAUSE Open Workshop – NordICHI'06*, str. 1–3.

[5] IJsselsteijn, W. A., de Kort, Y. A. W. i Poels, K. (2013). *The Game Experience Questionnaire.* Eindhoven: Technische Universiteit Eindhoven. Preuzeto 19.9.2019. s https://research.tue.nl/en/publications/the-game-experience-questionnaire

[6] International Standardization Organization (ISO). (2010). *ISO DIS 9241-210:2010. Ergonomics of human system interaction - Part 210: Human-centred design for interactive systems.* Preuzeto 10.7.2019. s https://www.iso.org/standard/52075.html

[7] McCarthy, J. i Wright, P. (2004). *Technology as Experience*. MIT Press.

[8] Mlinar, P. (2018). *Padobran za višekratnu uporabu bit će izbačen iz Fortnitea*. Preuzeto 19.9.2019. s https://www.hcl.hr/vijest/padobran-visekratnu-uporabu-bit-ce-izbacen-iz-fortnitea-130454/

[9] Sauro, J. (2011). *Measuring usability with the system usability scale (SUS)*. Preuzeto 10.7.2019. s https://measuringu.com/sus/

[10] Takatalo, J., Häkkinen, J., Kaistinen, J. i Nyman, G. (2010). Presence, Involment and Flow in Digital Games. U: R. Bernahupt (ur.), *Evaluating User Experience in Games* (str. 23-46). Springer.

Prilog 1. Upitnik *System Usability Scale* (SUS)

1. Mislim da bih često igrao/la ovu igru.
2. Igra je nepotrebno složena.
3. Mislim da je igra jednostavna za igranje.
4. Mislim da mi je potrebna tehnička osoba za igranje ove igre.
5. Razne funkcije u ovoj igri su dobro integrirane.
6. Mislim da ima previše nedosljednosti u ovoj igri.
7. Mislim da bi većina ljudi vrlo brzo naučila igrati ovu igru.
8. Igra je vrlo teška za igranje.
9. Osjećao/la sam se vrlo pouzdano igrajući ovu igru.
10. Morao/la sam naučiti puno stvari prije nego sam mogao/la igrati ovu igru.

Prilog 2. Upitnik *Game Experience Questionnaire* (GEQ)

2.1 Pitanja iz osnovnog dijela upitnika GEQ

1. Osjećao/la sam se zadovoljno.
2. Osjećao/la sam se vješto.
3. Zanimala me priča u igri.
4. Mislio/la sam da je igra zabavna.
5. Bio/la sam potpuno zaokupljen/a igrom.
6. Osjećao/la sam se sretno.
7. Igra me učinila loše raspoloženim/om.
8. Razmišljao/la sam o drugim stvarima.
9. Igru sam smatrao/la zamornom.
10. Osjećao/la sam se kompetentno.
11. Mislio/la sam da je igra teška.
12. Igra je estetski ugodna.
13. Zaboravio/la sam sve oko sebe.
14. Osjećao/la sam se dobro.
15. Bio/la sam dobar/ra u igri.
16. Osjećao/la sam se dosadno.
17. Osjećao/la sam se uspješno.
18. Osjećao/la sam se zamišljeno.
19. Osjećao/la sam da mogu istraživati stvari.
20. Uživao/la sam.
21. Bio/la sam brz/a u postizanju ciljeva igre.
22. Osjećao/la sam se neugodno.
23. Osjećao/la sam pritisak.
24. Osjećao/la sam se razdražljivo.
25. Izgubio/la sam trag vremena.
26. Osjećao/la sam se izazovno.

27. Smatrao/la sam igru impresivnom.
28. Bio/la sam duboko koncentriran/a u igri.
29. Osjećao/la sam se frustrirano.
30. Bilo je to bogato iskustvo.
31. Izgubio/la sam vezu s vanjskim svijetom.
32. Osjećao/la sam vremenski pritisak.
33. Morao/la sam uložiti mnogo truda u igru.

2.2 Pitanja iz *post-game* modula upitnika GEQ

1. Osjećao/la sam olakšanje.
2. Osjećao/la sam se loše.
3. Bilo mi je teško vratiti se u stvarnost.
4. Osjećao/la sam se krivim.
5. Osjećao/la sam se kao da sam pobijedio/la.
6. Smatrao/la sam da gubim vrijeme.
7. Osjećao/la sam se energično.
8. Osjećao/la sam se zadovoljan/a.
9. Osjećao/la sam se dezorijentirano.
10. Osjećao/la sam se iscrpljeno.
11. Osjećao/la sam da sam mogao/la učiniti više korisnih stvari.
12. Osjećao/la sam se moćno.
13. Osjećao/la sam se umorno.
14. Osjetio/la sam žaljenje.
15. Osjećao/la sam se posramljeno.
16. Osjećao/la sam se ponosno.
17. Imao/la sam osjećaj da sam se vratio/la iz avanture.

ICT-AAC aplikacije kao medij za komunikaciju i učenje

Ivana Rašan, Ivan Slivar, Matea Žilak, Željka Car i Jasmina Ivšac Pavliša*
Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva
*Sveučilište u Zagrebu Edukacijsko-rehabilitacijski fakultet
*{ivana.rasan, ivan.slivar, matea.zilak, zeljka.car}@fer.hr, *jivsac@erf.hr*

Sažetak

Rad opisuje rezultate multidisciplinarnih istraživanja Kompetencijske mreže ICT-AAC koja okuplja znanstvenike i stručnjake u području razvoja programskih rješenja za djecu s teškoćama u razvoju i djecu tipičnog razvoja u formi ozbiljnih igara za pokretne pametne uređaje i računala.

Ključne riječi: ozbiljne igre, djeca s teškoćama u razvoju, pristupačnost, komunikacija, jezične sposobnosti

Uvod

Kad se prije više od pet godina govorilo o projektu naziva *„Kompetencijska mreža zasnovana na informacijsko-komunikacijskim tehnologijama za inovativne usluge namijenjene osobama sa složenim komunikacijskim potrebama"* ili u skraćenoj verziji o *ICT-AAC projektu* kako u stručnim krugovima, tako i u široj javnosti, naziv projekta kao i njegov akronim zvučao je enigmatično te je predstavljao svojevrsnu nepoznanicu na ovom geografskom području. Kad se danas spomene *„Kompetencijska mreža ICT-AAC*[9]*"*, situacija je u potpunosti drugačija. Pozitivan kontekst na spomen ICT-AAC-a danas je popraćen različitim pozitivnim asocijacijama kao na primjer: **brojne razvijene mobilne i web aplikacije u formi ozbiljnih igara** koje se koriste u okviru stručne podrške za djecu s teškoćama u razvoju, ali i tijekom poučavanja djece tipičnog razvoja. Multidisciplinarni tim stručnjaka i znanstvenika ulaže značajne napore u istraživanja mogućnosti novih tehnologija za jačanje kapaciteta korisnika igara, ali i u kontinuirano praćenje novih mogućnosti za proširenje suradnje u različitim područjima.

Multidisciplinarni pristup nije jednostavan u svojoj primjeni, kada se na jednom mjestu okupe stručnjaci iz npr. četiri različita područja, ali je danas postao svojevrsni karakteristični *mentalni sklop* članova ICT-AAC tima. ICT-AAC tim je prepoznat od strane različitih dionika ne samo na području RH već i šire što je rezultiralo brojnim projektnim i istraživačkim suradnjama. Uz rad

[9] Kompetencijska mreža ICT-AAC, www.ict-aac.hr

na razvoju mobilnih i web aplikacija za potpomognutu komunikaciju i za učenje, tim značajno djeluje i na području osvješćivanja šire javnosti o izazovima s kojima se susreću djeca s teškoćama u razvoju i osobe s invaliditetom. Osnova za razvoj ICT-AAC aplikacija su igra i uvažavanje razvoja pojedinih vještina, te se timski rješavaju izazovi kako određeni korisni sadržaj oblikovati u što atraktivnijem digitalnom obliku koji će potom polučiti maksimizaciju efekta korištenja istog kod krajnjih korisnika. U nastavku rada bit će predstavljene ICT-AAC aplikacije podijeljene u nekoliko kategorija s obzirom na svrhu korištenja, analizirane korištene tehnologije te prikazan osvrt na njihovu primjenu. Aplikacije imaju mogućnost dodavanja vlastitog korisničkog sadržaja, veliku razinu prilagodbe korisničkog sučelja potrebama korisnika te ugrađene opcije glasovnog izgovora sadržaja, uz mogućnost dodavanja vlastitih zvučnih zapisa.

ICT-AAC aplikacije za podizanje svijesti o pristupačnosti

Ozbiljne igre imaju za cilj podizanje svijesti javnosti, a pogotovo studenata i ostalih uključenih u dizajn i razvoj pristupačnih rješenja društva. **Kviz**[10] je web-aplikacija koja omogućuje korisniku zorni prikaz barijera s kojima se susreću osobe s invaliditetom te o načinima kako se te barijere mogu umanjiti pravilnim dizajnom sjedišta weba. **Pričajmo slikama**[11] je igra u kojoj se igrač upoznaje s potpomognutom komunikacijom koja je namijenjena svima koji ne mogu komunicirati uobičajenim putem.

ICT-AAC aplikacije za potpomognutu komunikaciju

Ove aplikacije namijenjene su različitim profilima korisnika te im je primarna uloga poticanje komunikacije. Obilježja korisnika, kao što su vrsta teškoća, individualne potrebe i razina komunikacijskog i jezično-govornog razvoja, određuju način i svrhu za koje će se aplikacija koristiti. **ICT-AAC Komunikator**[12] nudi mogućnost slaganja i prikaza više ekrana sa simbolima za potpomognutu komunikaciju kroz koje se može kretati ”listanjem”. Dodirom slike simbola na ekranu aplikacija reproducira odgovarajući zvučni zapis. Uz zvučni zapis, svakom simbolu pridružen je i odgovarajući tekst koji se može prikazivati ispod slike simbola, ovisno o postavkama aplikacije. Komunikator sadrži temeljne rječničke kategorije (aktivnosti i događaji, glagoli, mjesta...) s odgovarajućim skupom simbola koji se mogu nadograditi vlastitim simbolima ili fotografijama čime omogućava prilagodbu

[10] HAKOM Kviz, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/web-aplikacije/hakom-kviz
[11] Pričajmo slikama, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/web-aplikacije/pricajmo-slikama
[12] ICT-AAC Komunikator, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/apple-ios-aplikacije/komunikator

individualnim potrebama korisnika. **ICT-AAC Komunikator+**[13] (Slika 1) omogućuje slaganje i prijenos fraza i rečenica pomoću postojećih simbola iz tri nekomercijalne galerije (ARAASAC, Mulberry i Sclera), nadogradnju vlastitim simbolima ili fotografijama. Frazu je moguće reproducirati uz postojeći zvučni zapis simbola ili snimiti vlastiti zapis i reproducirati ga. Uporabom vlastitih fotografija i zvučnog zapisa ovakav sustav potpomognute komunikacije moguće je prilagoditi individualnim potrebama korisnika.

Slika br. 7: Pričajmo slikama (lijevo); Komunikator + (desno)

ICT-AAC aplikacije za jačanje jezičnih sposobnosti i predvještina čitanja i pisanja

Ove aplikacije su namijenjene djeci s različitim vrstama teškoća, ali i djeci tipičnog razvoja školske i predškolske dobi u svrhu poučavanja te logopedske podrške za odrasle osobe. **ICT-AAC e-Galerija**[14] omogućava slaganje priča pomoću niza sličica koje mogu biti fotografije snimljene integriranom kamerom uređaja, slike iz galerije uređaja ili simboli iz triju nekomercijalnih galerija. Djeci rane dobi koja imaju odstupanja ili teškoće u razvoju potrebna je vizualna podrška koja im olakšava prizivanje prošlih događaja i/ili razumijevanje uzročno-posljedičnog slijeda što im ova aplikacija i omogućava. Opisana aplikacija ujedno omogućava i poticanje konverzacije. **ICT-AAC e-Galerija Senior**[15] je aplikacija namijenjena poticanju jezičnog izražavanja u osoba čije teškoće nastaju uslijed oštećenja mozga (afazije i traumatske ozljede mozga). Osim poticanja samostalnog izražavanja temeljem praćenja slijeda događaja, potiče se prepričavanje i stvaranje uzročno-posljedičnih veza. **ICT-AAC Komunikacijski ključevi**[16] je aplikacija namijenjena logopedima koji pružaju podršku osobama s afazijom. Jedna od najčešćih teškoća na koje

[13] ICT-AAC Komunikator+, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/komunikator-plus
[14] ICT-AAC e-Galerija, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/e-galerija
[15] ICT-AAC e-Galerija Senior, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/egalerija-senior-android
[16] ICT-AAC Komunikacijski ključevi, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/komunikacijski-kljucevi

nailaze osobe s afazijom jesu teškoće imenovanja. Aplikacija olakšava imenovanje simbola uz pomoć pisanih i akustičkih ključeva.

ICT-AAC Slovarica[17] na atraktivan način pruža rana iskustva s pismom, uparenim slikovnim i zvučnim zapisom. ICT-AAC Glaskalica[18] je prva aplikacija na hrvatskom jeziku koja je osmišljena za prijenosne uređaje i pomaže pri savladavanju fonološke svjesnosti koja predstavlja jednu od osnovnih predvještina čitanja. Koncept aplikacije uvažava razvojni slijed fonološke svjesnosti kao i razinu fonološke složenosti riječi. ICT-AAC Mala Glaskalica[19] je verzija aplikacije Glaskalica proizašla temeljem zahtjeva Udruge roditelja OKO i Hrvatske zajednice za Down sindrom. Od Glaskalice se razlikuje prema skupu riječi za koje se pogađaju glasovi, kao i dizajnu aplikacije. ICT-AAC Pamtilica[20] potiče usvajanje novih riječi te utvrđivanje veze slovo-glas kroz koncept pronalaženja i upamćivanja parova.

ICT-AAC Pisalica[21] olakšava učenje pravilnog pisanja velikih i malih tiskanih slova koje nije zastupljeno u aplikacijama na hrvatskome jeziku. ICT-AAC Jezična gradilica[22] namijenjena je poticanju morfološkog razvoja djece predškolske dobi. Na vizualno atraktivan i zabavan način približava gramatičke morfeme za imensku i glagolsku množinu. ICT-AAC Učimo prijedloge[23] nudi mogućnost poučavanja jednostavnijih ili složenijih prijedloga te mogućnost odabira razine igre ovisno o stupnju zahtjevnosti.

ICT-AAC aplikacije koje prate nastavno gradivo

ICT-AAC Vremenski vrtuljak[24] na zabavan i interaktivan način upoznaje djecu s godišnjim dobima, danima u tjednu i mjesecima u godini te olakšava usvajanje točnog redoslijeda. **ICT-AAC Koliko je sati** potiče snalaženje u vremenu pomoću iskazivanja trajanja događaja u vremenskim jedinicama te učenje koliko je sati na analognom i digitalnom satu (slika 2).

[17] ICT-AAC Slovarica http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/apple-ios-aplikacije/slovarica
[18] ICT-AAC Glaskalica, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/glaskalica
[19] ICT-AAC Mala Glaskalica, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/apple-ios-aplikacije/mala-glaskalica
[20] ICT-AAC Pamtilica, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/pamtilica
[21] ICT-AAC Pisalica, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/pisalica
[22]ICT-AAC Jezična gradilica, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/apple-ios-aplikacije/ict-aac-jezicna-gradilica
[23] ICT-AAC Učimo prijedloge, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/ict-aac-ucimo-prijedloge-android
[24] ICT-AAC Vremenski vrtuljak, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/vremenski-vrtuljak

ICT-AAC aplikacije razvijene za djecu s višestrukim teškoćama

U suradnji sa stručnjacima iz Specijalne bolnice Gornja Bistra razvijene su dvije aplikacije koje su rezultat prepoznatih značajki digitalnih uređaja koje djeci s višestrukim oštećenjima mogu pružiti digitalno stimulativno okruženje. **ICT-AAC Prskalice**[25] služi za podučavanje uzročno-posljedične veze putem prezentacije jednostavnih podražaja različitih modaliteta koji su razvojno primjereni za djecu s višestrukim teškoćama i djecu rane dobi. **ICT-AAC Učimo boje**[26] osmišljena je kao motivacija i svojevrsni alat za upoznavanje s bojama. Temelji se na korištenju audiovizualnih elemenata bliskih djeci s naglaskom na jednostavnost i jasnu prepoznatljivost (slika 2). Obje aplikacije mogu koristiti i djeca tipičnog razvoja.

Slika br. 8: Koliko je sati (lijevo); Učimo boje (desno)

ICT-AAC aplikacije prilagođene djeci s poremećajem iz spektra autizma

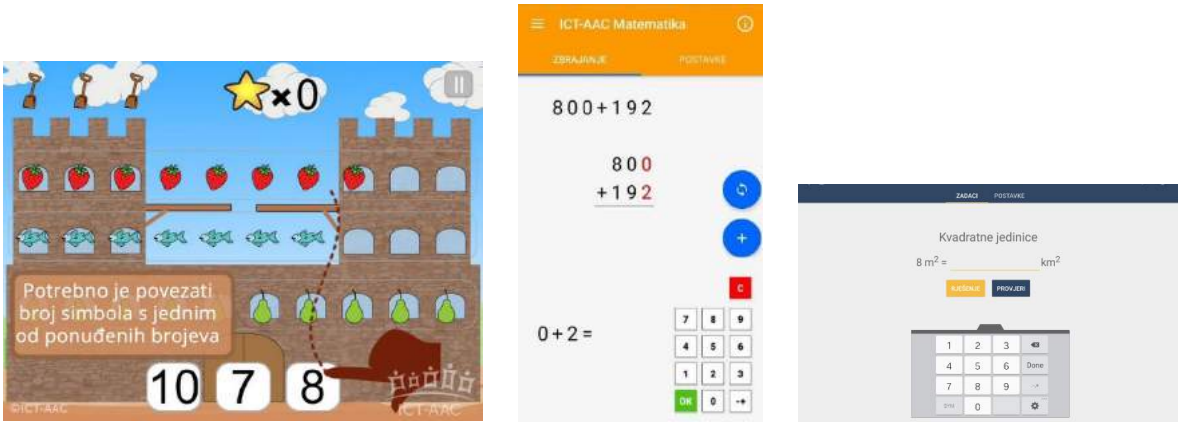
MULTI-SKLAD Ponašalica[27] olakšava razumijevanje različitih socijalnih situacija putem vizualne podrške u vidu grafičkih simbola ili fotografija. **MULTI-SKLAD Vizualni raspored**[28] je podrška u prihvaćanju prijelaza iz aktivnosti u aktivnost, generalizaciji naučenih vještina, promjenama u okolini te omogućava sekvencioniranje zadataka s više koraka ili više uzastopnih događaja.

Aplikacije za poticanje matematičkih (pred)vještina

U matematičkom opismenjavanju djece svoj doprinos daju „matematičke“ ICT-AAC aplikacije kojima se potiče razvoj matematičkih predvještina i vještina, te savladavanje osnovnih i naprednih matematičkih operacija. Djeci je jednostavnije razumjeti matematičke koncepte ako su ih imali prilike iskusiti

[25] ICT-AAC Prskalice http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/prskalice
[26] ICT-AAC Učimo boje http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/ucimo-boje
[27] [27] MULTI-SKLAD Ponašalica http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/ponasalica
[28] MULTI-SKLAD Vizualni raspored http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/web-aplikacije/aplikacija-vizualni-raspored

na konkretnim primjerima te je u tome vrijednost ovih aplikacija. **ICT-AAC Matematička igraonica**[29] je aplikacija kojom se potiče razvoj sljedećih matematičkih (pred)vještina: razumijevanje postojanja objekta, sposobnost prepoznavanja i razlikovanja količine (slika 3), manipulacija brojevima, razumijevanje redoslijeda brojeva te općenito razumijevanje brojevnog sustava). **ICT-AAC Matematički vrtuljak**[30] jedna je od najpopularnijih ICT-AAC aplikacija koja predstavlja značajan doprinos razvoju matematičke pismenosti. Aplikacijom se potiče nekoliko značajnih vještina: brojanje, razlikovanje i uspoređivanje brojevnih količina, zbrajanje i oduzimanje do 10 te korištenje svih računskih operacija s brojevima do 100. **ICT-AAC Matematika**[31] omogućuje uvježbavanje operacija vezanih uz pisano zbrajanje i oduzimanje, množenje i dijeljenje (slika 3). Aplikacija se pokazala kao koristan materijal u svim aspektima nastave od same obradbe i motiviranja učenika do vježbanja kroz igru. **ICT-AAC Učimo mjere**[32] omogućuje aktivno učenje i samostalno vježbanje pretvaranja mjernih jedinica (slika 3).



Slika br. 3: Mat. igraonica (lijevo); Matematika (sredina); Učimo mjere (desno)

[29] ICT-AAC Matematička igraonica, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/web-aplikacije/matematicka-igraonica
[30] ICT-AAC Matematički vrtuljak, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/matematicki-vrtuljak-android
[31] ICT-AAC Matematika, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/matematika
[32] ICT-AAC Učimo mjere, http://www.ict-aac.hr/index.php/hr/ict-aac-razvijene-aplikacije/android-aplikacije/ucimo-mjere

Tablica br. 3: Popis ICT-AAC aplikacija prema platformi za koju su razvijene

Platforma	Nazivi aplikacija	Ukupan broj aplikacija
Android	Komunikator, Komunikator+, e-Galerija, e-Galerija Senior, Matematička igraonica, Domino brojalica, Matematički vrtuljak, Matematika, Slovarica, Glaskalica, Pamtilica, Prepoznaj pojmove, Učimo slogove, Učimo riječi, Učimo čitati, Ponašalica, Prskalice, Vremenski vrtuljak, Komunikacijski ključevi, Učimo mjere, Učimo boje, Koliko je sati, Jezična gradilica, Pisalica, Učimo prijedloge	25
iOS	Komunikator, Komunikator+, e-Galerija, e-Galerija Senior, Matematička igraonica, Domino brojalica, Matematički vrtuljak, Slovarica, Glaskalica, Mala Glaskalica, Pamtilica, Ponašalica, Prskalice, Učimo boje, Koliko je sati, Jezična gradilica, Pisalica	17
Web	Matematička igraonica, Matematički vrtuljak, Glaskalica, Vizualni raspored, HAKOM Kviz, Pričajmo slikama, Učimo boje, Koliko je sati, Jezična gradilica, Pisalica, Učimo prijedloge	11

Razvojne tehnologije

Do sada je u okviru portfelja razvijeno preko 50 aplikacija za pokretne uređaje s popularnim operacijskim sustavima Android i iOS te za pokretanje u popularnim internetskim preglednicima. Popis razvijenih aplikacija po platformama dostupan je u sljedećoj tablici (Tablica 1). Sve aplikacije dostupne su besplatno na internetskim trgovinama aplikacija Google Play i App Store te online na web-stranicama Kompetencijske mreže ICT-AAC[33].

Zaključak

Napori u istraživanjima ICT-AAC tima su se posljednjih godina usmjerili i na primjenu novih, suvremenih tehnologija u području poučavanja djece tipičnog razvoja, ali posebno u poticanju i poučavanju djece i osoba s teškoćama u razvoju. Istraživanja su usmjerena prema beskontaktnim sučeljima [1], virtualnoj i proširenoj stvarnosti (VR/AR) [2], bežičnim tehnologijama (RFID, beacon i sl.), pametnim nosivim uređajima [3] te analizi i implementaciji pristupačnosti višemedijskih sadržaja [4][5][6]. Razvijeno je i korisnički evaluirano nekoliko prototipova čija je svrha ispitati potencijal novih tehnologija u kontekstu poboljšanja kvalitete života osoba s teškoćama. Aplikacije se koriste širom cijele regije, a podaci o preuzimanjima govore i o dalekim zemljama poput SAD-a, Brazila itd. u dječjim vrtićima, školama, logopedskim kabinetima, kabinetima edukacijskih-rehabilitatora i na ostalim mjestima gdje se uvažavaju individualne potrebe djeteta i osobe te gdje se uči kroz igru.

[33] ICT-AAC Aplikacije, http://ict-aac.hr/index.php/hr/aplikacije

Reference

[1] Šoić, R., Vuković, M., Car, Ž. (2017). Enabling Text-To-Speech Functionality for Websites and Applications Using a Content-Derived Model. EAI Endorsed Transactions on Ambient Systems, 13, pp. 1-7.

[2] Žilak, M; Car, Ž; Ježić, G. (2018). Educational Virtual Environment Based on Oculus Rift and Leap Motion Devices. CSRN, 2802, pp. 143-151.

[3] Vuković, M., Car, Ž., Ivšac, J., Mandić, L. (2017). Smartwatch as an Assistive Technology: Tracking System for Detecting Irregular User Movement. Intl. Journal of E-Health and Medical Communications (IJEHMC), 9. pp.23-34.

[4] Feješ, A., Ivšac Pavliša, J., Slivar, I. (2014). Communication and language and a AAC system-a case of a girl with Wolf-Hirschhorn syndrome. Discover Communication, pp. 43-43.

[5] Dolić, J., Pibernik, J., Car, Ž. (2013). Design and Development of Symbol Based Services for Persons with Complex Communication Needs. Acta graphica, 24 ,1/2; pp. 19-28.

[6] Semanjski, I., Saric, T., Janda-Hegedis, Z., Car, Z., Vukovic, M. (2012). e-Accessible service system: Calibrator and Communicator. Lecture Notes in Computer Science, 7327, 241-250.

Izrada multiplatformske igre pomoću programskog jezika OpenGL i Jave

Marijana Borovec i Marko Prosen
Diplomanti Fakulteta organizacije i informatike, Sveučilište u Zagrebu
Vlastiti projekt
marborovec@gmail.com, maprosen@gmail.com

Sažetak

U radu će biti opisani ključni dijelovi izrade multiplatformske igre pomoću programskog jezika OpenGL i Jave, uz korištenje pomoćnih alata. Dotaknut ćemo se razvoja, mehanike igre, tima, animacija, mapa i priče te načina komponiranja svih elemenata u 2D igricu.

Ključne riječi: libGDX, algoritmi, programiranje, animacije, Java

Uvod

Taktička obmana je multiplatformska igra razvijena u programskom jeziku Java uz korištenje OpenGL-a, dok se na mobilnim uređajima Apple-a programski kod kompajlira uz pomoć RoboVM kompajlera. Podržane platforme su: Windows, Android, Linux, IOS i MacOS.

Inspiracija za igricu dolazi iz taktičkih igra iz 90-tih poput X-Com, Jagged Alliance.

Radi se o strateškoj igri na poteze u kojoj igrač kontrolira više članova tima. Svaki član tima ima svoju priču koja je usko povezana s vještinama koje članovi tima posjeduju. Igrica dopušta igraču da sam odabere koje vještine će tim naučiti kroz igru. Što igra ide dublje u priču, to se i tim sve više razvija. Igra ima više mogućih završetaka koji ovise o tome koji put će odabrati igrač.

Razvoj

Značajni algoritmi koji su implementirani bez korištenja vanjskih biblioteka:

- Algoritam vidljivosti od 90 stupnjeva (engl. Ray casting) [1]
- Heuristički A* algoritam za pronalaženje suboptimalnog puta kretanja [2]
- Flood Fill algoritam koji izračunava sve dostupne kretnje kod igrača [3]

- Object pool uzorak dizajna koji iskorištava postojeće alocirane objekte jer je operacija GC (engl. Garbage Collector) sakupljača smeća skupa, pa se time izbjegava pokretanje GC-a. [4]

Performanse

Moderni procesori nemaju problema s pokretanjem ovakvih 2D igrica, no kod mobilnih platformi itekako treba paziti na performanse. Prilikom dizajna odlučili smo držati se sljedećih smjernica:

- Maksimalna rezolucija tekstura: 2048X2048 pixela – kod nekih grafičkih procesora mobilnih uređaja je ovo limit.
- Object pool – iskorištavanje alociranih objekata radi izbjegavanja kolekcije smeća.
- Minimiziranje potrebe za promjenama tekstura jer prilikom izmjene tekstura OpenGL (glFlush) mora poslati podatke prema grafičkom procesoru koji može izazvati pad performansi. [5]
- Optimizacija Shader-a prema OpenGL 2.0 standardu.

Mehanika igre

Igra se temelji na potezima, tako da svaki lik ima određeni broj akcijskih bodova koje može iskoristiti kada je na redu za igranje. Igra počinje s tri akcijska boda koji se mogu povećavati zavisno o situacijama na misiji. Dodani akcijski bodovi se mogu dobiti:

- ako lik primijeti da je drugi član tima kojem pripada pogođen
- ako se oglasio alarm
- ako vođa tima ima razvijenu vještinu dodavanja dodatnih akcijskih bodova

Količina akcijskih bodova je označena kao bijeli pravokutnici iznad svakog lika. Na slici 1 se može vidjeti broj akcijskih bodova našeg tima, ali i protivničkog. U ovom slučaju protivnik nema niti jedan akcijski bod.

O broju akcijskih bodova ovisi i koliko se određeni lik može kretati kroz mapu. Pucanje uzima jedan akcijski bod, osim u slučaju snajpera kojem uzima dva akcijska boda. No i to je moguće reducirati na način da nauči vještinu koja igraču omogućuje da gubi samo jedan akcijski bod kod svakog pucanja.

Rotacija oko vlastite osi uzima akcijske bodove, no tek kod svake pete rotacije.

Slika br. 1: Prikaz broja akcijskih bodova lika

Slika br. 2: Prikaz pozicija i usmjerenja

Lik se može kretati u 4 različita smjera, ovisno o orijentaciji, no trčati se može samo naprijed. Trčanje ubrzava kretnju, no isto tako smanjuje preciznost ciljanja.

Lik može biti u tri pozicije dok stoji mirno:

- Stajanje – lik može hodati i pucati
- Klečanje – lik ne može hodati, ali može pucati
- Na leđima – lik se ne može kretati ni pucati

Pucanje

Protivnik na vidiku pruža liku mogućnosti pucanja ako ima dovoljno akcijskih bodova. Ako lik nema akcijskih bodova mora pričekati slijedeći red i tada iskoristiti svoju priliku.

Neprijatelj na vidiku ne znači i pogodak. Ako je preciznost ciljanja niska, recimo radi trčanja kao što smo naveli prije, postoji velika mogućnost da će lik promašiti pogodak. Isto tako sa svakim pucanjem postotak preciznosti se povećava, čime se povećava i mogućnost pogotka. Postotak preciznosti se može vidjeti iznad lika u obliku plave trake. Ako se lik nalazi blizu neprijatelja njegov postotak preciznosti će uvijek biti 100%.

Pokrivanje je još jedna situacija u kojoj lik ima mogućnost pucanja tijekom neprijateljevog poteza. Ako je lik u pokrivanju i naiđe neprijatelj, on će pucati onoliko puta koliko mu je preostalo akcijskih bodova i s preciznošću ciljanja koja mu je preostala.

Slika br. 3: Pucanje i pokrivanje

Tim

Tim se sastoji od više jedinstvenih likova sa svojim pričama i vještinama koje igrač razvija kroz igricu.

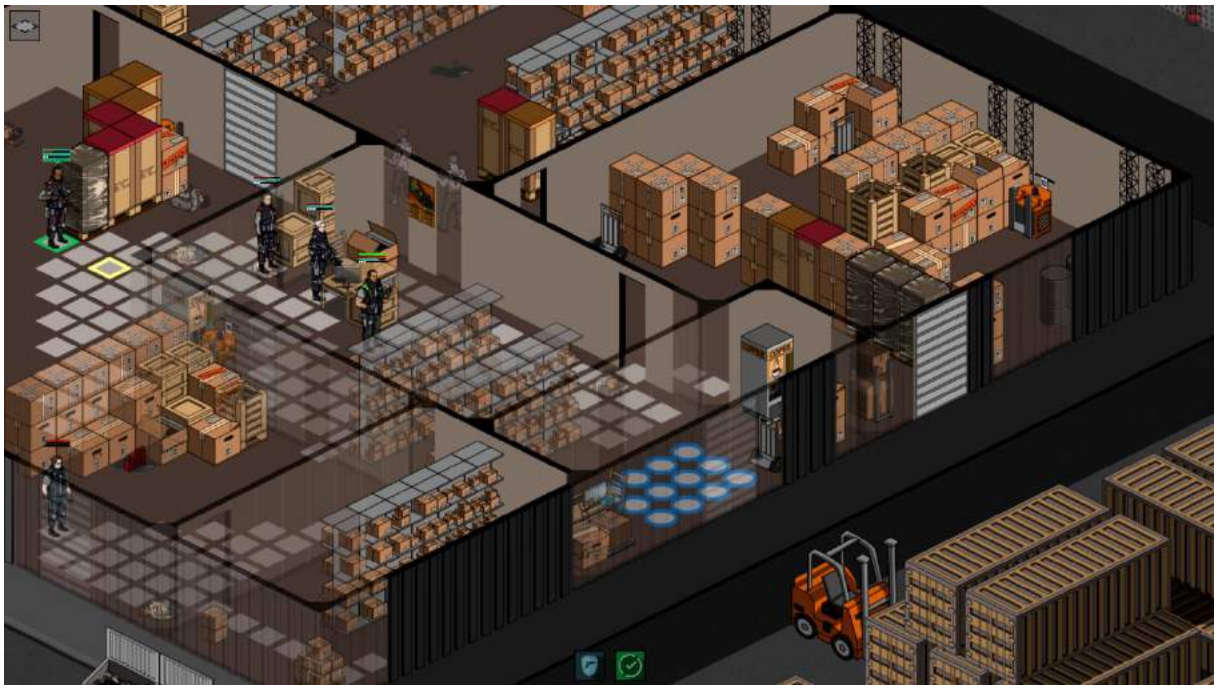
Vještine članova tima mogu se podijeliti na slijedeće kategorije:

- Voditelj tima – mogućnost podizanja morala timu (preko akcijskih bodova).
- Razbijač – zbog jačeg oklopa može primiti više udaraca neprijateljske strane.
- Snajper – veća preciznost oružja na veću udaljenost.
- Tihi ubojica – prigušivač na pištolju omogućuje pucanje bez podizanja alarma neprijatelju.
- Haker – sposobnosti skeniranja mu omogućuju otkrivanje pozicija neprijatelja u susjednim sobama.
- Doktor – sposobnost liječenja tima i sebe.

Članovi tima su ti koji čine kompletan tim te su potrebne vještine svih članova tima kako bi uspješno pobijedili protivnika i uspješno završili misiju. Razvoj vještina ovisi o iskustvu stečenom kroz igricu, a ono se stječe pomoću eliminiranja neprijatelja i korištenja specijalnih vještina. Postoji i dodatni bonus koji se dobije ukoliko alarm neprijatelja nakon završetka misije nije zazvonio.

Vidno polje 90 stupnjeva

Svaki lik vidi ono što mu se nalazi pod kutom od 90 stupnjeva, što igraču ostavlja prostora za šuljanje i prikriveno kretnje bez alarmiranja neprijatelja. No to isto tako povećava mogućnost šuljanja neprijatelja timu iza leđa ili prikrivanja u prostorijama izvan igračevog vidnog polja.



Slika br. 4: Prema bijelim pravokutnicima se mogu vidjeti vidna polja likova (90 stupnjeva)

Slika br. 5: Primjer razgovora s likom

Animacije

Animacije su podijeljene na dva dijela: organsko kretanje likova u misijama (hodanje, trčanje, pucanje, padanje) te kretanja u stripovima i razgovor s likovima.

Sve animacije su rađene pomoću alata Spine (Esoteric Software) čiji fokus je na 2D animacijama za igrice. Animacije nisu rađene pomoću klasičnih sličica (engl. Sprite sheet) već pomoću matematičkih transformacija pojedinih dijelova tijela likova.

Misije

Kroz igricu se igrač može susresti s nekoliko tipova misija, a to su: obrana, napad, demoliranje, krađa, podmetanje, spašavanje, eliminiranje, hakiranje. Svaka od misija je uspješna samo ako su svi u timu živi i ako je misija uspješno obavljena.

Igrač mora biti veoma oprezan kod izbora članova tima za određenu misiju, uspješnost misije ovisi o tome. Na neke misije mogu ići svi članovi tima, no na određene misije se može birati samo po nekoliko članova. Bitno je izabrati one koji imaju najbolje vještine za misiju koja igraču slijedi.

Misije se ne mogu preskakati i svaka mora biti uspješno izvršena kako bi se prešlo na daljnje misije.

Mape

Taktička obmana sadrži 20-tak različitih mapa kroz koje igrač prolazi i kkoje otkriva. Svaka mapa pripada određenoj frakciji, dakle igrač ulazi u njihova glavna sjedišta, zgrade ili čak skrivene lokacije.

Mape su jedinstvene, ručno nacrtane u isometric perspektivi. Svaka od mapa reflektira osobnost frakcije kojoj pripada i samim time se može vidjeti njihov potpis u svakom kutku.

Mape nisu odmah otkrivene, dio koji igrač nije istražio je u crnom. Dakle, otkrivene su onoliko koliko ih je igrač istražio. Svakim otvaranjem prostorije unutar mape igrač dobije uvid u još jedan dio mape.

Alat koji se koristio za izradu mapa je Tiled Map Editor koji omogućuje izradu 2D levela. Ono što je interesantno u ovom alatu je da dopušta osim izradu mapa i unos raznih dodatnih informacija koje su korisne za funkcioniranje igrice. Dodavanjem dodatnih informacija na elemente mape, dobiva se dodatna fleksibilnost kod prikaza mapa i elemenata na njoj.

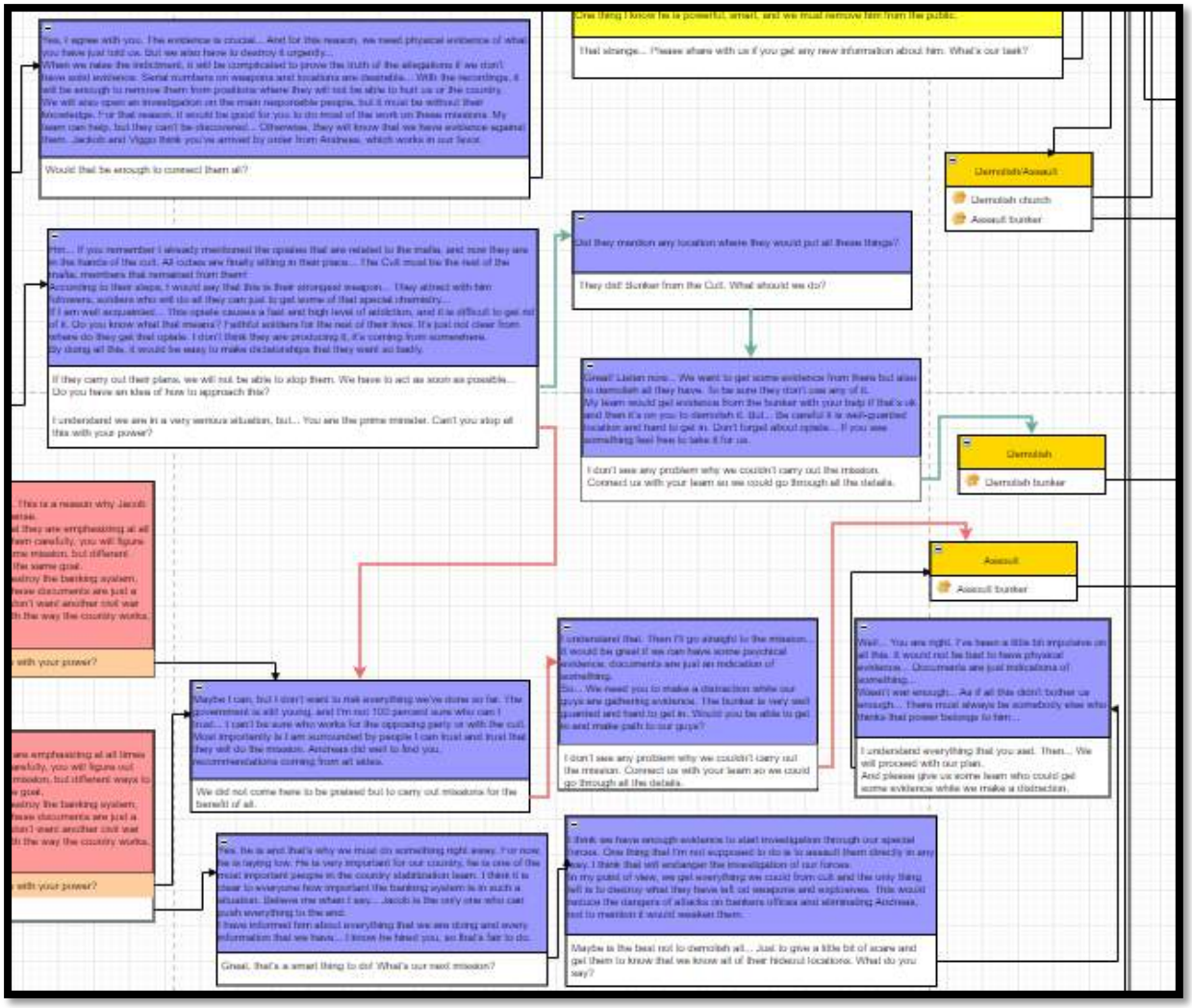
Slika br. 6: Primjer mape

Mehanika donošenja odluka (Decision-making mechanic)

Mehanika donošenja odluka u igrici dovodi u igricu različito iskustvo svaki put kada igrač odluči ponovo igrati igricu od početka. Sam igrač stvara svoju priču, stvara neprijatelje i prijatelje usput, odlučuje na čijoj strani se želi boriti.

Sama priča je puna intriga, laži i odluka kome vjerovati. Iz tog razloga igrač mora oprezno birati kojim putem želi ići, jer će njegov izbor utjecati na avanture kroz igricu i izbor misija koje će igrati.

Iz svih navedenih razloga razvoj priče je nelinearan, što znači da svaki izbor u razgovoru s određenim likom igrača možda vodi i na različiti put. Ponekad i izbor misije igrača vodi u sasvim novom smjeru, u kojem ako igra drugi put neće završiti.



Slika br. 7: Primjer mehanike donošenja odluka (zeleni izbor vodi na različitu misiju u odnosu na crveni izbor)

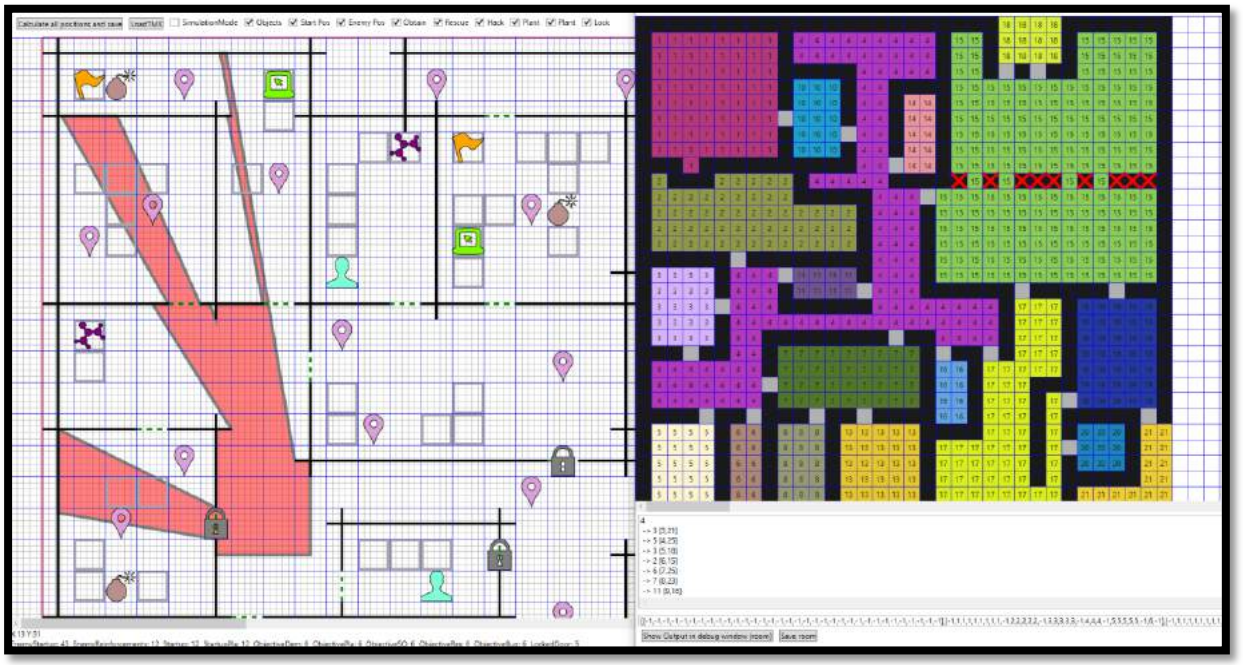
Pomoćni alati koji su se koristili za izradu igrice

Izrada igrice Taktička obmana je zahtijevalo određene pomoćne alate koje smo razvijali prema potrebi u programskom jeziku C#/WPF:

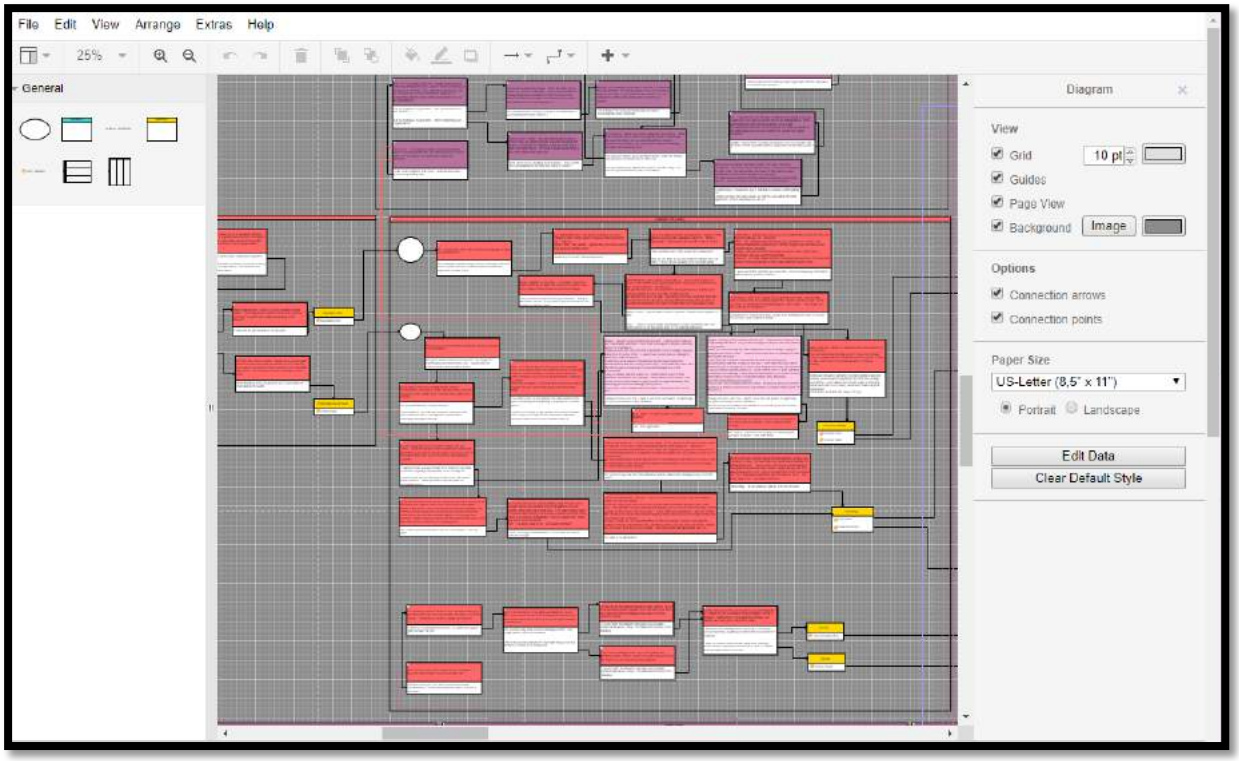
- Alat za postavljanje parametara na objekte zidova – služio nam je kako bi definirali usmjerenje zida, tip zida (otvorena/zatvorena vrata, specijalni zid).
- Alat za izračunavanje položaja svih objekata na mapi – izračun položaja objekata, zidova, mogućih lokacija neprijatelja, specijalnih objekata za misije i zaključanih vrata.
- Alat za izradu grafa nelinearne priče – izrada nelinearne priče preko grafa u kojem se definiraju razgovori, veze između razgovora/odgovora, te se definiraju misije i tijek nakon njih.
- Alat za čitanje i izvoz priče u format pogodan za igricu – alat pruža lakši način provjere razgovora između tima i sugovornika. Ono daje dojam

koji se dobije koji se čita u igrici te služi za lakšu provjeru tijeka priče. Alat također omogućuje višejezičnost i izvoz u JSON format koji se poslije uvozi u igricu.

Slika br. 8: Alat za postavljanje parametara na objekte zidova



Slika br. 9: Alat za izračun svih objekata na mapi



Slika br. 10: Alat za izradu nelinearne priče

Zaključak

Iako su danas pojedini alati za izradu igrica veoma razvijeni, kao što su to primjerice Unity i Unreal Engine, mi smo odabrali framework libGDX zbog boljeg uvida u izradu igrica. Naišli smo na izazove koji su elegantnije riješeni u navedenim modernijim alatima, ali smo obogatili znanje o funkcioniranju game engina „ispod haube“. Pozitivna strana ovog pristupa razvoju je da se može bolje optimizirati algoritam prema zahtjevima igrice.

Reference

[1] https://permadi.com/1996/05/ray-casting-tutorial-table-of-contents/, posjećeno 7.9.2019.
[2] https://en.wikipedia.org/wiki/A*_search_algorithm, posjećeno 7.9.2019.
[3] https://en.wikipedia.org/wiki/Flood_fill, posjećeno 7.9.2019.
[4] https://en.wikipedia.org/wiki/Object_pool_pattern, posjećeno 7.9.2019.
[5] https://developer.apple.com/library/archive/documentation/3DDrawing/Conceptual/OpenGLES_ProgrammingGuide/OpenGLESApplicationDesign/OpenGLESApplicationDesign.html#//apple_ref/doc/uid/TP40008793-CH6-SW3, posjećeno 7.9.2019.
https://developer.android.com/guide/topics/graphics/opengl, posjećeno 7.9.2019.
https://developer.samsung.com/game/opengl, posjećeno 7.9.2019.

Web tehnologije za prikaz 3D objekata

Aleksandar Trajkov i Mario Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
aletrajko@foi.hr, mario.konecki@foi.hr

Sažetak

Za lakši i brži razvoj temeljnih domena čovječanstva, kao što su inženjerstvo, medicina, strojarstvo, arhitektura ili fizioterapija, potrebno je primijeniti tehnologije koje su bogate potencijalom. Upravo u ovom kontekstu do izražaja dolazi primjena 3D tehnologije. Već integrirani kao web platforme, alati Sketchfab, Google Poly i p3D.in nude interaktivno iskustvo prikaza 3D objekata, kako jednostavnih tako i kompleksnijih scena. Mogućnosti je mnogo, ali jedno je sigurno - razvoj 3D tehnologije bit će od rastućeg značenja ne samo u industriji videoigara i zabave nego i u ostalim sferama ljudske svakodnevnice.

Ključne riječi: 3D, tehnologija, videoigre, razvoj, napredak, inovacije, web

Uvod

U proteklom desetljeću gotovo da ne postoji tehnologija koja je ostvarila veliki iskorak, a da se i dalje kontinuirano ne razvija. Jedna od ovih tehnologija je i 3D tehnologija koja donosi korist za značajan broj sfera svakodnevnog života: medicina, strojarstvo, građevina, fizioterapija itd.

Bilo da je riječ o printanju tkiva, printanju mosta ili skeniranju i modeliranju kuće, 3D tehnologija pruža podršku i potencijal za još veći napredak u prethodno spomenutim, već ostvarenim inovacijama. Navedena tehnologija služi također i kao temelj mnogim daljnjim dostignućima u svijetu 'gaming' industrije koja raste konzistentno i neprestano se razvija. Isti oni fundamentalni principi koji su zaslužni za ideju magnetske rezonance, uvelike su primjenjivi za nove grafičke izazove s kojima se industrija videoigara konstantno susreće. No, tu ne staje spektar mogućnosti koje 3D tehnologija donosi sa sobom, a neki od njih koji su bitni za domenu videoigara bit će obrađeni u nastavku ovog rada.

Povijest, sadašnjost i budućnost 3D-a

Još od 70-tih godina prošlog stoljeća, krenula su istraživanja u svrhu razvoja tehnologije koja su se pokazala kao osnova za većinu današnjih grafika. Pierre

Bezier svoj je utisak ostavio radom na parametarskim krivuljama i površinama koje su se pokazale kao prethodnici danas popularne 'ray-tracing' tehnologije. Još neka od značajnijih postignuća su Phong i Gouraud ('key frame-based' animacija), J. Blinn (mapiranje tekstura), razvoj Photoshop-a i pojava OpenGL tehnologije tijekom 90-ih godina 20. stoljeća. [1]

U današnje vrijeme se upotreba 3D-a masovno 'prelila' u nekoliko razdvojenih upotreba u zabavnoj i gaming industriji, što u smislu izmjenjene realnosti (engl. *Augmented Reality – AR*), kao što su primjerice filteri na kamerama mnogih aplikacija (Snapchat, Facebook, Instagram, Pokemon GO), što u smislu virutalne realnosti/stvarnosti (engl. *Virtual Reality – VR*). Danas su već dostupne pojave kao što su virtualna terapija, virtualne šetnje, 3D videoigre, poučavanje/prezentiranje korištenjem 3D modela i vizualizacija za studente medicine, strojarstva, građevine itd.

Slika br. 1: 3D model Fakulteta Organizacije i Informatike dostupan za virutalnu šetnju na platformi Sketchfab

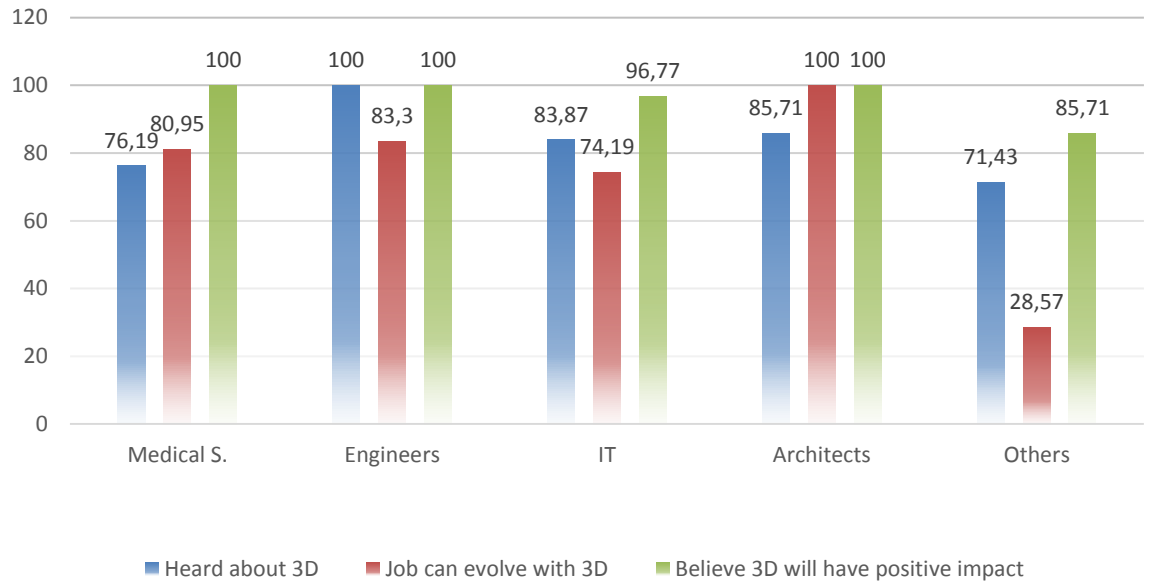
Danas već postoje tehnologije koje, uz gore navedene inovacije, pružaju mogućnost printanja organa i krvožilnog sustava, 3D printanja auto dijelova, pa čak i očuvanja potpunih građevina skeniranjem, a potom i sastavljanjem u 3D model. Osim primjene u spomenutim industrijama, 3D je daleko najviše iskorišten u filmskoj industriji kao osnova za popularan „CGI" (eng. *Computer-generated imagery"*) [2], odnosno računalno-generirani prikaz. Još od prvog prikaza u filmovima tipa Transformers ili Avatar, CGI se pokazao kao pun pogodak i glasno odjeknuo kinematografskim domenama kao „next big thing". Današnja ulaganja filmografskih kuća i producenata prelaze stotine milijuna dolara po filmu kako bi CGI korišten u njihovom filmu bio korak dalje u svijetu

realistične animacije. Pretpostavke postoje kako bi se uz buduću masovniju primjenu 3D tehnologije, čovječanstvo moglo riješilo nekoliko problema koji kotiraju visoko po razini prioriteta. Plastični otpad zamijenio bi se posebnim termalnim i razgradivim materijalima za printanje domova, dijelovi automobila i bilo kakvo materijaliziranje vršilo bi se pomoću 3D printera itd. Ovakvim potezima utjecalo bi se na kvalitetu života te beskućništvo (masovnijim printanjem domova pale bi cijene nekretninama te bi one postale dostupne većem broju stanovnika).

Posebne tehnike bio-printanja koje se trenutno razvijaju obećavaju iskorjenjivanje bolesti kao što su artritis, dok se printanjem organa i tkiva najučinkovitije moguće dolazi do transplantacije organa sa smanjenim rizikom odbacivanja budući da se isprintani organi i tkiva rade od matičnih stanica pacijenta. Ove i još mnogo drugih inovacija i ideja u planu su za budućnost, a svima je jedan nazivnik zajednički – 3D.

Primjena 3D tehnologije u modernom dobu

Prije svega, bitno je napomenuti o kakvim se brojkama radi kada je riječ o 3D tehnologiji i mlađoj populaciji. Prosječan broj osoba koje vjeruju u potencijalno dobre promjene koje 3D nosi sa sobom iznad je svih očekivanja te se može donijeti zaključak kako je današnja akademska mladež vrlo dobro upoznata s tehnološkim novitetima (graf 1).

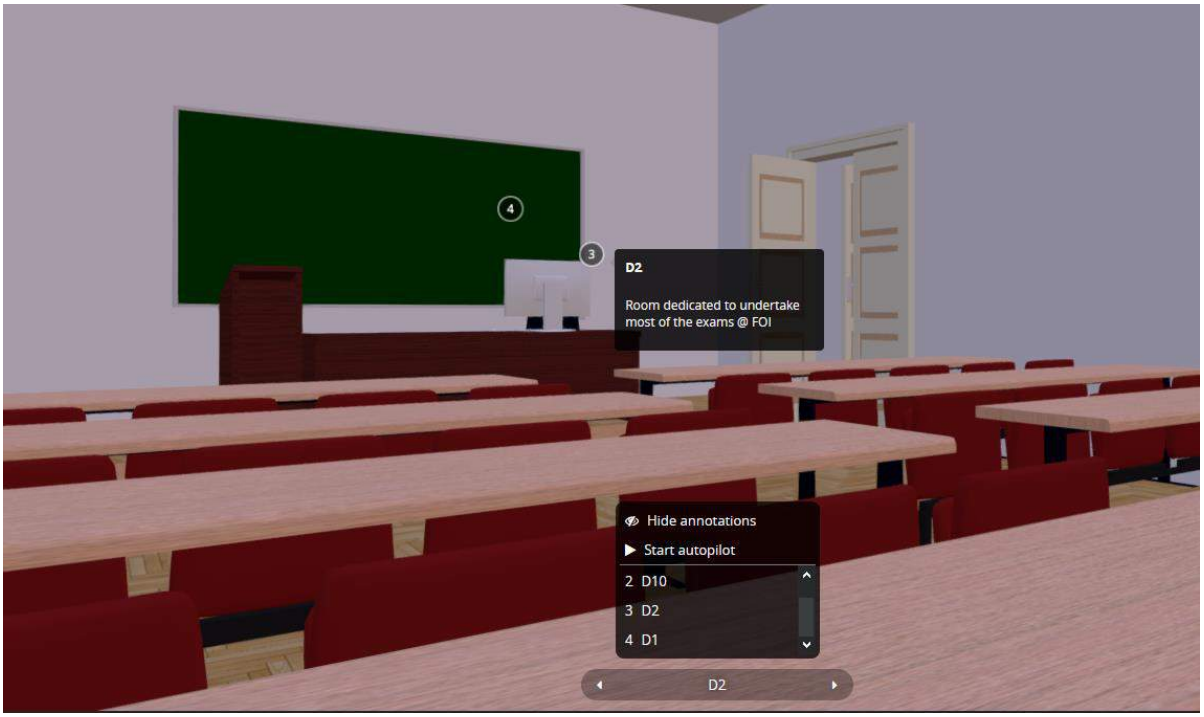


Graf br. 1: Rezultati ankete provedene na uzorku od 80 mladih osoba

Vrlo je važno razumjeti i domenu iz koje dolaze ispitanici pa tako ne čudi 100% informiranosti u struci inženjerstva i strojarstva, dok je pozitivno iznenađenje vezano za sudionike istraživanja s medicinskom pozadinom s

obzirom na to da je čak 76.19% mladih iz ove skupine upoznato s 3D tehnologijom, a još više njih vjeruje kako bi ista mogla donijeti pozitivne promjene u njihovoj matičnoj sferi.

U današnje vrijeme se primjena 3D-a već prakticira u mnogim područjima, no posljednjih nekoliko godina dostupne su određene 3D online platforme koje olakšavaju i približavaju uporabu i rad s 3D objektima i nude vrlo laku interakciju s istima što rezultira sve većim brojem korisnika ove tehnologije.



Slika br. 2: Dvorana 2 @ Fakultet Organizacije i Informatike, Sketchfab

Sketchfab [4], Googly Poly [5] i p3D.in samo su neke od platformi za vrlo laku interakciju s modelima kao što je model sa slike 2. Virtualna iskustva nikada nisu bila lakša i bliža jer ovakve tehnologije ne zahtijevaju opremu od nekoliko tisuća kuna već je potrebno imati samo internetski preglednik i malo dobre volje. Čak i uz minimalna ulaganja (HoloLens i sl., bilo koje prosječne 3D naočale) u rangu od 500 do 700 kuna, razina skoka u kvaliteti je i više nego iznenađujuća.

Svaka od navedenih platformi, ima i više nego dovoljno opcija pomoću kojih se diferencira, a naravno postoje i mnoge sličnosti između navedenih platformi. Najevidentnija razlika je u renderiranju i generalno bojama, no to ne znači kako je nešto drastično bolje od drugog jer na kraju krajeva svaka scena i stil renderiranja ne pašu jedno drugome. Važno je da za realne scene prikaz bude što realniji, dok je za neke druge 3D objekte ipak važnije da je uključeno više efekata kao što su sjaj, boja, lakoća rukovanja i slično.

Slika br. 3: Mogućnosti renderiranja scena - Sketchfab, p3D.in i Google Poly

Vrlo je interesantno korištenje značajki ovih platformi kao što su ugradnja na web stranicu (engl. *embed)* pri čemu se jednostavnim koracima omogućuje vrlo laka interakcija s objektima prikazanima na odabranoj sceni. Vrijedno je spominjanja i kako Sketchfab ima opciju povezivanja s VR naočalama koja je vrlo jednostavna te je moguće i odabrati način šetnje na sve 3 platforme kako bi se dobio osjećaj veličine i efekt realizma, no ipak je ultimativna realnost sadržana u pravom VR iskustvu uz VR naočale.

Važno je izdvojiti određenu tehnologiju koja izdiže Sketchfab iz mase sličnih tehnologija u 3D-web relaciji, a to je 3D oznaka (slika 2). Naime, zapisom u obliku trodimenzionalne oznake stječe se mogućnost „spremanja pogleda" odnosno pozicije u prostoru i na taj način ova platforma pruža bezbroj mogućnosti za svoje klijente iz područja obrazovanja (npr. oznake mjesta od interesa na modelu vijka za studente strojarstva), medicine (naglasak na specijalno područje i mogućnost sagledavanja tog područja iz više kutova) ili fizioterapeutskog područja (npr. mogućnost vježbanja hodanja u virtualnoj okolini te dosezanja dnevnih ciljeva – 2 km, 5 km itd. s interaktivnim objektima u zanimljivoj okolini).

Slika br. 4: Rehabilitacija uz odgovarajuću opremu – vrlo realan scenarij

Možda i najpoznatija primjena 3D-a danas vrti se na granici između videoigara i filmova u smislu animiranja 3D modela (engl. *rigging*), pa tako više nije rijetkost vidjeti potpuno „skenirane" osobe u igricama (Kevin Spacey – Call of Duty: Advanced Warfare; Keanu Reeves – Cyberpunk 2077.) ili robote koji su nastali skeniranje ljudskih „kostura" i sličnim tehnikama. Grafičke mogućnosti

današnjih grafičkih kartica (kombinirane serijski) omogućavaju podnošenje ogromnih napora u smislu renderiranja i najzahtjevnijih animacija i scena.

Slika br. 5: Keanu Reeves, u potpunosti 'digitalan' - Cyberpunk 2077 [3]

Zaključak

Vrlo je jasno kako budućnost nekolicine industrija direktno počiva na 3D tehnologiji dok veliki broj njih indirektno ima sve više koristi svakim daljnjim napretkom. Svaki taj napredak donosi nove iskorake i u svijet videoigara, u kojem zahtjevi vezani za grafiku konstantno rastu. Neupitno je kako će se većina 3D tehnologija koristiti i u daljnjem razvoju različitih platformi i usluga, kako novih igara tako i novih ideja vezanih za povećanje kvalitete ljudske svakodnevnice.

Reference

[1] A History of Computer Graphic Modeling - Digital School, 06-Mar-2014.
[2] Apodaca, A. A., Gritz, L., & Barzel, R. (2000). Advanced RenderMan: Creating CGI for motion pictures. Morgan Kaufmann.
[3] A. van de Velde, “CDPR: Keanu Reeves as Johnny Silverhand in Cyberpunk 2077 Was ‘A Natural Pick’ For The Character,” Wccftech, 21-Jun-2019.
[4] Johnson, J., Miller, D., & Palmer, K. L. (2018). Advancing 3D Digitization for Libraries, Museums, and Archives.
[5] Scopigno, R., Callieri, M., Dellepiane, M., Ponchio, F., & Potenziani, M. (2017). Delivering and using 3D models on the web: are we ready? Virtual Archaeology Review, 8(17), pp. 1-9.

Izrada trodimenzionalne platformske igre u programskom alatu Unity

Denis Huber, Danijel Radošević i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
denhuber@foi.hr, darados@foi.hr, mladen.monecki@foi.hr

Sažetak

U ovom radu opisuje se proces i izrada trodimenzionalne platformske igre u programskom alatu Unity. Cilj igre je istražiti nivo i pri tome sakupiti kocke koje omogućuju igraču otvaranje posebnih vrata kako bi prešao na iduću razinu. Na početku rada opisat će se ukratko osnovne karakteristike programskog alata Unity, a nakon toga proces izrade, dizajna i implementacije kreirane igre.

Ključne riječi: računalna igra, trodimenzionalni platformer, Unity, C#

Uvod

U ranim danima industrije računalnih igara developeri su morali raditi na vlastitim pogonima (eng. *engine*) kako bi razvili svoje igre. Pogon najčešće uključuje funkcionalnosti kao što su iscrtavanje grafike (2D ili 3D), detekciju kolizije, svojstva fizičkog svijeta, zvuk, skriptiranje, animiranje, umjetnu inteligenciju, mrežnu podršku i drugi [8]. Danas je situacija nešto drugačija. Netko tko želi razvijati vlastitu računalnu igru može koristiti neki od gotovih pogona igara, a danas ih na tržištu ima mnogo: Unity, Unreal Engine 4, Godot, Armory, CryEngine, Defold, Monogame i drugi [8]. Od navedenih progamskih alata, u svrhu izrade ovog rada korišten je Unity.

Unity je jedna od najpoznatijih razvojnih okolina za razvoj računalnih igara. Igre razvijene u ovom alatu mogu se izvesti na preko 25 platformi [6]. Alat ima moderni editor s moćnim modularnim alatima. Funkcionalnosti igre se pišu u obliku C# koda koji se zatim veže uz objekte i dodaju u scenu. Upravo takvim pristupom izrađen je glavni dio ovog rada. Osim za razvoj računalnih igara, Unity se može koristiti i za izradu filmova i animacija. Sposobnost uređivanja u stvarnom vremenu je nevjerojatno korisna karakteristika bez obzira na vrstu projekta koji se radi [5].

U ovom radu opisana je izrada trodimenzionalne platformske igre u programskom alatu Unity. Platformeri su jedan od prvih žanrova računalnih igara, a pojavili su se već početkom 80-ih godina prošlog stoljeća [2]. Kako je

tehnologija napredovala tako su i platformeri pratili taj razvoj. S vremenom su i oni napravili skok u treću dimenziju. Najpoznatiji predstavnici ovog žanra na osobnim računalima su *Mirror's Edge*, *Psychonauts*, *A Hat In Time* i drugi [7]. S vremenom se razvilo i niz podžanrova, no osnovne mehanike su kod svih vrsta iste.

Dizajn lika igrača i razina igre

U kreiranoj igri, igrač je u ulozi spretnog akrobata. Cilj igre je svladati mnoge prepreke nivoa uz upotrebu osnovnih mehanika igranja: skakanje, hvatanje za zid i kratak nalet (eng. *dash*). Model glavnog lika je inicijalno skiciran u programskom alatu Photoshop i kreiran u obliku 3D modela u besplatnom alatu za modeliranje, Blenderu [1].

Slika br. 1: Model igrača

Same razine igre su dizajnirane tako da podsjećaju na *puzzle* platformere u kojima igrač povremeno treba zastati, promotriti okolinu u kojoj se nalazi te odrediti najbolji put za postizanje željenog cilja.

Kao što je već rečeno, igraču su osim standardnog kretanja i trčanja na raspolaganju tri osnovne mehanike: skakanje, hvatanje za zid i kratak nalet. Igraču je dostupna i još jedna osnovna mehanika, a to je dupli skok kojeg je moguće ostvariti samo na određenim mjestima na nivou pomoću odgovarajućih ploča. Glavni nivo je kreiran tako da nema zacrtanu putanju kretanja već igrač može birati koji dio razine želi prvo istražiti. S obzirom na mogućnost da igrač padne s platformi, u igri je implementiran sustav

sigurnosnih točaka (eng. *checkpoints*). U slučaju pada, igrač se vraća na najbližu kontrolnu točku.

Slika br. 2: Izgled druge razine kreirane igre

Svi modeli u igri su kreirani u alatu Blender dok su animacije preuzete iz Unity online dućana (*Unity Asset Store*). Izgled igre i korištena paleta boja inspirirana je starim znanstveno-fantastičnim ilustracijama iz 70-ih.

Implementacija kamere gledanja

Kamera je implementirana u obliku tri hijerarhijski poredana objekta: korijena, pivota i Unity kamere. Korijenu je dijete pivot dok je pivotu dijete kamera koja je za određenu vrijednost udaljena od korijena. Smisao ovog pristupa jest da se svaki objekt rotira po jednoj osi dok djeca uvijek nasljeđuju rotaciju roditelja.

Jedan od glavnih problema koji je bilo potrebno riješiti jest preklapanje kamere sa zidovima. Igrač bi u svakom trenutku trebao moći vidjeti svog lika. Za rješavanje tog problema se koristi poseban oblik *Raycasting* metode zvan *Spherecast*. Ona od korijena do kamere projicira sferne oblike i prati da li je došlo do sudara s drugim objektima. Ako je došlo do sudara s nekim objektom, kamera se postupno pomiče na udaljenost kod koje je registriran sudar. U suprotnom, kamera slobodno lebdi na zadanoj udaljenosti. U istoj funkciji je implementirana i mogućnost zumiranja. Igra preko *Input* klase prati vrijednost srednje tipke miša i, ukoliko je uočeno kretanje, za danu vrijednost povećava ili smanjuje udaljenost kamere od korijena. Funkcija *Clamp* iz Unity klase *Mathf* pritom osigurava da ta udaljenost ostaje unutar određenog raspona.

```
float misScroll = Input.mouseScrollDelta.y;
if (misScroll != 0){
zadanaUdaljenostKamere -= misScroll * Time.deltaTime * 10;
zadanaUdaljenostKamere =
Mathf.Clamp(zadanaUdaljenostKamere, 3f, 5f);
}
```

Osnovno kretanje igrača

Kretanje igrača je implementirano u dva dijela:

1. Prazan objekt čijim se položajem upravlja putem *CharacterController* komponente i *Kontroler* klase.
2. Model osobe koji je podređen prvom objektu i animira se s obzirom na ulazne parametre igrača.

Većina koda se odnosi na manipulaciju jednog specifičnog vektora i primjenom rezultata tih operacija na pozivajući objekt. Pri tome treba napomenuti da je redoslijed izvođenja tih operacija izuzetno bitan radi postizanja željene interakcije. Igrač se pomiče kroz *Move* metodu *CharacterController* komponente, kroz nju se ujedno rješavaju sve kolizije s drugim tijelima u igri.

Primarna stvar je određivanje je li igrač prizemljen ili je u zraku. Korištenjem *Spherecast* metode detektira se kolizija igračevih nogu i površine. Ako je igrač prizemljen tada može vršiti osnovne mehanike igranja, u suprotnom mu se onemogućava daljnje skakanje. Samo skakanje je implementirano tako da se kod pritiska tipke za skok, Y vrijednost vektora kretanja postavi na zadanu silu skoka. Lik igrača će tako skočiti u vis dok ga gravitacija postupno usporava. Kada lik dosegne najvišu točku skoka i krene padati, na njega se primjenjuje sve veća sila gravitacije kako bi se padanje ubrzalo. Iako je to nerealistično, povećava dinamiku kretanja igrača.

Igrač se uvijek treba kretati relativno s obzirom na kameru. Vektor kretnje igrača se računa tako da se direktni vektori kamere zbrajaju te se množe s vrijednostima *Inputa* kretanja igrača (strelica ili ASDW tipke). Time se dobiva osnovni smjer i sila kretanja. Rezultat se mora normalizirati kako bi brzina kretanja u svim smjerovima bila ista. Konačno, ako vektor kretnje nije nulvektor i igrač trenutačno ne visi na zidu, lik igrača se rotira u XZ smjeru vektora kretnje. Y komponenta se pri rotaciji ignorira kako bi igrač ostao paralelan s podom.

Posebne mehanike igrača

Kratki nalet je jedna od osnovnih igraču uvijek dostupnih vještina. Implementacija ove mehanike temelji se na brojaču. Kada igrač aktivira funkciju naleta, brojač se smanjuje sa svakim pozivom za vrijednost

Time.deltatime. Sve dok je brojač veći od nule, vektor kretnje je uvećan. Kada brojač istekne, nalet je završen.

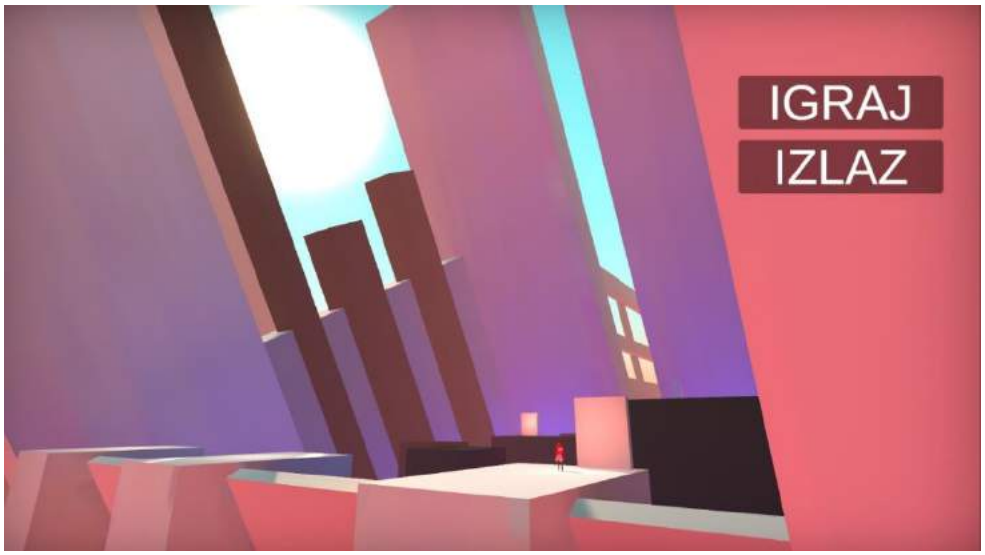
Još jedna dostupna vještina je hvatanje za zid. Ova mehanika se aktivira u trenutku kada igrač u zraku naleti na zid dok je tipka za hvatanje aktivirana. Hvatanje za zid funkcionira tako da: što se igrač u trenutku lijepljenja brže kreće, to će više kliziti po okomitoj površini. Moguće je stoga da lik slučajno i sleti sa zida. U tom slučaju se funkcija odmah prekida i lik igrača počinje padati. Ako je igrač i dalje uz zid, vektor lijepljenja se svakim novim pozivom smanjuje za dio svog suprotnog vektora. Kada postane dovoljno malen, izjednačava se sa nulvektorom i lik miruje. Klizanje se izvodi tako da se dohvaća normala danog zida i projicira se vektor lijepljenja na njenu ravninu. Time se osigurava da se lik uvijek kreće paralelno sa zidom. Stanje hvatanja za zid se prekida u tri slučaja: kada igrač prestane držati gumb za hvatanje, kada odluči skočiti ili kada na brojaču istekne vrijeme.

Konačno, vještina koju igrač može povremeno koristiti je dupli skok (eng. *boost*). Sustav na kojem funkcionira je sličan naletu, samo se ovdje umjesto vremena mjeri brzina. Ako je igrač već u skoku i ponovno aktivira skok, lik će se naglo katapultirati u zrak te postepeno gubiti brzinu dok ne počne padati. Tek tada igrač napušta stanje duplog skoka. Igrač pritom gubi dupli skok i mora otići do nove kontrolne ploče da ga ponovno aktivira.

Učitavanje razina i izvršavanje cilja razine

Prije početka same igre se učitava osnovni izbornik putem kojeg može pokrenuti igru ili izaći iz aplikacije. Taj izbornik je implementiran preko Unity *Canvas* objekta. Prelazak iz scene na scenu se vrši putem okidača, preko poziva klase *SceneManager* koja dohvaća indeks trenutno aktivne scene i povećava ga za jedan. Igra sadrži tri scene: izbornik te prvu i drugu razinu igre. Nakon završetka drugog nivoa, igra se vraća na početni izbornik.

Cilj igre je sakupiti kocke koje se nalaze na nivou kako bi se otključala vrata za izlazak. Ova funkcionalnost se sastoji od dva dijela. Prva je klasa *Sakupljac* koja prati broj sakupljenih kocki na nivou te vraća poruke igraču o trenutnom stanju sakupljenih kocki. Nakon što se sakupe sve kocke na nivou, deaktivira se prepreka koja skriva izlazna vrata iz nivoa. Drugi dio sustava je klasa *Kocka* koja se veže za okidač kocke koja se želi pokupiti. Preko *OnTriggerEnter* funkcije prati se je li funkciju aktivirao objekt s oznakom *Player*. Ako je to slučaj, tada se ažurira broj sakupljenih kocaka te se instanca sakupljene kocke deaktivira.



Slika br. 3: Glavni izbornik igre

Zaključak

Unity je vrlo moćna razvojna okolina za razvoj računalnih igara te nudi brojne mogućnosti kako bi pojedinac mogao realizirati svoju ideju. U ovom radu kreiran je prototip trodimenzionalne platformske igre te su opisane neke specifične mehanike ovog žanra koje su implementirane u kreiranoj igri. Za realizaciju ovakvog projekta potrebna su znanja iz mnogih domena, kako tehničke prirode tako i umjetničke.

Reference

[1] Blender. (18. srpnja 2019.). Dostupno na: https://www.blender.org/
[2] K. T. Jensen, „Run, Jump, and Climb: The Complete History of Platform Games“. (17. srpnja 2019.). Dostupno na: https://www.geek.com/games/run-jump-and-climb-the-complete-history-of-platform-games-1748896/
[3] Top 12 Free Game Engines For Beginners & Experts Alike. (17. srpnja 2019.) Dostupno na: https://conceptartempire.com/free-game-engines/
[4] Unity. (18. srpnja 2019.). Dostupno na: https://unity.com/
[5] Unity for Film: Real-Time Filmmaking, Explained. (18. srpnja 2019.). Dostupno na: https://unity.com/solutions/film/real-time-filmmaking-explained
[6] Unity Public Relations. (18. srpnja 2019.). Dostupno na: https://unity3d.com/public-relations
[7] What are the best 3D Platformers on Steam? (18. srpnja 2019.). Dostupno na: https://www.slant.co/topics/6467/~3d-platformers-on-steam
[8] Wikipedia: Game Engine. (17. srpnja 2019.). Dostupno na: https://en.wikipedia.org/wiki/Game_engine

Izrada računalne igre u tri dimenzije sa zagonetkama

Hrvoje Hodak, Robert Kudelić i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
hhodak@foi.hr, rkudel@use.startmail.com, mlkoneck@foi.hr

Sažetak

U ovom radu opisan je proces izrade prototipa trodimenzionalne računalne igre u programskom alatu Unity. Kreirane su tri razine. Na svakoj razini je potrebno riješiti zagonetku kako bi se prešlo na iduću razinu. Sve zagonetke su povezane no opet specifične.

Ključne riječi: računalna igra, igra zagonetke, Unity, C# , programiranje

Uvod

Kreirana igra u sklopu ovog rada inspirirana je sličnim komercijalno uspješnim računalnim igrama iz žanra igara zagonetki (eng. *puzzle games*). Glavni predstavnici računalnih igara ove kategorije su *The Talost Principle* hrvatskog tima za razvoj računalnih igara *Croteam* [5], *Portal* i *Portal 2* američke kompanije *Valve* [3], *Limbo* [1], *Braid*, *World of Goo*, *Grim Fandango* i druge [4].

Korišteni alati za razvoj prototipa spomenute računalne igre su programski alati Unity i Visual Studio. Unity je moćan alat koji nudi podršku za razvoj 2D, 3D igara kao i igara virtualne stvarnosti. Također se koristi za vizualizaciju proizvoda, animiranje, izradu filmova i sl. [6]. Visual Studio je razvojno okruženje za izradu stolnih, mobilnih i web aplikacija i servisa [2].

Dizajn kreirane igre

Cilj kreirane igre je logičko rješavanje zagonetki. Svaka razina igre se temelji na jednom specifičnom logičkom problemu kojeg igrač mora savladati kako bi napredovao na iduću razinu. Nakon uspješnog savladavanja zagonetke, igrač na svakoj razini pronalazi ključ putem kojeg otključava glavna vrata koja ga vode na iduću razinu.

Mehanike kretanja su standardne za igre u trodimenzionalnom prostoru: kretanje igrača se vrši pritiskom na tipke A, S, D i W dok se igrač rotira koristeći miš kao ulaznu jedinicu. Osim mehanike kretanja, igrač ima još neke mogućnosti. Klikom lijeve tipke miša igrač može pokupiti i nositi interaktivne

predmete koje otpuštanjem lijeve tipke miša može ispustiti. Klikom na desnu tipku miša igrač može predmet odbaciti od sebe u prostor (u slučaju da ima „u rukama“ predmet). Osim toga, pritiskom na tipku „E“ igrač može pokupiti interaktivne objekte dok s tipkom „F“ otvara glavna vrata. Posebne tipke su tipka „H“ koja igraču daje određenu pomoć kako bi savladao određenu razinu te tipka „R“ s kojom igrač može resetirati trenutnu razinu na kojoj se nalazi na početni položaj u slučaju da je došao u neki neželjeni položaj na razini.

Prilikom pokretanja kreirane igre igrač prvo dolazi u glavni izbornik gdje mu se nude četiri opcije: igranje igre, postavke igre, pomoć i izlaz iz igre. U postavkama igre igrač može upravljati glasnoćom zvuka te postavljati određene postavke vezane uz vizualni prikaz igre. Tu su također igraču upute o kontrolama igre. Pomoć služi za davanje osnovnih informacija igraču kako igrati igru.

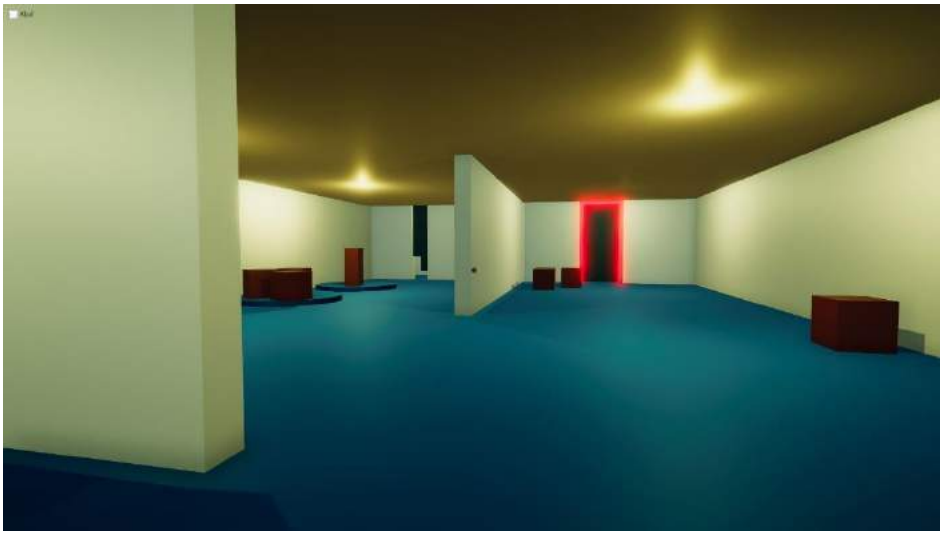
Slika br. 9: Postavke igre

Dizajn nivoa i zagonetki

Prva razina igrača stavlja u prostoriju u kojoj se nalazi pet kutija, dvoja vrata i jedan gumb koji igrač može stisnuti. Dok su vrata zatvorena ili u procesu zatvaranja, gumb svijetli crvenom bojom a dok su otvorena ili u procesu otvaranja, gumb svijetli zelenom bojom. Iza zatvorenih vrata se nalazi ključ za otvaranje vrata koja vode na iduću razinu. Otvaranjem vrata igrač može ući u prostoriju s ključem i pokupiti ga no neće stići izaći van. Ako se to desi, igrač mora resetirati stanje razine i krenuti iz početka. Kako bi savladao ovu zagonetku igrač se mora dosjetiti podmetnuti kutije ispod vrata kako se ona ne bi mogla zatvoriti do kraja. Tada može nesmetano pokupiti ključ te otključati glavna vrata i završiti nivo.

Slika br. 10: Prva razina igre

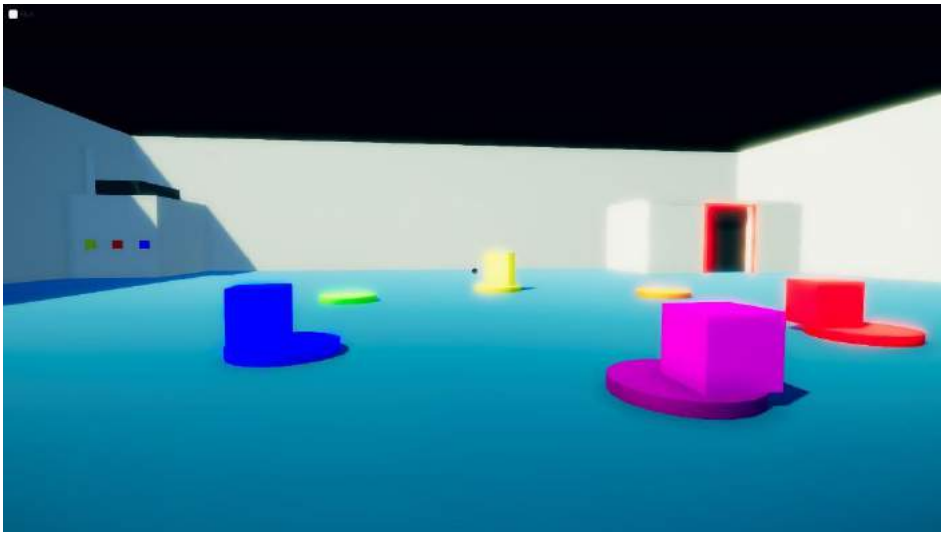
Na drugoj razini se u prostoriji nalazi deset kutija, dvije platforme i dvoja vrata. Specifičnost ove razine je ta što se vrata iza kojih se nalazi ključ otvaraju pomoću platformi koje reagiraju na masu. Još jedan detalj je da sve kutije nisu iste mase. Kako bi savladao ovaj problem, igrač mora rasporediti dostupne kutije na platforme u takvom omjeru da može proći u prostoriju do ključa.



Slika br. 11: Druga razina igre

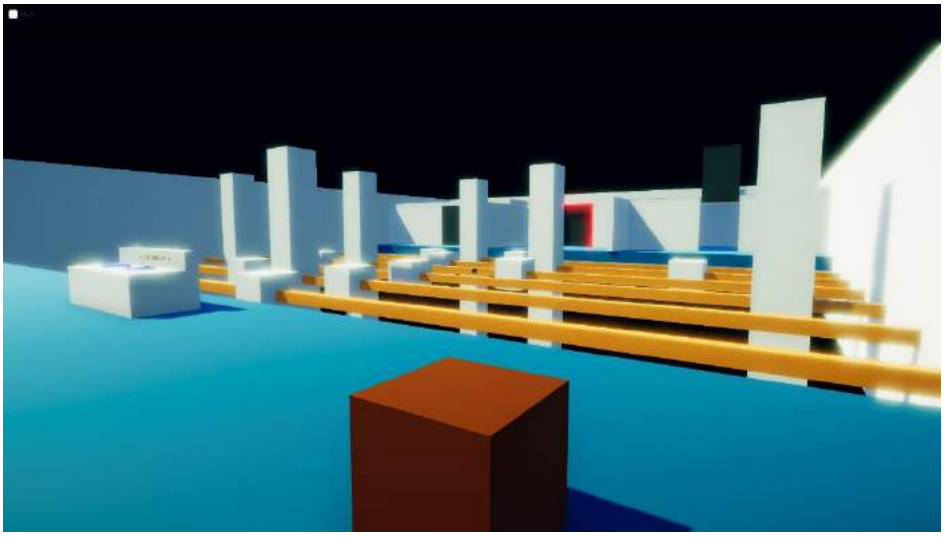
Na trećoj razini igrač se nalazi u velikoj prostoriji u kojoj se nalaze dvoja vrata, osam platformi u različitim bojama (žuta, narančasta, crvena, ljubičasta, plava, zelena i dvije crne) te tri gumba (žuti, crveni i plavi). Pritiskom na gumbe, igrač dobiva kutiju u boji gumba koji je pritisnuo. Kako bi savladao razinu, igrač mora na platforme u bojama postaviti kutiju te iste boje. No igrač nema kutiju koja je narančaste, zelene i ljubičaste boje. Kako bi dobio kutije tih boja, igrač mora na crne platforme postaviti dvije kutije a ono što će dobiti je boju koja je kombinacija boja dvije kutije na crnim platformama. Na taj način, primjerice,

zelenu kutiju može dobiti postavljanjem žute i plave kutije na crne platforme. Nakon što igrač riješi zagonetku s platformama, otvaraju se vrate prostorije s ključem i igrač može ići na sljedeću razinu.



Slika br. 12: Treća razina igre

Četvrta razina je najkompleksnija i sadrži mnoštvo interaktivnih elemenata: troja vrata, dvanaest kutija, dvanaest platformi, dva guma i deset pomičnih platformi. S dva interaktivna gumba igrač pomiče platforme koje se nalaze iznad provalije. Također, s kutijama i platformama koje se nalaze na podu može određene pomične platforme blokirati kako se one ne bi pomicale. Cilj ove razine je pomaknuti sve pomične platforme tako da kreiraju put preko provalije na drugu stranu. Jednom kada igrač pređe na drugu stranu, treba aktivirati platformu koja otključava prostoriju do ključa. Zatim je opet potrebno pomične platforme tako poredati kako bi igrač mogao pokupiti ključ. Konačno, potrebno je formirati novi niz pomičnih platformi kako bi se izašlo iz razine van.



Slika br. 13: Četvrta razina igre

Implementacija određenih mehanika

U nastavku će biti spomenute određene mehanike kreirane igre te kako su one implementirane u programskom alatu Unity. Pomoć prilikom pisanja skripti za funkcionalnost ove igre pronađena je na *YouTube* kanalima korisnika Holistic3d [9] i Blackeys [8] te na službenim stranicama Unity alata [7].

Sljedeći programski kod je kod klase *IgracKontrole.cs* koji je odgovoran za kretanje igrača:

```
void FixedUpdate() {
   pomakHorizontalno = Input.GetAxis("Horizontal");
   pomakVertikalno = Input.GetAxis("Vertical");
   if (Uzemljen() && Input.GetKeyDown(KeyCode.Space)) {
      rbIgrac.AddForce(Vector3.up * snagaSkoka, ForceMode.Impulse);}
   if (Input.GetKey("left ctrl")) brzina = 10.0f;
   else { brzina = 6.0f;}

   Vector3 pomak = new Vector3(pomakHorizontalno, 0.0f, pomakVertikalno);
   rbIgrac.transform.Translate(pomak * brzina * Time.deltaTime);
   if (Input.GetKeyDown("escape")){
      Cursor.lockState = CursorLockMode.None;
      SceneManager.LoadScene(0);}
   if (Input.GetKeyDown(KeyCode.R)){
      ResetirajLevel();}
   if (Input.GetKeyDown(KeyCode.H)){
      StartCoroutine(PrikaziPoruku());}
}
```

U nastavku je prikazan programski kod klase *IgracMis.cs* i služi za upravljanje kamerom:

```
void Update(){
   var promjenaMisa = new Vector2(Input.GetAxisRaw("Mouse X"),
                Input.GetAxisRaw("Mouse Y"));
   promjenaMisa = Vector2.Scale(promjenaMisa, new Vector2(osjetljivost *
             glatkoca, osjetljivost * glatkoca));
   glatkiPomak.x = Mathf.Lerp(glatkiPomak.x, promjenaMisa.x, 1f /
             glatkoca);
   glatkiPomak.y = Mathf.Lerp(glatkiPomak.y, promjenaMisa.y, 1f /
             glatkoca);
   pomakMisa += glatkiPomak;
   pomakMisa.y = Mathf.Clamp(pomakMisa.y, -90f, 90f);
   transform.localRotation = Quaternion.AngleAxis(-pomakMisa.y,
                Vector3.right);
   Igrac.transform.localRotation = Quaternion.AngleAxis(pomakMisa.x,
                Igrac.transform.up);
}
```

I ako zadnji dio, dan je primjer interakcije s okolinom, točnije kod koji pripada klasi *VrataLevel.cs* i služi za otvaranje glavnih vrata kada su potrebni uvjeti ispunjeni:

```
private void OnTriggerStay(Collider kolizija) {
   if (kolizija.gameObject.tag == "Igrac" && Input.GetKeyDown(KeyCode.F)
      && igrac.GetComponent<IgracKontrole>().imaKljuc.isOn &&
      desnoKriloOtvoreno == false && lijevoKriloOtvoreno == false){
         OtvoriVrata();}
}
```

Zaključak

S obzirom na strelovit razvoj računalne tehnologije, igraćih konzola i mobilnih uređaja, sve je veća potreba za računalnim igrama kao sredstvom zabave. Upravo zbog toga je industrija računalnih igara jedna od industrija s ponajvećim rastom iz godine u godinu. Unity kao programski alat za razvoj računalnih igara uvelike olakšava proces izrade i nudi dobru podršku za realizaciju željenih ideja. U ovom radu kreiran je prototip igre sa zagonetkama. Izazov u izradi ovog tipa igre je u tome da zagonetke budu po težini dobro balansirane kako igra ne bi bila prelagana ili preteška.

Reference

[1] LIMBO. (27. kolovoza 2019.). Dostupno na: https://playdead.com/games/limbo/

[2] Microsoft Visaul Studio. (27. kolovoza 2019.). Dostupno na: https://visualstudio.microsoft.com/

[3] Portal 2. (27. kolovoza 2019.). Dostupno na: http://www.thinkwithportals.com

[4] The 7 Best PC Puzzle Games of 2019. (27. kolovoza 2019.). Dostupno na: https://www.lifewire.com/best-pc-puzzle-games-4582059

[5] The Talos Principle. (27. kolovoza 2019.). Dostupno na: http://www.croteam.com/talosprinciple/

[6] Unity – Public Relations. (27. kolovoza 2019.). Dostupno na: https://unity3d.com/public-relations

[7] Unity Learn: Roll-a-ball. (2. veljače 2019.). Dostupno na: https://learn.unity.com/project/roll-a-ball-tutorial

[8] YouTube – Blackeys videos. (15. veljaće 2019.). Dostupno na: https://www.youtube.com/user/Brackeys/videos

[9] YouTube – Holistic3d videos. (31. ožujka 2019.). Dostupno na: https://www.youtube.com/channel/UCp_SOgsRYdLfIEWLjM62ZJg/videos

Korištenje stabla ponašanja za implementaciju umjetne inteligencije

Patrik Dolovski i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
patrik.dolovski@gmail.com, mladen.konecki@foi.hr

Sažetak

U ovom radu opisano je korištenje stabla ponašanja u svrhu implementacije umjetne inteligencije računalne igre iz prvog lica u programskom alatu Unreal Engine 4. U radu su opisani elementi stabla ponašanja kao osnova algoritma upravljanja glavnih neprijatelja igre.

Ključne riječi: računalna igra, igra iz prvog lica, Unreal Engine 4, umjetna inteligencija, stablo ponašanja, blueprint, C++, algoritmi

Uvod

Računalne igre su nerijetko pokretači tehnološkog razvoja računalne tehnologije a jednako tako i sredstvo za demonstraciju novih mogućnosti naprednih tehnologija. Najčešće se to odnosi na performanse grafičkih kartica jer se njihov potencijal u računalnim igrama može uvijek iskoristiti do krajnjih granica. Određeni naslovi računalnih igara su također bili dobri primjeri demonstracije naprednih tehnika kreiranja agenata koji su bili pogonjeni umjetnom inteligencijom. Primjeri takvih igara su *Tom Clancy's Splinter Cell: Blacklist*, *F.E.A.R.*, *The Last Of Us*, *Minecraft*, *Far Cry 2*, *Starcraft*, *Rocket League*, *Halo: Reach* i mnoge druge [1][3]. Odlike dobro implementirane umjetne inteligencije u igri je da neprijatelj uvijek reagira na drugačiji način, da se prilagođava novim situacijama te da ne čini jedno te iste greške više puta. Tako se igra *Clancy's Splinter Cell: Blacklist* pretvara u strateško nadmetanje poput šaha gdje je potrebno nadmudriti „pametne" neprijatelje koji štite neko područje [2]. Igra *The Last Of Us* je kasnila s izdavanjem 5 mjeseci upravo zbog želje da se razvije vrlo sofisticirana umjetna inteligencija u igri koja bi igraču pružila vrhunski doživljaj realističnog ponašanja [4].

Unreal Engine 4 je besplatna i jedna od najpoznatijih razvojnih okolina za razvoj računalnih igara [7]. Razvijen je od strane Epic Games-a a zadnja verzija, verzija 4, je izašla na tržište 2014. godine [8]. Unreal Engine 4 nudi podršku za izradu umjetne inteligencije kroz prethodno razvijene komponente percepcije i sustava stabla ponašanja koji su korišteni prilikom izrade ovog

rada [5][6]. Prednost Unreal Engine-a u odnosu na neke druge razvojne okoline je mogućnost vizualnog skriptiranja putem tzv. *Blueprint*-ova.

Umjetna inteligencija entiteta igre

Kreirana igra implementira klasične mehanike igre pucanja iz prvog lica. Umjetna inteligencija ponašanja protivnika igrača je implementirana na način da protivnici patroliraju razinama igre. Proces patroliranja se sastoji od toga da se protivnici kreću u nekom zadanom području te u tom procesu mogu detektirati igrača vizualnim i slušnim osjetilima. Kako bi se implementirala ova mehanika, potrebno je definirati *Blackboard* objekt koji ima svrhu memorije umjetne inteligencije. U taj objekt se pohranjuju varijable detektiranog igrača, njegova lokacija i podatci o patrolnim točkama. Za glavni dio ponašanja umjetne inteligencije protivnika zaduženo je stablo ponašanja.

Elementi stabla ponašanja

Elementi stabla ponašanja su podijeljeni u 4 skupine:

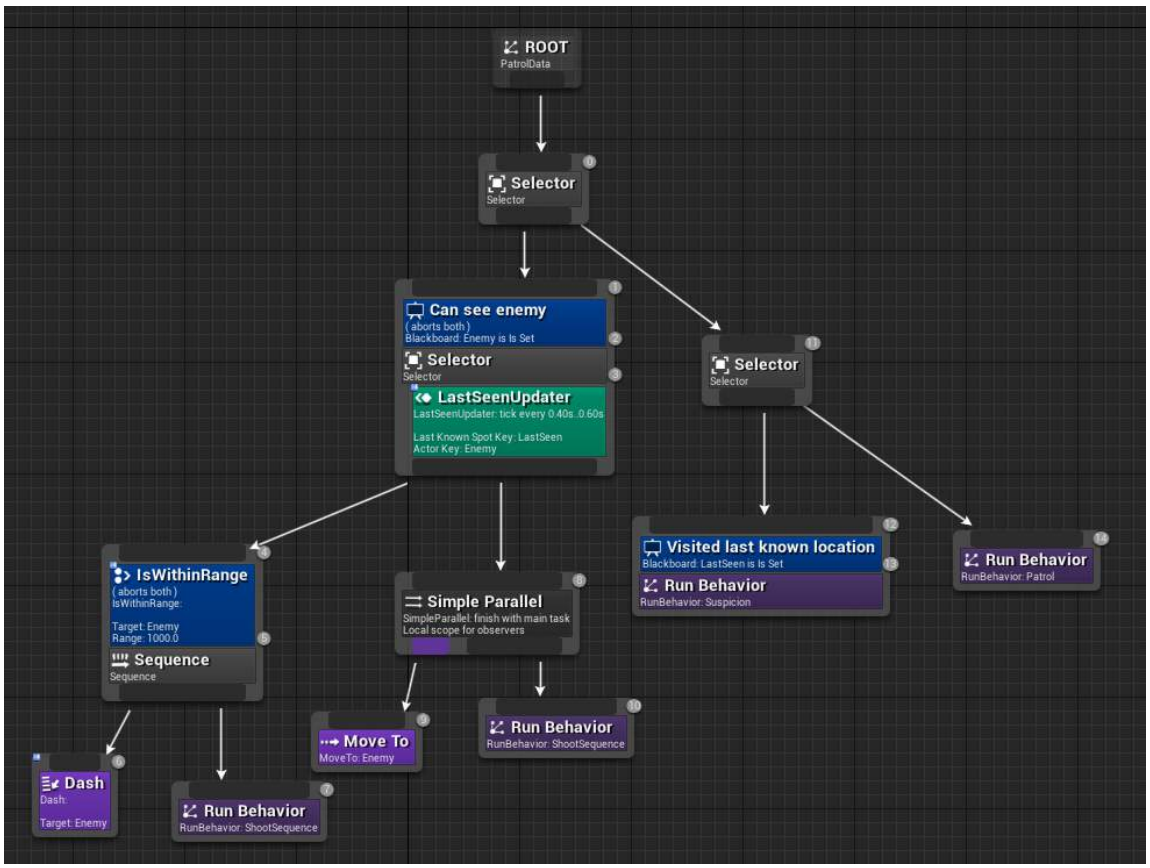
- *Composite* – ovi čvorovi definiraju korijen i pravila izvršavanja grane. Sadrže elemente tipa *selector*, *sequence* i *impleparallel*. *Selector* i *sequence* čvorovi omogućavaju izvršavanje čvorova stabla s lijeve strane prema desnoj s razlikom gdje *selector* staje nakon prvog uspješno izvršenog lista. *Simpleparallel* tip omogućava istovremeno izvršavanje dva lista od kojih je jedan glavni a drugi pomoćni.
- *Task* – to su listovi stabla koji izvršavaju određene poslove. To mogu biti predefinirani poslovi poput hodanja do određene lokacije, reprodukcija animacije ili poslovi vlastite izrade.
- *Decorator* – to je element koji se dodaje na postojeće *composite* čvorove kako bi se označili uvjeti izvođenja čvora.
- *Service* – to je element koji se kombinira s *composite* elementima te služi za komunikaciju podataka iz *Blackboard* elemenata s čvorovima.

Stablo ponašanja protivnika

Stablo ponašanja protivnika započinje elementom koji je korijen stabla te dolazi do prvog *selektora* koji stablo dijeli na dvije glavne grane. U lijevoj grani nalazi se selektor koji se veže uz *decorator* zadužen za provjeru stanja varijable *Enemy*. Također, sadrži *service* komponentu koja svakih pola sekunde osvježava vrijednost posljednje lokacije igrača. Tu su moguće dvije akcije: ako je igrač u određenom zadanom dometu pokreće se sekvenca *dash* i akcija pucanja po igraču ili kretanje prema igraču i pucanje.

U desnoj grani stabla *selector* je zadužen za odabir između dva lista: pokretanja *suspicion* način rada koji ovisi o zadnjoj poznatoj lokaciji igrača ili

pokretanja standardnog patroliranja između zadanih patrolnih točaka. Stablo ponašanja je prikazano na slici 1.



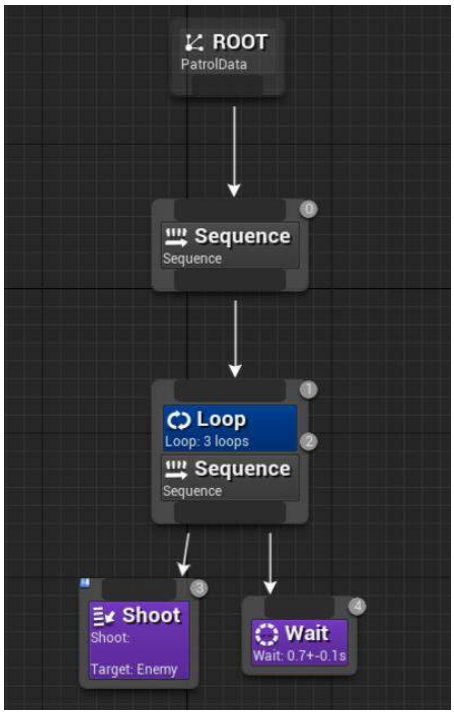
Slika br. 1: Stablo ponašanja protivnika igrača

ShootSequence **način rada**

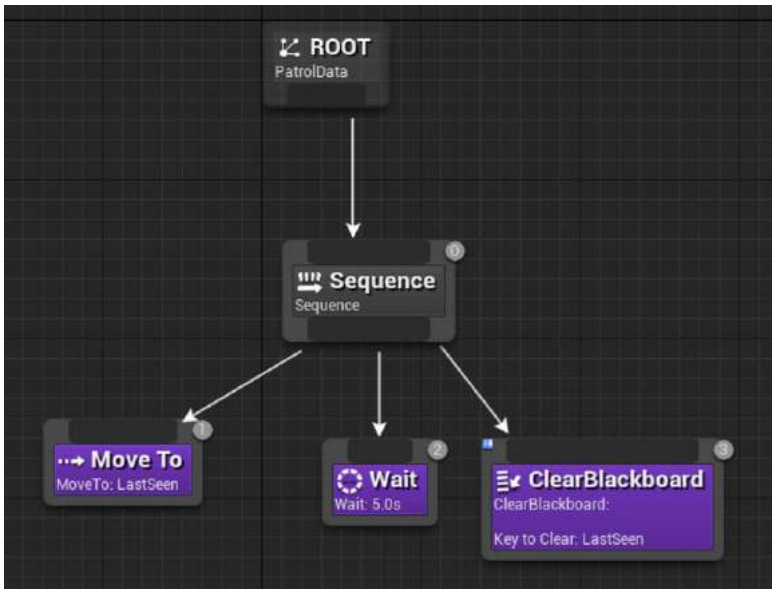
ShootSequence na slici 2 je podstablo zaduženo za pucanje po igraču. Sekvenca od 3 ponavljanja obavlja radnju pucanja i zatim čeka 0.7 sekundi.

Suspicion **način rada**

Suspicion način rada na slici 3 je podstablo zaduženo za primjenu sumnjičavog načina ponašanja. Likovi kontrolirani umjetnom inteligencijom kreću se prema posljednjoj poznatoj lokaciji igrača te tamo čekaju pet sekundi. Nakon završetka čekanja briše se posljednja poznata lokacija iz „Blackboard“ elementa.



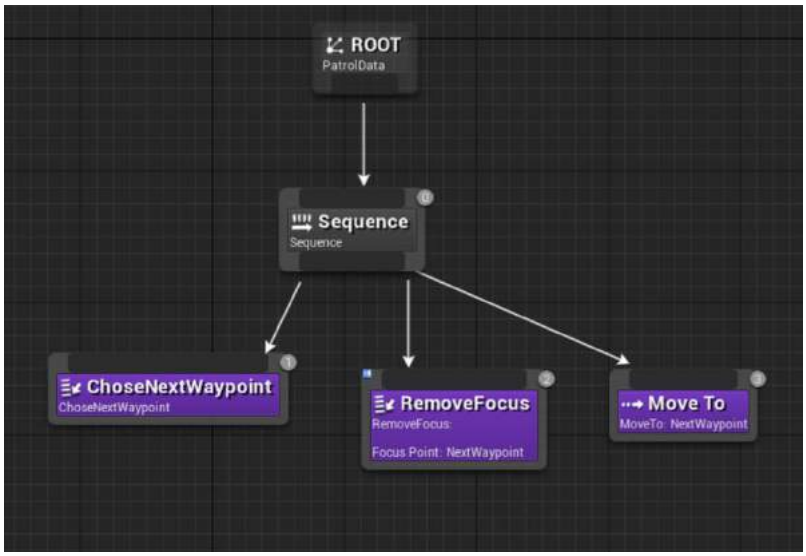
Slika br. 2: ShootSequence način rada



Slika br. 3: Suspicion način rada

Patrol **način rada**

Patrol način rada na slici 4 je podstablo zaduženo za patroliranje umjetne inteligencije kroz nivo. Sastoji se od sekvence koja pokreće zadatak biranja sljedeće točke patroliranja. Nakon tog izvršava se zadatak *RemoveFocus* gdje dolazi do uklanjanja stare fokusne točke i dodavanje nove. Naposljetku, umjetna inteligencija se kreće prema novoj točki patroliranja.



Slika br. 4: Patrol način rada

Zaključak

Unreal Engine 4 se zbog svoje fleksibilnosti i podrške koncepata percepcije i kreiranja stabala ponašanja pokazao kao adekvatan alat za implementaciju osnovnih koncepata umjetne inteligencije za akcijsku igru pucanja iz prvog lica. Grafički način definiranja uvelike olakšava realizaciju ovih koncepata. Također, grafički prikaz je vrlo elegantan jer se na pregledan način vizualizira način ponašanja.

Ovaj prototip je jednostavna demonstracija implementacije umjetne inteligencije na temelju koje bi se mogla razviti složenija stabla ponašanja koja bi omogućila sofisticiranije ponašanje neprijatelja. Također, igra se može nadograditi i na način da se uvede više različitih vrsta neprijatelja te da svaki neprijatelj ima nešto drugačije stablo ponašanja što bi doprinijelo raznovrsnosti ponašanja neprijatelja.

Reference

[1] 9 Games with the Best Artificial Intelligence. (24.kolovoza 2019.). Dostupno na: https://www.gameskinny.com/8zkeq/9-games-with-the-best-artificial-intelligence
[2] Game Analysis - Guard AI in Splinter Cell. (25. kolovoza 2019.). Dostupno na: http://www.davidtangness.com/journal/2014/11/6/game-analysis-guard-ai-in-splinter-cell-blacklist
[3] Top 5 Video Games That Have Made The Best Use Of AI. (24. kolovoza 2019.). Dostupno na: https://www.analyticsindiamag.com/top-5-video-games-that-have-made-the-best-use-of-ai/
[4] The Last of Us: AI so good you'll weep when you kill it. (25. kolovoza 2019.) Dostupno na: https://www.digitaltrends.com/gaming/we-talk-

with-naughty-dog-about-how-the-ai-in-the-last-of-us-will-make-you-feel-bad-about-yourself/
[5] Unreal Engine 4 Documentation: AI Perception. (25. kolovoza 2019.). Dostupno na: https://docs.unrealengine.com/en-US/Engine/ArtificialIntelligence/AIPerception/index.html
[6] Unreal Engine 4 Documentation: Behaviour trees overview. (25. kolovoza 2019.). Dostupno na: https://docs.unrealengine.com/en-US/Engine/ArtificialIntelligence/BehaviorTrees/BehaviorTreesOverview/index.html
[7] What is Unreal Engine 4. (25. kolovoza 2019.). Dostupno na: https://www.unrealengine.com/en-US/what-is-unreal-engine-4
[8] Wikipedia: Unreal Engine. (24. kolovoza 2019.). Dostupno na: https://en.wikipedia.org/wiki/Unreal_Engine

Izrada igre obrane tornjevima u programskom alatu Unity

Aldin Alagić i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
aalagic@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu se opisuje proces razvoja prototipa računalne igre tipa obrane tornjevima (eng. *tower defense*). Igra je razvijena u programskom alatu Unity. U radu se opisuju osnovne mehanike žanra i njihova implementacija uz popratne odsječke programskog koda. U kreiranoj igri implementirane su mehanike dolaska neprijatelja u valovima, izgradnja tornjeva s različitim svojstvima, mogućnosti nadogradnje tornjeva i njihova prodaja.

Ključne riječi: računalna igra, obrana tornjevima, Unity, C#, programiranje, algoritmi

Uvod

Igra obrane tornjevima je podžanr strateških računalnih igara u stvarnom vremenu. Temeljna obilježja su obrana teritorija od neprijateljskih napadača koji najčešće u valovima u određenim vremenskim intervalima idu prema glavnom cilju kojeg žele uništiti. Igrač se brani od napadača tako što postavlja obrambene zgrade oko ili na put kojim neprijatelji prolaze [7]. Ovaj žanr je 2007. godine postao na tržištu računalnih igara izuzetno popularan u obliku *Flash* igara koje su se igrale putem Internet preglednika. *Flash Element Tower Defense*, kreirana na temelju *Element TD* mape u *Warcraft III: Reign of Chaos* igri, je jedna od prvih predstavnika moderne verzije žanra obrane tornjevima [1]. Do danas, ovaj žanr je ostao izuzetno popularan a najbolji predstavnici ovog žanra su igre *Defense Grid*, *Plants vs. Zombies*, *Orcs Must Die*, *Kingdom Rush*, *Bloons TD 5*, *Fieldrunners 2* i druge [3].

Sa svakom razinom ili valom, neprijateljske jedinice su sve jače a glavni zadatak igrača je upravljati ekonomijom i izgradnjom adekvatnih obrambenih struktura na određenim lokacijama kako bi uspio savladati neprijatelje. Ovaj koncept igre se pokazao kao dobar koncept za učenje nekih osnovnih matematičkih obilježja kod učenika u dobi između 10 i 12 godina [2]. S obzirom na izazovnost zadataka, postoji i znanstveni rad koji govori o razvijenim algoritmima prilagodbe koji mogu onda uspješno savladati igre ovog tipa [4].

Za razvoj kreirane igre koristio se programski alat Unity [5]. Ovaj alat je primarno namijenjen za razvoj računalnih igara iako se danas u njemu izrađuju i proizvodi druge vrste (animacije, filmovi, reklame i sl.). Podržava izvoz na preko 25 platformi stoga je izuzetno dobar alat za razvoj igara ovog žanra s obzirom da je ovaj žanr, zbog svoje jednostavnosti mehanika igranja, pogodan za igranje i na osobnim računalima, i na konzolama, tabletima i pametnim telefonima.

Osnovni dizajn kreirane igre

Cilj kreirane igre je implementirati najtipičnije mehanike ovog žanra. Cilj igre je obraniti ulazna vrata od neprijatelja koji se kreću u valovima koji periodički nastupaju. Valovi se međusobno razlikuju po broju i vrsti neprijatelja koji dolaze. U kreiranoj igri postoje četiri vrste neprijatelja koji se razlikuju po brzini kretanja i količini životne energije. Ako neprijatelj uspije doći do kraja staze, tj. do ulaznih vrata, tada igrač gubi životne bodove. Ako životni bodovi igrača dođu do nule, tada je igrač izgubio igru. Kako bi se obranio, igrač gradi tornjeve koji se nalaze u okolici puta kojim neprijatelji prolaze. Tornjevi se kupuju osnovnom valutom igre, *Bitcoinom*. Igrač na početku razine dobiva određeni iznos novaca, a također uništavanjem neprijatelj dobiva dodatne jedinice. Igraču je na raspolaganju četiri vrste tornjeva koji imaju svoja specifična svojstva. Tornjeve je moguće, u zamjenu za novac, nadograditi kako bi ih se pojačala određena svojstva poput snage metaka, dometa i sl. Tornjevi se mogu i prodati te u slučaju prodaje igrač dobiva natrag dio uloženog novca.

Implementacija igre

Za razvoj kreirane igre korišteni su brojni modeli, materijali, slike, zvukovi i drugi elementi koji su preuzeti s *Unity Asset Storea* [6]. Što se tiče izrade programske logike, korišteni su materijali u obliku video poduka s YouTube kanala korisnika *Brackeys* [8]. Implementacija cijele igre organizirana je u 29 skripti koje upravljaju programskom logikom igre.

Slika br. 1: Glavni izbornik kreirane igre

Igra je kreirana u dvije scene: prva scena je izbornik koji se prikazuje prilikom pokretanja igre. U izborniku se nalaze tipične opcije postavka grafike i zvuka, opcije za igranje igre, odabir željenog nivoa i izlaz iz igre. Igraču je na raspolaganju klizač putem kojeg može točno odrediti težinu igre. Težina igre će utjecati na količinu životnih bodova koje igrač ima i na količinu novaca koja mu je raspoloživa na početku nivoa.

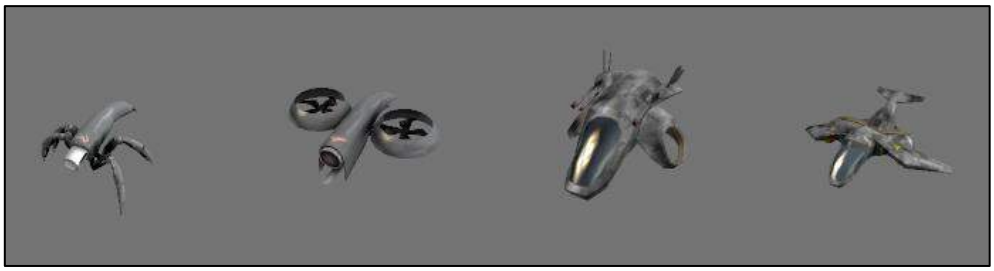
```
public void DifficultyChange (float value) {
    difficulty = (int)value;
    difficultyText.text = difficulties[difficulty].title;
    startMoneySlider.value = difficulties[difficulty].money / 100;
    enemyHealthSlider.value=difficulties[difficulty].enemyHealthFactor*10;
    timeBetweenWavesSlider.value = difficulties[difficulty].timeBetweenWaves;
}
```

Druga scena je scena u kojoj igrač igra igru. U ovoj sceni igrač može vidjeti nivo: putanju kojom se kreću neprijatelji te pozicije gdje može izgraditi tornjeve. Također, s desne strane sučelja može vidjeti informacije o tornjevima koje može izgraditi, njihova svojstva i cijenu. Na dnu korisničkog sučelja vidljive su informacije o kontrolama u igri.

Slika br. 2: Sučelje igre prilikom igranja

Neprijatelji

U igri se pojavljuju četiri vrste neprijatelja: paukovi, dronovi, tenkovi i letjelice. Modeli neprijateljskih jedinica su preuzeti iz paketa modela *SciFi Enemies and Vehicles – Popup Asylum* koji je moguće pronaći na *Unity Asset Storeu*. Neprijatelji se razlikuju po izgledu, brzini kretanja, načinu kretanja, količini novaca koje igrač dobije prilikom savladavanja i količini životnih bodova. Neprijatelji ne mogu napadati no potrebno im je definirati njihov put kretanja.



Slika br. 3: Vrste neprijatelja u igri

Dvije funkcije su glavne za interakciju s neprijateljima: funkcija *Attacked()* koja se izvršava u trenutku kada je neprijatelj napadnut i funkcija *Die()* koja se izvršava kada životni bodovi neprijatelja padnu ispod nule. Kada je neprijatelj napadnut, gubi životne bodove koji su vidljivi iznad modela neprijatelja. Kada neprijatelj umre, tada se smanjuje broj preostalih neprijatelja, igrač dobiva novce, izvodi se animacija eksplozije neprijatelja, neprijatelj se uništava i pušta se zvuk eksplozije.

```
public void Attacked(float amount){
    health -= amount;
    healthBar.fillAmount = health / startHealth;
    if (health <= 0 && !dead)  Die();
}

void Die() {
    dead = true;
    WaveSpawner.enemiesAlive--;
    PlayerStats.Money += value;
    PlayerStats.KillCount++;
    GameObject effectInstance =
    Instantiate(DeathEffect,transform.position,Quaternion.identity);
    Destroy(effectInstance, 4f);
    audioManager.PlayDelayed("Explosion", 0.2f);
    audioManager.Play("Enemy death")
}
```

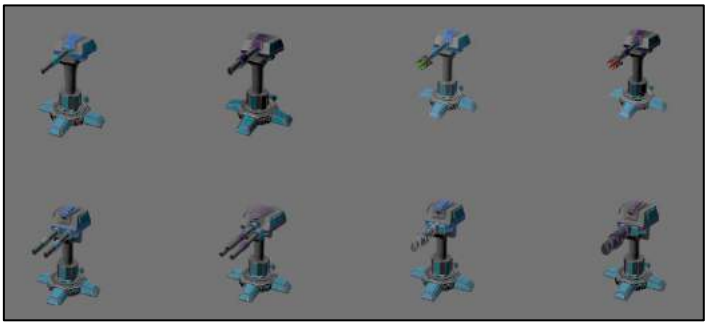
Unutar funkcije *Update()* realizira se kretanje neprijatelja po točkama puta. U slučaju da neprijatelj nije mrtav, dohvaća se smjer u kojem se treba kretati neprijatelj. Vektor smjera je dobiven razlikom pozicije mete i trenutne pozicije neprijatelja. Nakon toga slijedi pomjeranje neprijatelja pomoću funkcije *Translate()*. Ova funkcija pomjera neprijatelja na osnovu umnoška normaliziranog vektora smjera, brzine neprijatelja i delta vremena. Vektor smjera mora biti normaliziran kako ne bi utjecao na vrijednost umnoška. Posljednji argument funkcije *Space.World* označava da se neprijatelj kreće u odnosu na cijelo okruženje. U slučaju približavanja konačnoj poziciji točke prema kojoj ide, odabire se nova točka kretanja.

```
void Update() {
    if (!enemy.dead) {
      Vector3 direction = target.position - transform.position;
      transform.Translate(direction.normalized * enemy.speed *
    Time.deltaTime, Space.World);
      transform.forward = Vector3.Normalize(direction);
      If(Vector3.Distance(transform.position,target.position)<= (0.4f))
       GetNextWaypoint();
    }
    enemy.speed = enemy.startSpeed;
}
```

Tornjevi

U kreiranoj igri igraču su na raspolaganju četiri vrste tornjeva. Kao osnova za izradu tornjeva korišteni su modeli iz paketa *SciFi Simple Gun Turret Set - Tarko 3D*. Dva tornja imaju posebne osobine osim klasičnih parametara jačine, brzine i radijusa pucanja:

- Toranj s laserom – toranj koji kontinuirano usporava i oduzima životne bodove neprijateljima
- Toranj s raketom – toranj koji prilikom ispaljivanja rakete čini štetu svim jedinicama u određenom radijusu.



Slika br. 4: Vrste tornjeva u igri

Svaki toranj se može nadograditi kako bi se pojačale određene karakteristike tornja. Funkcija *LockOnTarget()* omogućava tornjevima da naciljaju označenog neprijatelja. Tornjevi u pravilu biraju najbližu metu koja im se nalazi u njihovom radijusu pucanja. Prvo se dohvaća smjer u kojem se toranj treba zarotirati. Pomoću funkcije *Lerp()* se rotacija interpolira između trenutne pozicije i željene pozicije.

```
void LockOnTarget() {
    Vector3 direction = target.position - transform.position;
    Quaternion lookRotation = Quaternion.LookRotation(direction);
    Vector3 rotation = Quaternion.Lerp(rotatingPart.rotation, lookRotation,
    Time.deltaTime * rotateSpeed).eulerAngles;
    rotatingPart.rotation = Quaternion.Euler(rotation.x, rotation.y, 0f);
```

```
}
```

Kada je neprijatelj pronađen i naciljan tada se okida funkcija *Shoot()* za pucanje po neprijatelju. Ova funkcija stvara instancu projektila na poziciji vrha cijevi tornja. Nakon toga se putem funkcije Seek() izvodi kretanje projektila od početne točke prema neprijatelju.

```
void Shoot() {
    if (useRocket) audioManager.Play("Rocket shot");
    else audioManager.Play("Regular shot");

    foreach (var firePoint in firePoints) {
      if (firePoint != null) {
        GameObject bulletGO = Instantiate(bulletPrefab,
        firePoint.position, firePoint.rotation);
        Bullet bullet = bulletGO.GetComponent<Bullet>();
        if (bullet != null) bullet.Seek(target);
      }
    }
}
```

Zaključak

Cilj ovog rada bio je razviti prototip jednostavne strateške igre obrane tornjevima koja implementirane osnovne mehanike ovog žanra u programskom alatu Unity. Unity se pokazao kao vrlo prikladan alat za razvoj igara ovog tipa jer nudi brojne funkcionalnosti koje olakšavaju izradu mehanika igre ovog tipa. Vrlo brzo se mogu pohvatati osnovni koncepti na kojima se onda može graditi naprednije znanje. Osim znanja programiranja, potrebno je znati i osnove vektorske algebre i analitičke geometrije. Razvoj računalne igre je generalno složen proces koji obuhvaća brojne elemente s kojima se developer susreće prilikom razvoja računalne igre.

Reference

[1] Flash Element TD. (5. rujna 2019.). Dostupno na: http://www.freewebarcade.com/game/flash-element-td/

[2] Hernández-Sabaté, A., Joanpere, M., Gorgorió, N., & Albarracín, L. (2015). Mathematics learning opportunities when playing a tower defense game. *International Journal of Serious Games*, 2(4), 57-71.

[3] J. Allen: The 8 Best Tower Defense Games of 2019. (5. rujna 2019.). Dostupno na: https://www.lifewire.com/best-tower-defense-games-4690505

[4] Rummell, P. A. (2011, July). Adaptive ai to play tower defense game. In *2011 16th International Conference on Computer Games (CGAMES)*, IEEE, 38-40.

[5] Unity. (5. rujna 2019.). Dostupno na: https://unity.com/

[6] Unity Asset Store. (5. rujna 2019.). Dostupno na: https://assetstore.unity.com
[7] Wikipedia: Tower defense. (5. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Tower_defense
[8] YouTube: How to make a Tower Defense Game playlist. (8. rujna 2019.). Dostupno na: https://www.youtube.com/playlist?list=PLPV2KyIb3jR4u5jX8za5iU1cqnQPmbzG0

Proceduralno generiranje razina 2D igre u programskom alatu Unity

Domagoj Ćurko, Danijel Radošević i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
dcurko@foi.hr, darados@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu je predstavljen algoritam za proceduralno kreiranje razina te njegova implementacija korištenjem programskog alata Unity. U praktičnom dijelu rada izrađen je prototip dvodimenzionalne računalne igre koja koristi algoritam za proceduralno generiranje razina.

Ključne riječi: računalna igra, dvodimenzionalna igra, dinamičko generiranje nivoa, Unity, C#, programiranje, algoritmi

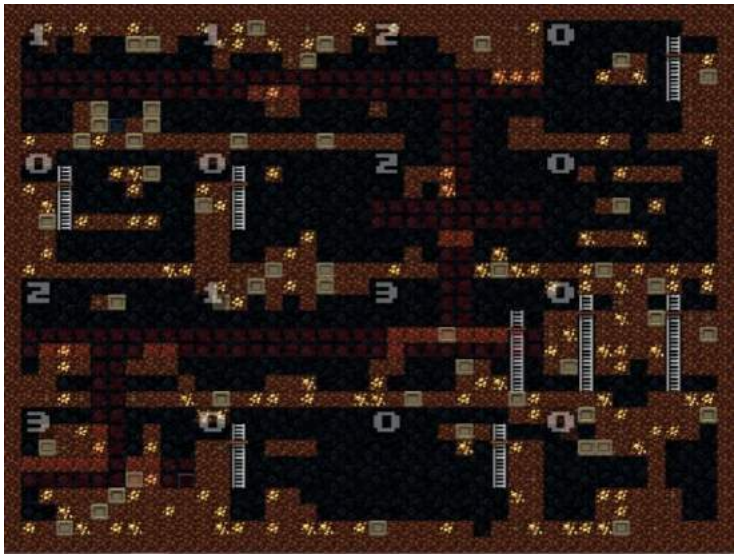
Uvod

Proceduralno generiranje je proces kada se podatci kreiraju putem izvođenja nekog algoritma a ne ručno [6]. U računalnim igrama, proceduralno generiranje se koristi kako bi se u igrama stvorila velika količina sadržaja. Primjerice, u serijalu igre *Civilization* karta svijeta se proceduralno generira. Na taj način igra postaje zanimljivija kod ponovnog igranja. Proceduralno generiranje mape ili svijeta koristi se i u mnogim drugim igrama: *Don't Starve*, *Spelunky*, *Stardew Valley* i dr. [5]. Ekstreman primjer je igra *No Man's Sky* gdje se proceduralno generira cijeli svemir s nevjerojatno velikim brojem planeta na kojima je opet proceduralno generiran životinjski svijet i vegetacija [1]. U sklopu ovog rada izrađen je prototip dvodimenzionalne platformske igre koja proceduralno generira razine igre. Kreirani prototip je razvijen u programskom alatu Unity [3]. Unity je programski alat koji je primarno namijenjen za razvoj računalnih igara. Igre se mogu izvesti na preko 25 platformi te alat omogućuje brži razvoj računalne igre jer alat predstavlja pogon igre (eng. *game engine*) te se time uvelike smanjuje količina posla [4].

Proceduralno generiranje razina

Kreirana igra se temelji na procesu proceduralnog generiranja nivoa u igri *Spelunky*. Svaka razina temelji se na dvodimenzionalnom polju soba, ili „mreži“. Dimenzije jednog nivoa su 4 sobe vertikalno puta 4 sobe horizontalno. Igrač nivou uvijek počinje u jednog od soba u gornjem redu. Izlaz iz nivoa se nalazi u jednoj od soba u donjem redu. Ono što je potrebno je osigurati da

postoji put od ulaza do izlaza nivoa. Kako bi se to osiguralo, stvara se takozvani „kritični put“. Njega čine sobe koje su direktni najkraći put od ulaza do izlaza nivoa. Na slici 1 je prikazan jedan generirani nivo spomenute igre [2].



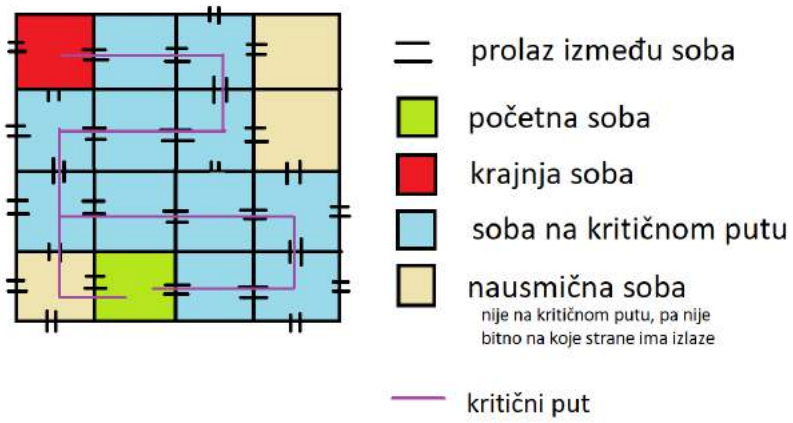
Slika br. 1: Generiranje razina u igri Spelunky

Svaka soba koja je na kritičnom putu mora biti određenog tipa, tj. mora imati izlaze na točno određenoj strani kako bi prolazak do izlaza bio moguć. Preostale sobe mogu biti bilo kojeg tipa jer se u njih može a ne mora ulaziti.

Ideja proceduralnog generiranja razine

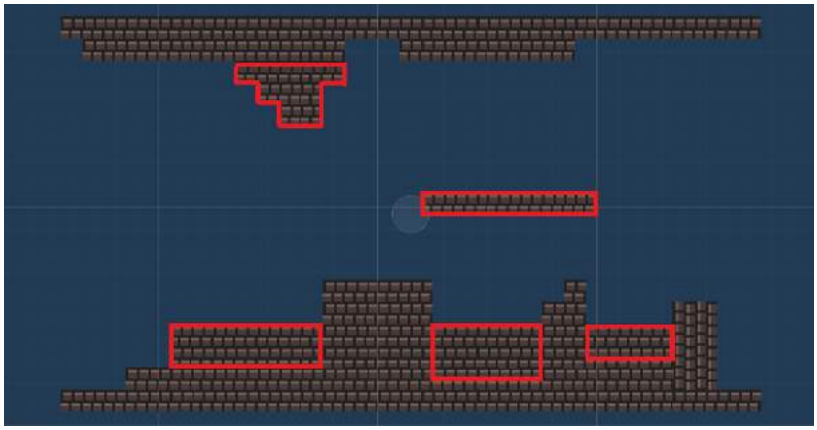
Cilj ovog rada bio je kreirati proceduralno generiranje razina koje bi bilo slično onome u igri *Spelunky*. U kreiranoj igri koriste se slični koncepti kao kod navedene igre: jedna razina sastojat će se od dvodimenzionalnog polja soba jednake veličine koje su unaprijed dizajnirane te imaju svoje propadajuće izlazne točke. Iako su sobe dizajnirane, i unutar same sobe postoje određene varijacije terena koje kod svakog generiranja sobe daju drugačiji oblik terena stoga i svaka soba ima svoje karakteristike proceduralnog generiranja. Također se koristi isti koncept stvaranja kritičnog puta no u kreiranoj igri taj put će biti od donjih soba prema gornjim sobama. Sobe koje nisu na kritičnom putu se također generiraju u potpunost nasumično s obzirom da nisu presudne za prohod nivoa. Ove restrikcije pomažu da se algoritam pojednostavi i da se dobiju funkcionalni nivoi koji imaju sve osnovne funkcionalnosti dizajniranog nivoa.

U kreiranoj igri postoje dvije osnovne vrste soba: sobe koje imaju izlaz lijevo i desno te sobe koje imaju izlaz na sve 4 strane. Ovaj odabir je nastao na temelju činjenice da sobe tog tipa imaju veliki potencijal za vertikalno spajanje što u nekim slučajevima može rezultirati time da na razini postoji i više mogućih putova do kraja nivoa kao to je to vidljivo na slici broj 2.



Slika br. 2: Primjer nasumično generirane razine

Kao što je već rečeno, unutar same sobe postoje određeni elementi nasumičnosti. To je osmišljeno da unutar sobe postoje određeni dijelovi koji se također nasumično stvaraju. Svaki element terena ima određenu vjerojatnost nastanka i na taj način se dodatno postiže varijabilnost izgleda razina.



Slika br. 3: Objekti koji su varijabilni

Implementacija

Za implementaciju kritičnog puta korištena je dvodimenzionalna matrica cjelobrojnog tipa. To polje se može interpretirati kao šahovsko polje u kojem svako polje sadrži jednu cjelobrojnu vrijednost. Prilikom inicijalizacije, svim poljima se dodjeljuje vrijednost 0. Ukoliko se generira soba koja ima izlaze horizontalno, tada se dodjeljuje polju vrijednost 1 a ako se generira polje koje ima izlaze na sve strane tada se polju dodjeljuje broj 2. Algoritam kreće od nasumičnog polja na najnižoj vertikalnoj razini te generira kritični put. Prilikom generiranja kritičnog puta završno polje mora biti u gornjem redu polja. Nakon generiranja kritičnog puta, sva ostala polja koja su ostala na vrijednosti 0 se određuju nasumično na vrijednosti 1 ili 2.

```
void OdrediOstaleSobe() {
```

```
        while (!kraj) {
            prijasnjaSoba = trenutnaSoba;
            bool uspjeh = false;
            while (!uspjeh) {
                int slucajniBroj = Random.Range(1, 6);
                if(slucajniBroj >= 1 && slucajniBroj <= 2)
                    uspjeh = ProbajPomakULijevuStranu();
                else if (slucajniBroj >= 3 && slucajniBroj <= 4) {
                    uspjeh = ProbajPomakUDesnuStranu();
                else if (slucajniBroj == 5)
                    uspjeh = ProbajPomakGore();}
            if(trenutnaSoba.y == dimenzijeRazine-1 &&((trenutnaSoba.x == 0
        && sobe[1,dimenzijeRazine-1]>0) || (trenutnaSoba.x == dimenzijeRazine
        - 1 && sobe[dimenzijeRazine - 2, dimenzijeRazine - 1] > 0))) {
            kraj = true;
            krajnjaSoba = trenutnaSoba;
            }
        }
}
```

Varijable *prijasnjaSoba* i *trenutnaSoba* su tipa *Vector2Int*, što znači da svaka od njih sadrži dvije cjelobrojne vrijednosti. One označuju poziciju polja na razini koja predstavljaju prijašnju sobu kritičnog puta i novu sobu u nizu. Pamtiti poziciju sobe koja je prethodno određena korisno je kako bi se kod pomaka prema gore, što znači da je nova soba na kritičnom putu iznad prethodne, vrijednost polja koje korespondira prethodnoj sobi mogla postaviti na 2, što znači da i ta soba mora imati izlaz prema gore. To je nužno jer, da bi se iz donje sobe prešlo u gornju, donja mora imati izlaz prema gore, a gornja izlaz prema dolje.

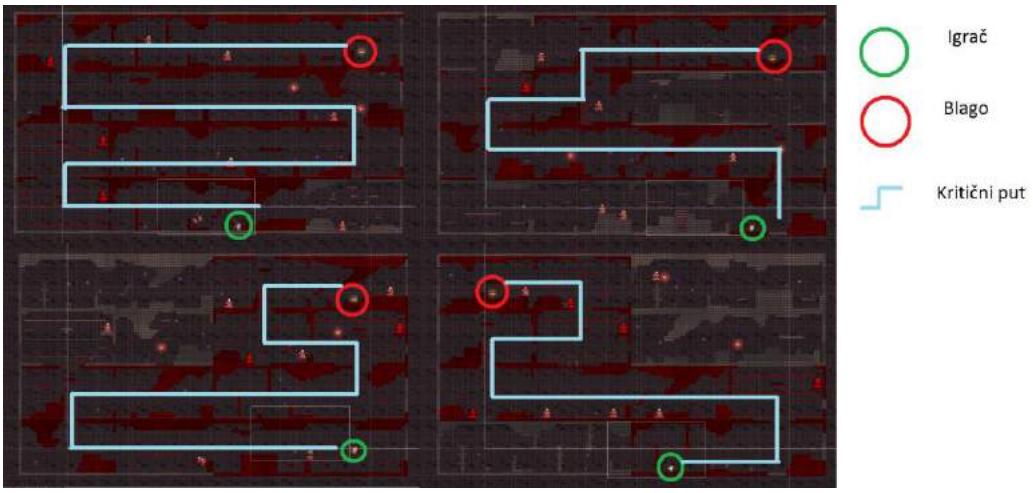
Sve dok nije kreiran kritični put u potpunosti, putem *while* petlje događaju se pomaci po razini tako da se na temelju nasumičnog broja odabire hoće li se pokušati trenutnu sobu (varijablu *trenutnaSoba*, koja će zapravo predstavljati sljedeću sobu na kritičnom putu) pomaknuti u lijevo, u desno ili prema gore. Ovo su funkcije koje provjeravaju je li pomak u određenom smjeru moguć, i ako je moguć, rade pomak u tom smjeru i vraćaju vrijednost *true*, u suprotnom vraćaju vrijednost *false* kako bi se ponovno pokušao pomak u neku drugu stranu.

```
bool ProbajPomakULijevuStranu(){
    if (trenutnaSoba.x > 0) {
        if(sobe[trenutnaSoba.x - 1, trenutnaSoba.y] == 0) {
            trenutnaSoba.x -= 1;
            sobe[trenutnaSoba.x, trenutnaSoba.y] = 1;
            return true;
        }
    }
    return false;
}
```

```
bool ProbajPomakUDesnuStranu(){
        if (trenutnaSoba.x < dimenzijeRazine - 1) {
            if (sobe[trenutnaSoba.x + 1, trenutnaSoba.y] == 0) {
                trenutnaSoba.x += 1;
                sobe[trenutnaSoba.x, trenutnaSoba.y] = 1;
                return true;
            }
    }
    return false;
}

bool ProbajPomakGore(){
        if (trenutnaSoba.y < dimenzijeRazine - 1){
            sobe[trenutnaSoba.x, trenutnaSoba.y] = 2;
            trenutnaSoba.y += 1;
            sobe[trenutnaSoba.x, trenutnaSoba.y] = 2;
            return true;
        }
        return false;
}
```

Nakon što su definirani indeksi soba na razini, na temelju tih vrijednosti moguće je stvoriti konkretne sobe.



Slika br. 4: Nekoliko nasumično generiranih razina

Zaključak

Kreirana igra je izrađena pomoću programskog alata Unity a za pisanje programskog koda korišten je alat Microsoft Visual Studio. Proceduralno generiranje razina ima veliki potencijal u stvaranju računalnih igara. Proceduralno generirane igre daju igraču osjećaj jedinstvenosti prilikom svakog ponovnog igranja igre. Opisani koncept proceduralnog generiranja razina se zasigurno može nadograditi s mnogih dodatnim mehanikama i zanimljivim konceptima. Primjerice, generiranje razine gdje je potrebno ući u

svaku sobu kako bi se prešao nivo bez obzira nalaze li se sve sobe na kritičnom putu. Ta mehanika bi se mogla postići kombinacijom skupljanja ključeva koji bi otključavali vrata koja se nalaze na kritičnom putu. Takav algoritam bi bio složeniji no i igra bi bila kompleksnija i zanimljivija.

Reference

[1] No Man's Sky. (11. rujna 2019.). Dostupno na: https://www.nomanssky.com/

[2] Spelunky Generator Lessons. (12. rujna 2019.). Dostupno na: http://tinysubversions.com/spelunkyGen/

[3] Unity. (11. rujna 2019.). Dostupno na: https://unity.com/

[4] Unity Scripting API. (12. rujna 2019.). Dostupno na: https://docs.unity3d.com/560/Documentation/ScriptReference/index.html

[5] Wikipedia: List of games using procedural generation. (11. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/List_of_games_using_procedural_generation

[6] Wikipedia: Procedural generation. (11. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Procedural_generation

Izrada akcijske računalne igre iz prvog lica u programskom alatu Unreal Engine 4

Josip Petanjek, Danijel Radošević i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
jpetanjek@foi.hr, danijel.radosevic@foi.hr, mladen.konecki@foi.hr

Sažetak

U ovom radu će se opisati će se opisati implementacija osnovnih mehanika akcijske računalne igre iz prvog lica u programskom alatu Unreal Engine 4. To se odnosi na implementaciju kamere, upravljanje i kretanje lika, korištenje osnovnih mehanika oružja, korisničko sučelje i umjetna inteligencija protivnika igrača.

Ključne riječi: računalna igra, igra iz prvog lica, Unreal Enginge 4, mehanike igranja, umjetna inteligencija, C++, blueprint, programiranje

Uvod

Računalne igre su postale jedan od najpopularnijih medija za zabavu. Industrija računalnih igara je u neprestanom porastu te se sve više i više igara neprestano razvija. U narednim godinama se očekuje prosječni porast industrije za čak 9% na godišnjoj bazi [6]. Glavna prednost računalnih igara u odnosu na druge umjetnosti je interaktivnost: korisnik kontrolira igru, donosi odluke i na taj način ulazi dublje u stvoreni svijet. Posebice u ovom žanru, igre iz prvog lica, gdje igrač doslovno poprima pogled glavnog lika igre.

Motivacija za ovaj rad je želja za realizacijom jednog ozbiljnog projekta od početka do kraja. Iako ovaj rad neće ići u velike dubine, pokriti će sve one osnovne elemente koje bi jedna akcijska igra iz prvog lica trebala imati, gledano iz perspektive realizacije funkcionalnosti igre.

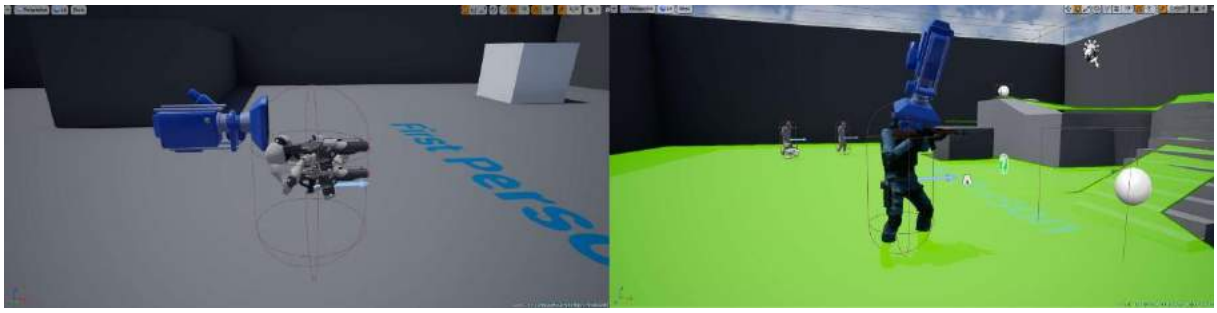
Žanr akcijskih igara iz prvog lica je jedan od najpopularnijih žanrova računalnih igara te na tržištu postoji velik broj kvalitetnih naslova. Ponajbolji predstavnici ovog žanra su Titanfall 2, Wolfenstein serijal, Doom, Call of Duty serijal, Overwatch, Counter-Strike serijal, Half-Life serijal i mnogi drugi [5]. Ove godine se primjerice održao veliki turnir u popularnog igri ovog tipa, Fortnite, te je nagradni fond bio 100 milijuna dolara [4].

Unreal Engine 4 je jedan od najpopularnijih besplatnih razvojnih okolina za razvoj računalnih igara s izuzetno dobrom podrškom za razvoj

trodimenzionalnih igara svakakvih vrsta, pa tako i akcijskih igara iz prvog lica [7]. Stoga je upravo ova razvojna okolina odabrana za realizaciju izrade prototipa akcijske igre iz prvog lica. Prednost Unreal Enginge-a je mogućnost izrade programske logike putem vizualnog skriptiranja tzv. *blueprint*-ovima [2].

Pogled iz prvog lica

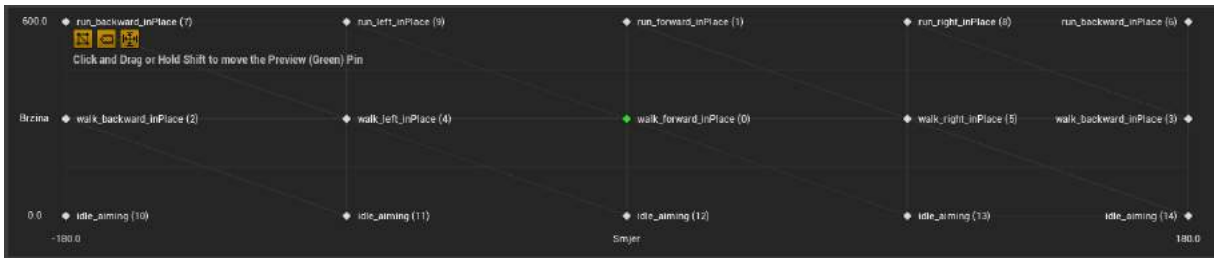
Opis izrade igra započinje s opisom izrade adekvatne kamere. Starije akcijske igre su koristile zaseban set animacija za pogled iz prvog lica te zaseban set za animacije ostalih likova u igri. To je dovodilo do situacije da kada je igrač pogledao prema dolje nije mogao vidjeti tijelo lika kojim upravlja. U novije vrijeme se koristi samo jedan set animacija te je model lika igrača prisutan u igri kao i svi ostali entiteti. Slika 1 daje usporedbu ove dvije implementacije.



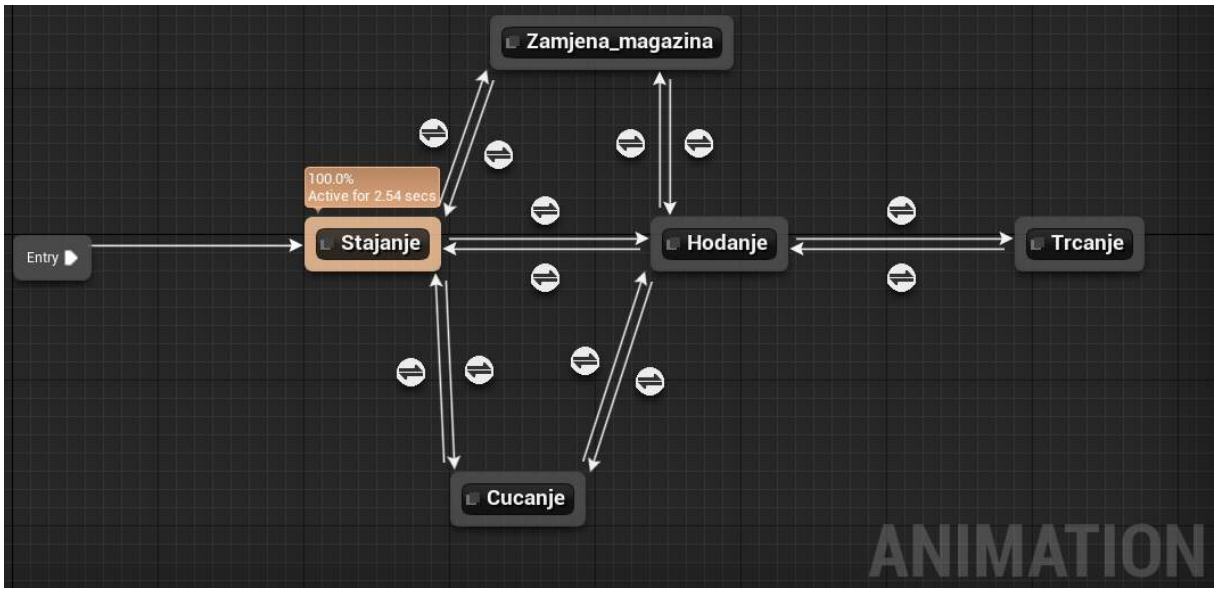
Slika br. 1: Usporedba dvije vrste implementacija kamere

Sustav kretanja i animiranja likova

Sustav za kretanje se sastoji od nekoliko osnovnih dijelova: animacije za kretanje lika, stanja animiranja te pravila za prijelaz iz stanja u stanje. Animacije lika obrađene su u miješanom prostoru, sama stanja bit će definirana putem stroja stanja a pravila prijelaza u nacrtu za animacije. Miješani prostor ovisi o smjeru i brzini kretanja. Brzina lika se mijenja s obzirom na vrstu kretanja (hodanje, trčanje, čučanje) a smjer je u rasponu od -180 do 180 stupnjeva [1]. Nakon definiranja koje se animacije žele miješati i pod kojim uvjetima, potrebno je odrediti stanja koja će određivati varijablu brzine.

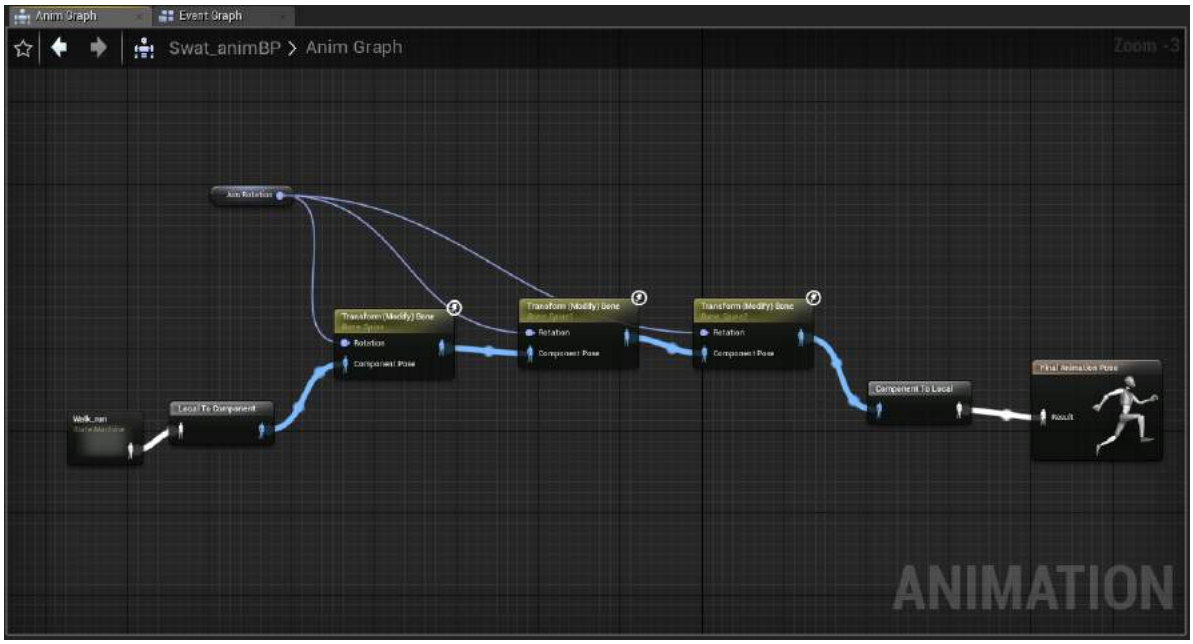


Slika br. 2: Miješani prostor za akciju hodanja



Slika br. 3: Stroj stanja igrača

Kako bi se postavile animacije na model lika igrača, potrebano je koristiti animacijski nacrt (eng. *Animation Blueprint*) u kojem se uspostavlja stroj stanja igrača te su definirane 3 kosti čija se pozicija mijenja s kretnjom rotacije za ciljanje kako bi se postigao efekt pogleda prema smjeru gledanja. Gotova animacije se tada stavlja u finalnu animacijsku pozu [3].

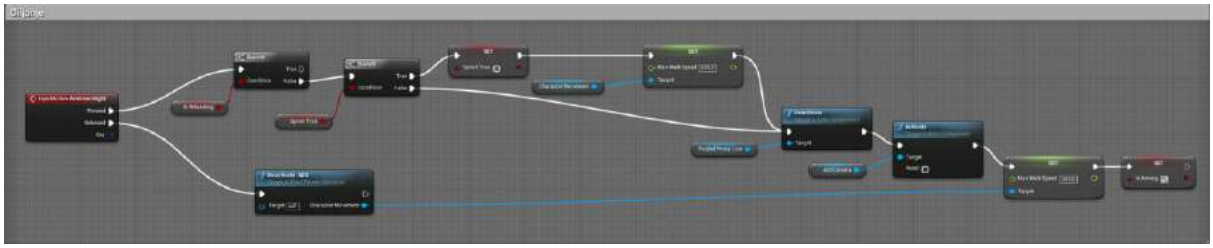


Slika br. 4: Animacijski nacrt igrača

Mehanike korištenja oružja

Mehanika ciljanja iz oružja aktivira se pritiskom na desnu tipku miša. Prilikom aktivacije potrebno je provjeriti da li igrač mijenja magazin ili trči. Ako su

uvjeti zadovoljeni, deaktivira se pogled iz prvog lica te se aktivira pogled ciljnika oružja, postavlja se adekvatna brzina kretanja te se označava to stanje varijablom.

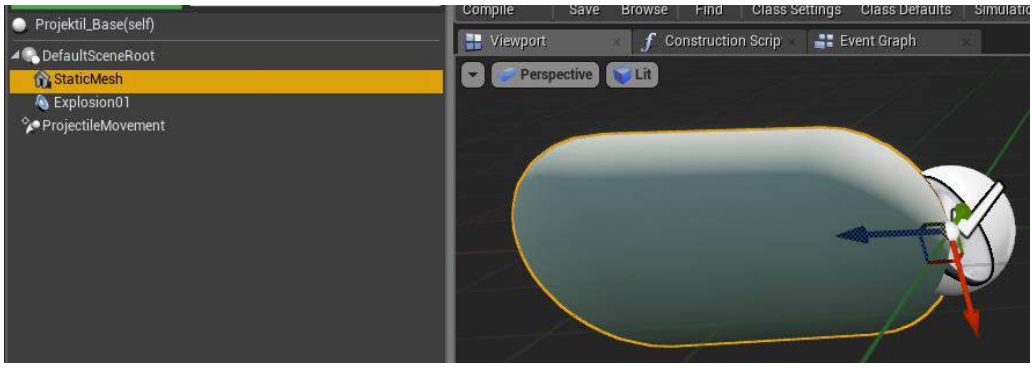


Slika br. 5.: Nacrt ciljanja iz oružja

Kao i u većini igara ovog tipa, kako bi korisnik pokupio oružje koje se nalazi na nivou, potrebno je samo doći do pozicije oružja. Nakon što se detektira kolizija između igrača i oružja, ono se dodaje igraču kao raspoloživo oružje te se dealocira model oružja na nivou.

Slika br. 6: Model sekundarnog oružja

Mehanika pucanja iz oružja je implementirana na način da se ispaljuju projektili iz puške koji su pod utjecajem gravitacije te imaju određenu brzinu. Pucati je moguće također u slučaju ako igrač ne mijenja magazin puške ili ne trči. Također je potrebno provjeriti ima li igrač potrebnu municiju za oružje koje želi koristiti. Pritiskom na lijevu tipku miša, stvara se projektil na vrhu puške koji se ispaljuje u prostor. Prilikom pucanja na vrhu puške je vidljiva eksplozija te se municija smanjuje za jedan. Prilikom kolizije metka s objektima na razini, također se na odredištu stvara vidljiva eksplozija kao povratna informacija o tome gdje je metak završio.



Slika br. 7: Sastav projektila

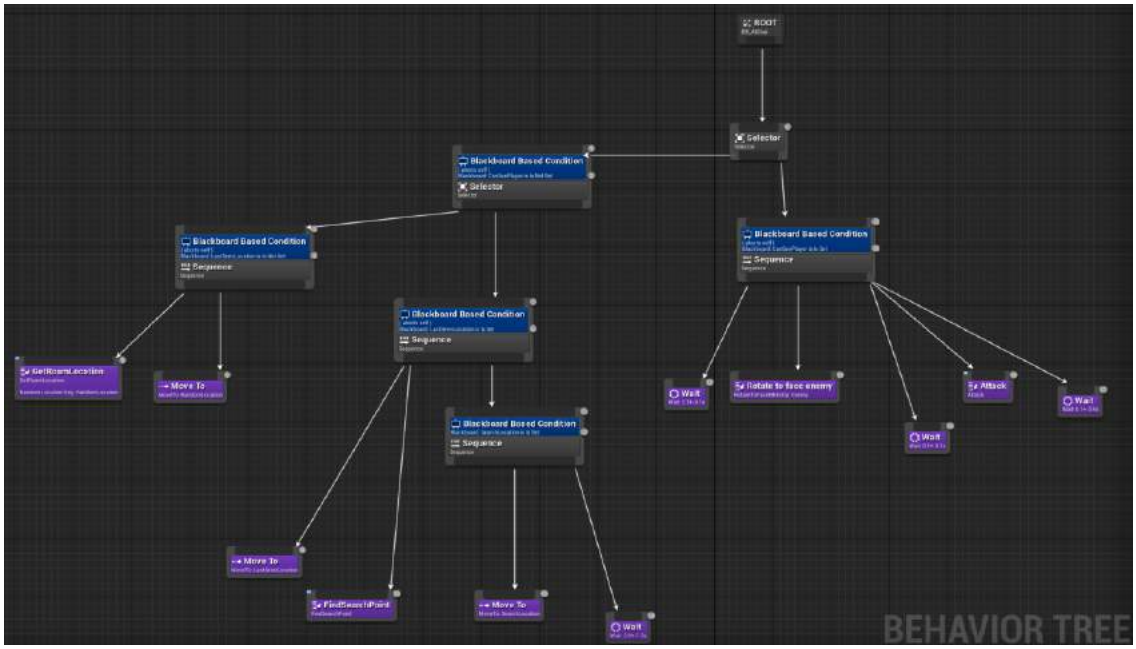
Umjetna inteligencija protivnika igrača

Umjetna inteligencija protivnika igrača je definirana pločom (eng. *Blackboard*) u koju se pohranjuju potrebni podatci a za samo ponašanje neprijatelja odgovorno je drvo odluka [8].

Sama ploča je samo set varijabli koje se koriste za postizanje funkcionalnosti umjetne inteligencije.

- Neprijatelj (engl. *Enemy*) je varijabla tipa objekt, koja sadrži podatke o igraču
- Može vidjeti neprijatelja (engl. *CanSeePlayer*) je Bool operator koji je zadužen za pamćenje stanja
- Zadnja viđena lokacija (engl. *LastSeenLocation*) je vektor koji je potreban za postizanje funkcionalnosti praćenja lika
- Lokacija za traženje (engl. *SearchLocation*) je vektor koji je potreban za pronalazak igrača
- Slučajna lokacija (engl. *RandomLocation*) je vektor koji se koristi za kretanje neprijatelja dok još nije uočio igrača

Logika se bazira na sljedećem konceptu: ako neprijatelj ne vidi igrača i nema podatke o njegovoj zadnjoj lokaciji, tada se kreće prema slučajnoj lokaciji na nivou. Kada neprijatelj opazi igrača tada se počne kretati prema njegovoj lokaciji i započne paljbu prema igraču. Ako igrač pobjegne iz vidnog kruga, tada se neprijatelj kreće do zadnje poznate lokacije gdje je igrač bio opažen. U slučaju da neprijatelj više ne zna gdje je igrač, vraća se ponovno u kretanje po nasumičnim točkama.



Slika br. 8: Drvo odluka

Zaključak

Unreal Engine 4 nudi dobru podršku za implementaciju programske logike akcijske igre iz prvog lica. Lako je raditi s kamerama koje se definiraju na željeni način. Također, putem nacrta je vizualnim skriptiranje vrlo lako kreirati programsku logiku osnovnih radnji. Nacrti su vrlo pregledni te je jednostavno s njima baratati.

Također, postoji jako dobra podrška za implementaciju umjetne inteligencije. Drvo odluka je odličan mehanizam kako bise se implementirala osnovna logika ponašanja neprijatelja. Kombinacijom koncepata koji su podržani, moguće je ostvariti i mnogo kompleksnije ponašanje neprijatelja što bi dovelo do „pametnijeg“ ponašanja i samim time bi igru učinilo zanimljivijom.

Reference

[1] Blend spaces. (20. srpnja 2019.). Dostupno na: https://docs.unrealengine.com/en-US/Engine/Animation/Blendspaces/index.html

[2] Blueprint visual scripting. (17. srpnja 2019.). Dostupno na: https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html

[3] Creating A First Person Shooter – Unreal Engine 4 Course. (28. srpnja 2019.). Dostupno na: https://www.youtube.com/playlist?list=PLL0cLF8gjBprG6487lxqSq-aEo6ZXLDLg

[4] Fortnite goes big on esports for 2019 with $100 million prize pool. (17. srpnja 2019.). Dostupno na: https://techcrunch.com/2019/02/22/fortnite-goes-big-on-esports-for-2019-with-100-million-prize-pool/

[5] The best FPS games on PC. (17. srpnja 2019.). Dostupno na: https://www.pcgamesn.com/15-best-pc-first-person-shooters

[6] The Global Games Market Will Generate $152.1 Billion in 2019 as the U.S. Overtakes China as the Biggest Market. (17. srpnja 2019.). Dsotupno na: https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/

[7] Unreal Engine 4. (17. srpnja 2019.). Dostupno na: https://www.unrealengine.com/en-US/

[8] Unreal Engine: Behavior Tree Overview. (21. srpnja 2019.). Dostupno na: https://docs.unrealengine.com/en-US/Engine/ArtificialIntelligence/BehaviorTrees/BehaviorTreesOverview/index.html

Izrada računalne igre u programskom alatu Construct 3

Luka Perić i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
lperic@foi.hr, mladen.konecki@foi.hr

Sažetak

U ovom radu opisuje se proces izrade jednostavnog prototipa igre u programskom alatu Construct 3. Kreirana igra je 2D platformer. U radu će se opisati pojedini segmenti implementacije mehanika ovog žanra u spomenutom alatu. Glavne mehanike igre su kretanje igrača, pucanje, sakupljanje interaktivnih objekata. U igri postoje tri vrste neprijatelja gdje svaki neprijatelj ima svoja specifična svojstva.

Ključne riječi: računalna igra, 2D platformer, Construct 3, vizualno skriptiranje, mehanike igre, algoritmi

Uvod

Construct 3 je jedan od brojnih besplatnih programskih alata za razvoj računalnih igara. Danas na tržištu postoji mnogo razvojnih okolina u svrhu razvoja računalnih igara: *Unity*, *Unreal Engine 4*, *Godot*, *Armory*, *CryEngine*, *Defold*, *Monogame*, *Corona* i dr. [3]. Prednost alata *Construct 3* je da se može izvoditi u pregledniku: to je programski alat za razvoj dvodimenzionalnih igara koji se temelji na *HTML5* tehnologiji, a razvila ga ja kompanija *Scirra Ltd.* [6]. Ciljana skupina korisnika ovoga alata su korisnici koji nisu primarno programeri jer omogućava brzi razvoj igara vizualnim skriptiranjem i kriptiranjem logike ponašanja. Alat ima dobru dokumentaciju, tečaj za početnike, video upute te je stoga vrlo prikladan za početnike [1].

Žanr platformskih igara je jedan od najstarijih žanrova računalnih igara. Spada u podžanr akcijskih igara. Tipične mehanike ovog žanra je kretanje igrača po platformama gdje pri tome mora izbjegavati razne prepreke [7]. Nivo je često neravan i sam kao takav predstavlja izazov za prolazak dok se onda na nivou pojavljuju i razni neprijatelji. Smatra se da je igra *Space Panic,* izdana 1980. godine, prvi platformer [4] dok je igra *Jumping Flash*, izdana 1995. godine, prvi pravi trodimenzionalni platformer [2]. I dan danas, žanr 2D platformera je popularan a glavni predstavnici su: *Hollow Knight*, *Ori and the Blind Forest*, *Super Meat Boy*, *Limbo* i druge [5].

Razvoj igre

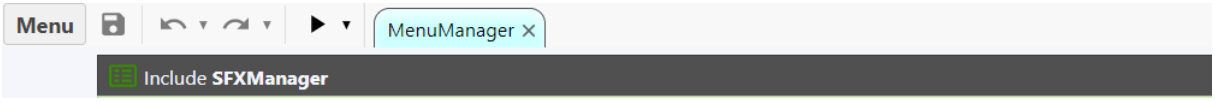
U razvijenoj igri kreirana je jedna razina u kojoj je cilj igraču skupiti što više zlata te poraziti neprijatelje koji mu stoje na putu kako bi ostvario željeni cilj. Savladavanje neprijatelja također igraču donosi dodatne bodove. Na taj način se igrača želi ohrabriti kako bi savladao neprijatelje i sakupljao zlatnike u prolasku nivoa. Igraču se dostupne mehanike kretanja, skakanja i ispaljivanja strijele. Na nivou se nalaze razne prepreke u obliku šiljaka koji mogu oduzeti zdravlje igraču. Također, na nivou se nalaze neprijatelji koji se kreću po zemlji a također postoje i neprijatelji koji se kreću po zraku. Na kraju nivoa, nalazi se glavni neprijatelj koji ima posebnu sposobnost stvaranja neprijatelja koji se kreću po tlu.

Prije samog igranja igre, igrač ima nekoliko opcija vezano uz zvuk i grafički prikaz igre. Također može birati jedan od dva ponuđena načina kretanja: tipkama *asdw* ili strelicama.

Konkretno, implementacija igre u programskom alatu Construct 3 izvedena je u tri scene. Svaka scena se veže uz *Event Sheet* objekt, odnosno listu događaja u kojoj se kreira logika igre. Svaki list događaja može imati referencu na više drugih događaja.

Slika br. 1: Veza scene i liste događaja

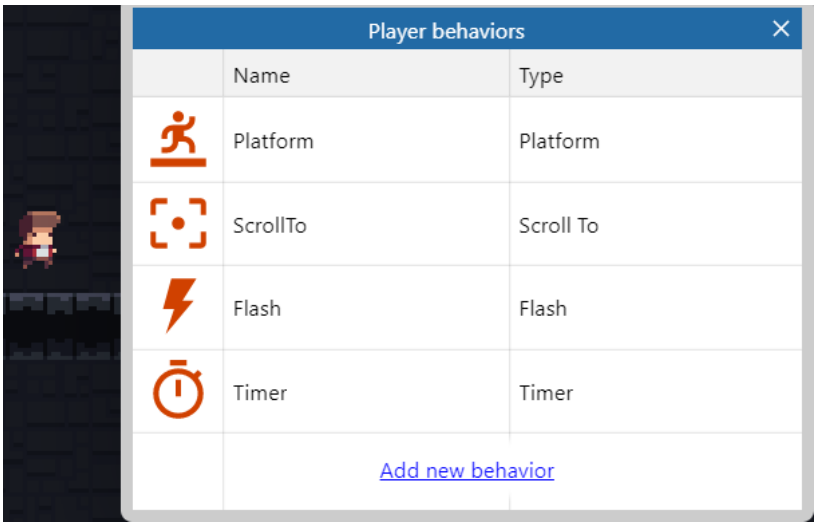
Na slici 1 je vidljivo na koji način se povezuje scena s listom događaja. Unutar liste događaja mogu se uključivati drugi događaji, primjerice događaj za kontrolu zvučnih efekata.



Slika br. 2: Uključivanje događaja za zvučne efekte

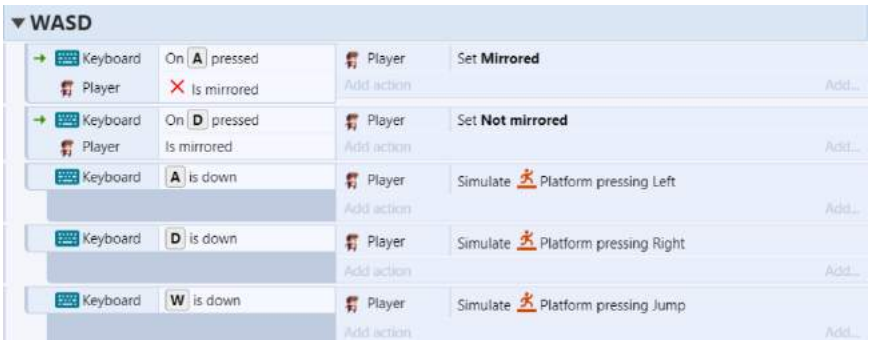
Mehanike igrača

Svakom objektu na sceni se može dodati niz tzv. ponašanja (eng. *Behavior*). To su gotove komponente koje dodjeljujemo određenom objektu. Kako bi se igraču omogućila mehanika kretanja lijevo i desno, potrebno mu je dodati *Platform* ponašanje. Jednako tako se putem različitih komponenti može objektima dodati potrebna funkcionalnost. Prikaz komponenti glavnog lika je prikazan na slici broj 3.



Slika br. 3: Komponente ponašanja igrača

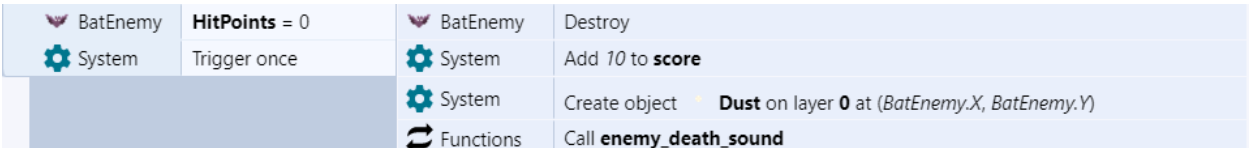
Platform komponenta na objektu igrača omogućava standardno kretanje igrača: hodanje u lijevu i desnu strane te mogućnost skoka. Za dodatnu kontrolu nad tim kontrolama isključuje se *Default controls* opcija i tada je moguće dodatno odrediti postavke kako se upravlja igračem.



Slika br. 4: Postavke kontrola igrača

Mehanika napada se u igri vrši ispaljivanjem strijele iz luka. Kako bi se ova mehanika adekvatno implementirala, potrebno je imati na umu razne scenarije. Primjerice, ako se igrač nalazi blizu zida i okrenut je prema zidu, tada moramo onemogućiti ispucavanje strijele jer za to nema mjesta. Kada nema nikakvih ograničenja, tada igrač može ispaliti strijelu: ispucavanje strijele je dozvoljeno i kada se igrač nalazi na podu i kada je u skoku. Strijela se sastoji od tri dijela: *BowPoint*, *BowLine* i *Arrow*. Strijela slobodno putuje prostorom sve dok ne naleti na neprijatelja ili zid nivoa. Poseban slučaj je da ako strijela ne naleti na prepreku a izađe iz vidnog polja ekrana, tada se ona također uništava. *BowPoint* je vrh strijele koji služi za detekciju kolizije. *BowLine* je bijela linija koja se pojavljuje iza strijele kao obris leta strijele. U slučaju kontakta strijele s neprijateljem, strijela se uništava te čini određenu štetu neprijatelju. Jednom kada se neprijatelju načini više štete nego što on

ima životnih bodova, neprijatelj se uništava i na njegovom mjestu se animira efekt prašine. Taj efekt se stvara tako da se napravi objekt tipa čestica (eng. *Particle*).



Slika br. 5: Postavke uništavanje šišmiša i stvaranje čestica prašine

Protivnici u igri

U kreiranoj igri postoje tri vrste neprijatelja: *PatrolEnemy*, *SlimeEnemy* i *BatEnemy*. *PatrolEnemy* je neprijatelj koji patrolira na određenom području. *SlimeEnemy* je jedinica koja je slična prvoj jedinici no ona svakih sekundu i pol napravi skok. BatEnemy je neprijatelj koji patrolira neko područje u zraku. Kako bi se implementiralo kretanje te jedinice u zraku u smjeru lijevo-desno koristi se *Sine* vrsta ponašanja. No također ta jedinica se vrlo blago kreće i u smjeru gore-dolje. Stoga se i za tu vrstu ponašanja koristi također *Sine* vrsta ponašanja samo u vertikalnom smjeru.

Poseban protivnik u igri je glavni neprijatelj koji ima posebnu mehaniku kretanja. Glavni neprijatelj ima *Line Of Sight* vrstu ponašanja. Ta vrsta ponašanja može detektirati određeni objekt u definiranoj liniji vida. Ako glavni neprijatelj spazi igrača u liniji vida tada se vrši akcija skoka te u određenom vremenskom intervalu također ima mogućnost stvaranja neprijatelja tipa *PatrolEnemy*. Prilikom pada glavnog neprijatelja na tlo, aktivira se potres ekrana (eng. *screen shake*).

Zaključak

Construct 3 se pokazao kao dobar alat za razvoj platformskih dvodimenzionalnih igara. Nudi podršku za najtipičnije mehanike koje se pojavljuju u ovom žanru. Vizualno skriptiranje uvelike olakšava implementaciju funkcionalnosti igre te omogućava i onima koji nemaju duboko programersko znanje da realiziraju osnovnu željenu logiku igre.

Reference

[1] Construct 3. (25. svibnja 2019.). Dostupno na: https://editor.construct.net
[2] First platform videogame in true 3D. (25. svibnja 2019.). Dostupno na: https://www.guinnessworldrecords.com/world-records/first-

platformer-in-true-3d?fb_comment_id=856925831026446_1802287559823597
[3] J. Petty: Top 12 Free Game Engines For Beginners & Experts Alike. (25. svibnja 2019.). Dostupno na: https://conceptartempire.com/free-game-engines/
[4] M. Klappenback: What is a Platform Game? (25. svibnja 2019.). Dostupno na: https://www.lifewire.com/what-is-a-platform-game-812371
[5] The best platform games on PC. (25. svibnja 2019.). Dostupno na: https://www.pcgamesn.com/best-platform-games
[6] Wikipedia: Construct (game engine). (25. svibnja 2019.). Dostupno na: https://en.wikipedia.org/wiki/Construct_(game_engine)
[7] Wikipedia: Platform game. (25. svibnja 2019.). Dostupno na: https://en.wikipedia.org/wiki/Platform_game

Izrada igre borbene arene za više igrača u programskom alatu Unity

Dario Poje i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
dpoje@foi.hr, mlkoneck@foi.hr

Sažetak

Tema ovog rada je izrada osnovnih funkcionalnosti računalne igre borbene arene implementirane u programskom alatu Unity. Igra je programiranja u C# programskom jeziku. U radu se opisuju osnovne implementirane mehanike žanra kao i podrška za mrežno igranje. Igra se temelji na standardnim mehanikama ovog žanra.

Ključne riječi: računalna igra, Unity, igra borbene arene, igra za više igrača, C#, programiranje

Uvod

Industrija računalnih igara je jedna od najbrže rastućih industrija te jedna od najprofitabilnijih industrija na svijetu [1]. Danas najveći udio u toj industriji pripada mobilnim igrama [6]. Jedan on najpopularnijih žanrova računalnih igara, igre borbene arene za više igrača (eng. *multiplayer online battle arena*), je žanr koji je popularan i na stolnim računalima i na mobilnim uređajima [2]. Najpoznatiji predstavnici ovog žanra su *League Of Legends*, *Dota 2* i *Smite*. Na mobilnim uređajima među najpopularnijim naslovima su *Vainglory* i *Mobile Legends: Bang Bang* [3].

Igre borbene arene za više igrača, ili još poznate kao akcijske strategije u stvarnom vremenu (eng. *action real-time strategy*) su igre podžanra strateških video igara. To je podžanr strategija u stvarnom vremenu gdje igrač najčešće upravlja jednim likom koji je dio tima koji se bori protiv suparničkog tima istog broja igrača. Glavni cilj je uništiti protivničku glavnu građevinu a pri tome igračima pomažu jedinice koje se periodički stvaraju i napadaju suprotnu stranu. No postoje i druge vrste ovog tipa igara kod kojih je cilj poraziti sve protivničke igrače. Svaki igrač najčešće ima svoj jedinstven set moći ili akcija te igrač postaje sve jači tijekom trajanja igre [5].

Unity je programski alat namijenjen za razvoj računalnih igara a razvila ga je kompanija *Unity Technologies*. Prva verzija ovog alata izašla je 2005. godine. A ove godine Unity je korišten za stvaranje preko pola svih mobilnih igara i 60%

svih igara virtualne stvarnosti [4]. Unity ima mogućnost izvoza igara na preko 25 platformi te postoji mnogo kvalitetnog materijala za učenje kako raditi u tom alatu.

U nastavku rada će biti opisane najvažnije metode koje su usko vezane uz osnovne mehanike igara borbene arene za više igrača: kretanje i animiranje igrača, traženje najkraćeg puta (eng *pathfinding*), osnovne mehanike borbe i mogućnost mrežnog igranja.

Kontrola kamere

Jako bitan aspekt igara borbene arene je pokretanje kamere koje je kod većine igara ovog tipa jednako. Nivo igranja je najčešće radi bolje preglednosti vidljiv pod kutom od 45 stupnjeva dok se onda kamera pomiče dijagonalno u odnosu na osnovne osi nivoa. Kamera se može pomicati strelicama (ili tipkama *ASDW*) a jednako tako i pomicanjem miša do ruba ekrana. Kotačić na mišu omogućuje kretanje kamere bliže ili dalje od nivoa. Implementirana je još jedna mogućnost a to je ubrzavanje kretanja kamere ako je pritisnuta tipa *shift*. Kako bi se brzina kretanja kamere izvodila jednakom brzinom neovisno o jačini računala na kojem se igra izvodi, koristi se *Time.deltaTime* parametar prilikom računanja brzina kretanja.

```
void MoveCamera () {
  float CamSpeed = 20f;
  float BorderTickness = 10f;
  Vector3 position = transform.position;
  if (Input.GetKey("left shift")|| Input.GetKey("right shift")) {
       CamSpeed = 50f;}
  if (Input.GetKey("d")|| Input.GetKey("right") || Input.mousePosition.x
    >= Screen.width - BorderTickness) {
       position.z += CamSpeed * Time.deltaTime;
       position.x -= CamSpeed * Time.deltaTime;}
  if (Input.GetKey("a")|| Input.GetKey("left") || Input.mousePosition.x
    <= BorderTickness) {
       position.z -= CamSpeed * Time.deltaTime;
       position.x += CamSpeed * Time.deltaTime;}
  if (Input.GetKey("s")|| Input.GetKey("down") || Input.mousePosition.y
    <= BorderTickness) {
       position.z += CamSpeed * Time.deltaTime;
       position.x += CamSpeed * Time.deltaTime; }
  if (Input.GetKey("w")|| Input.GetKey("up") || Input.mousePosition.y >=
    Screen.height - BorderTickness) {
       position.z -= CamSpeed * Time.deltaTime;
       position.x -= CamSpeed * Time.deltaTime;}
  float scroll = Input.GetAxis("Mouse ScrollWheel");
  position.y -= scroll * ScrollSpeed* 100f * Time.deltaTime;
  position.x = Mathf.Clamp(position.x, -CamLimit.x, CamLimit.x);
  position.y = Mathf.Clamp(position.y, minY, maxY);
  position.z = Mathf.Clamp(position.z, -CamLimit.y, CamLimit.y);
```

```
    transform.position = position;
}
```

Kretanje likova i mehanika borbe

Unity ima jako dobru podršku za kreiranje terena po kojem se može igrač kretati putem korištenja sustava navigacije (eng. *Navigation system*). Površina se definira kao *NavMesh* objekt koji mora biti *static* što znači da se on neće pomicati tijekom izvođenja igre. Nakon definiranja parametara, odabire se opcija *Bake* koja onda omogućava kretanje agenata po zadanoj površini. Svakom objektu koji se treba moći kretati po nivou treba dodati *NavMesh Agent* komponentu. Osim samog kretanja, ova komponenta omogućava i izbjegavanje prepreka.

Osnovna funkcionalnost kretanja je implementirana kod većine ovakvih igara na identičan način. Nakon odabir jedinice, desnim klikom miša je moguće kretati se po nivou. Ako se igra izvodi u mrežnom načinu rada, tada treba upravljati samo s lokalnim igračem.

```
if (!isLocalPlayer) return;

Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
    RaycastHit hit;
    if (Input.GetButtonDown("Fire2")) {
        if (Physics.Raycast(ray, out hit, 100)) {
            if (hit.collider.CompareTag("Enemy")) {
                targetEnemy = hit.transform;
                enemyclicked = true;}
            else {
                isAttacking = false;
                walking = true;
                enemyclicked = false;
                navAgent.destination = hit.point;
                navAgent.Resume();
            }
        }
    }
    if (enemyclicked) {
        moveandshot();}
    if (navAgent.remainingDistance <= navAgent.stoppingDistance) {
        walking = false;}
    else {
        if(!isAttacking) walking = true;
    }
anim.SetBool("isWalking", walking);
```

Ako je klik napravljen na protivničku jedinicu, tada će se lik kretati prema toj jedinici do trenutka kada neprijatelj ne dođe u radijus napada igrača. U tom trenutku će igrač napasti protivnika.

```
void moveandshot() {
    if (targetEnemy == null) {
      return;}
    navAgent.destination = targetEnemy.position;
    if (navAgent.remainingDistance >= shootingDistance) {
      navAgent.Resume();
      walking = true;}
    if (navAgent.remainingDistance <= shootingDistance) {
      transform.LookAt(targetEnemy);
      if (Time.time > nextFire) {
        isAttacking = true;
        nextFire = Time.time + timeBetweenShoots;
        Cmdfire();}
      navAgent.Stop();
      walking = false;
    }
}
[Command]
void Cmdfire() {
    anim.SetTrigger("attack");
    GameObject fireball = Instantiate(bulletPrefab, bulletSpawnPoint.position,
bulletSpawnPoint.rotation) as GameObject;
    fireball.GetComponent<Rigidbody>().velocity = fireball.transform.forward * 4;
    NetworkServer.Spawn(fireball);
}
```

Slika br. 1: Prikaz napada igrača u kreiranoj igri

Mogućnost mrežnog igranja

Kako bi se igra mogla izvoditi u modu za igranje s više igrača potrebno je omogućiti spajanja igrača na server. Kreirana igra se sastoji od jedne aplikacije koja može predstavljati i poslužitelja i klijenta, ovisno o tome spaja li se igrač već na postojeći poslužitelj ili će on biti u ulozi poslužitelja. Mrežno igranje se realizira putem komponente *NetworkMenager* koja omogućuje odabir objekata

koje je potrebno prikazati na zaslonu svakog igrača, sinkroniziranje s poslužiteljem i definiranje početnih pozicija za svakog igrača.

Zaključak

Unity uvelike olakšava implementacije ideje u konkretnu računalnu igru jer ima mnogo komponenti koje automatiziraju neke najčešće korištene funkcionalnosti koje jedna računalna igra ima. Tako i kod spomenutog žanra u ovom radu, uvelike je pojednostavljen proces traženja najkraćeg puta kao i mogućnost implementacije mrežnog igranja. U ovom radu su opisane neke temeljne mehanike ovog žanra no one predstavljaju tek golu osnovu na kojoj se onda gradi ostatak igre. Izrada kompletnih mehanika nekog žanra je poprilično složen proces te je potrebno dosta znanja i iskustva kako bi se čak i putem Unity-ja realizirala željena ideja.

Reference

[1] T. Dobrilova. How Much Is the Gaming Industry Worth? (13. rujna 2019.). Dostupno na: https://techjury.net/stats-about/gaming-industry-worth/

[2] MOBA explained: One of the most popular genres in esports is making a push on mobile. (13. rujna 2019.). Dostupno na: https://www.abacusnews.com/who-what/moba-explained-one-most-popular-genres-esports-making-push-mobile/article/2163097

[3] The Most Popular MOBA Video Games Right Now. (14. rujna 2019.). Dostupno na: https://www.ranker.com/list/most-popular-moba-video-games-today/ranker-games

[4] Unity: Public relations. (14. rujna 2019.). Dostupno na: https://unity3d.com/public-relations

[5] Wikipedia: Multiplayer online battle arena. (14. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Multiplayer_online_battle_arena

[6] Wikipedia: Video game industry. (13. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Video_game_industry

Izrada igre obrane tornjeva u programskom jeziku C++ i SDL-u

Tomislav Vugrinec, Danijel Radošević i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
tvugrinec@foi.hr, darados@foi.hr, mlkoneck@foi.hr

Sažetak

Tema ovog rada jest izrada igre obrane tornjeva (eng. *Tower Defense game*) u programskom jeziku C++ zajedno s alatima potrebnim za izgradnju izvornog C++ koda te korištenjem dinamičke *SDL 2.0* biblioteke čije se funkcionalnosti koriste za upravljanje događajima od korisnikova pritiska tipke na mišu ili tipkovnici pa sve do spremanja slika (eng. *sprite*) objekata igre u *VRAM* memoriju (eng. *Video Random Access Memory*) unutar grafičke kartice.

Ključne riječi: računalna igra, igra obrana tornjeva, C++, C++ STL, SDL, objektno orijentirano programiranje

Uvod

Igra obrane tornjeva je podžanr strateških igara u stvarnom vremenu (eng. *real-time strategy game*) [8]. Najčešći cilj igre je svladati valove neprijatelja koji se kreću zadanim putevima prema resursu koji igrač mora obraniti. Igrač se brani izgradnjom tornjeva koji se grade na putu ili oko puta kojim idu neprijatelji. Postoje različiti tipovi tornjeva koji imaju svoje jedinstvene vrste napada a najčešće se mogu i nadograđivati. Tipičan predstavnik tog žanra je igra *Bloons Tower Defense 5* [1]. Također postoje i igre koje imaju različite varijante osnovnih mehanika ovog žanra. Primjerice, u igri *Onslaught 2* osim osnovnih mehanika uvodi se koncept pojačanja (eng. *combo*) ukoliko su određeni tipovi tornjeva izgrađeni u neposrednoj blizini [3]. U sklopu ovog rada izrađen je prototip igre koji implementira osnovne mehanike ovog žanra.

Simple DirectMedia Layer (SDL) je razvojna biblioteka u cijelosti pisana u C programskom jeziku koja podržava razvoj aplikacija na svim poznatijim operacijskim sustavima i platformama: Windows, GNU/Linux, macOS, iOS i Android [4]. Uključivanje biblioteka se u C++ programskom jeziku vrši putem naredbe *include* te je potrebno uključiti adekvatne biblioteke ovisno o željenoj funkcionalnosti. Biblioteka omogućava spremanje grafičkih elemenata direktno u memoriju grafičke kartice i time se mogu postići veće performanse. SDL biblioteka služi za upravljanje vizualnim elementima, audio elementima, ulaznim jedinicama, dretvama, mrežnim postavkama, tajmerima i dr. [7]. Što

se tiče grafike, biblioteka podržava *OpenGL*, *Vulkan* ili *Direct3D*. Sama biblioteka je licencirana *zlib* licencom što korisniku omogućava besplatno korištenje i modifikaciju biblioteke stoga ne čudi činjenica da se i danas ova biblioteka koristi u komercijalne svrhe. Popularna igra *FTL: Faster Than Light* je izrađen putem ove biblioteke. Igre *Unreal Tournament* i *Unreal Tournament 2004* su također izrađene korištenjem ove biblioteke. Ranije verzije programskog alata *UnrealEngine* (verzije 1, 2 i 2.5) su također rađene korištenjem ove biblioteke [6]. Tvrtka *Valve* koja je najpoznatija po izradi platforme *Steam* je koristila ovu biblioteku za izradu *Steamoverlay* funkcionalnosti njihove platforme [5].

Alati razvoja računalne igre

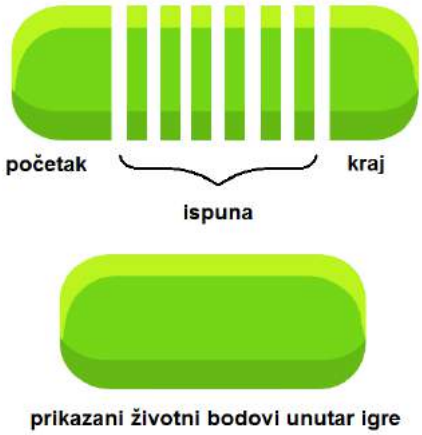
Za pisanje programskog koda korišten je proširivi uređivač teksta *Emacs 25.2.2.* Kod je pisan objektno orijentiranim pristupom te je podijeljen na više izvornih (*.cpp*) datoteka koje s datotekama zaglavlja (*.h*) čine jednu cjelinu. Za verzioniranje koda korišten je besplatni Git sustav. Projekt je kompajliran (eng. *compiled*) korištenjem *Clang+ 6.0.0* kompajlera i *Clang-tidy* alata *LLVM* projekta koji prilikom pisanja koda igre simultano provjerava postoje li greške ili potencijalne greške unutar napisanog koda vodeći se smjernicama C++ 14 standarda [2]. Način kompajliranja izvornih datoteka definiran je u *Makefile* datoteci korištenjem *make* sustava za izgradnju izvršnih datoteka (eng. *make build system*). Nivo igre rađen je u besplatnom programu *Tiled* koji je također otvorenog koda, a za interakciju igrača s igrom (događaji tipkovnice, miša, prikazivanje i spremanje slika igre) korištena je SDL 2.0 dinamička biblioteka (eng. *dynamic library*).

Nivo igre

Prilikom pokretanja igre, igrač odmah ulazi u prvi nivo igre. Na nivou je unaprijed određen put kretanja neprijatelja koji se stvaraju i kreću u valovima. Glavni zadatak je eliminirati neprijatelje prije nego što dođu do kraja puta. Ako neprijatelj dođe do kraja puta, igrač gubi životne bodove. Ako igrač izgubi sve životne bodove tada je igra gotova. Kako bi porazio neprijatelje, igrač mora graditi tornjeve pored definiranog puta neprijatelja koji uništavaju neprijatelje. Igrač mora strateški postavljati tornjeve po nivou kako bi uspio savladati neprijatelje.

Slika br. 1: Igračev pogled na igru

Vizualni element prikaza životnih bodova osmišljen je tako da se prilikom gubitka životnih bodova najprije uklanja ispuna a na kraju kada igrač izgubi posljednji životni bod, tada se uklanja početak i kraj tog vizualnog elementa prikazanog na slici broj 2.

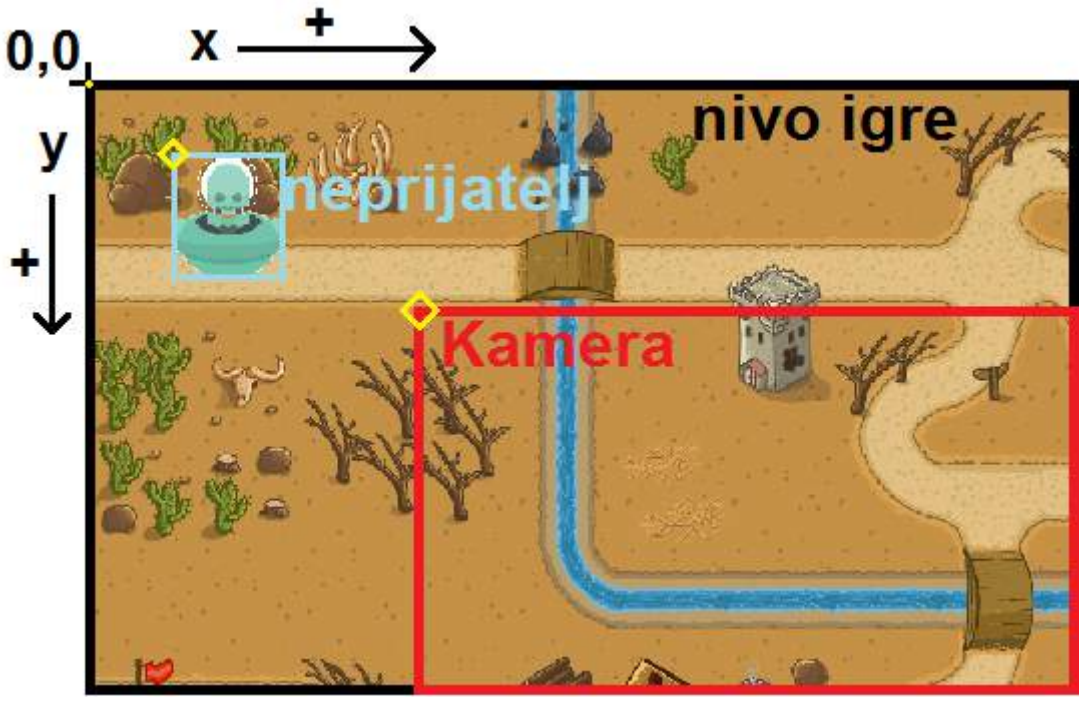


Slika br. 2: Vizualizacija prikaza životnih bodova igrača

Implementacija kamere

S obzirom da je razina mnogo veća od onoga što se stane prikazati na ekranu, potrebno je implementirati zasebnu komponentu kretanja kamere. Kamera je implementirana kao pravokutnik dužine i širine prozora aplikacije. Kretanje kamere se izvodi tako da se mijenjaju koordinate pozicije kamere. Kao sidrište koristi se gornja lijeva točka ekrana. Mehanika kretanja kamere je

implementirana na način da ako korisnik približi miš rubu ekrana na određenu udaljenost tada se pogled pomiče u smjeru kursora dok se ostali elementi prikazuju relativno prema promijenjenim pozicijama kamere što je prikazano na slici broj 3.



Slika br. 3: Vizualizacija koordinata i prikaz pogleda kamere

U slučaju sa slike, pozicija kamere je takva da igrač ne bi mogao vidjeti neprijatelja koji se stvorio u gornjem lijevom dijelu karte koji nije vidljiv. Relativno prikazivanje elemenata ovisno o kameri se može objasniti sljedećim kratkim isječkom koda:

```
void Enemy::prikazi(int camX, int camY) {
   sprite->render( (_position.x) - camX,
   (_position.y – sprite->getVisina()) - camY );
}
```

Budući da SDL prozor grafičke elemente počinje prikazivati od gore lijevog prema donjem desnom kutu počevši od polazišta s koordinatama (0, 0) pa sve do koordinate (duzina_prozora, visina_prozora), oduzimanjem koordinata kamere od koordinata neprijatelja može se primijetiti kako će rezultat prikaza biti s obje negativne koordinate , što je za SDL sasvim u redu i označava kako se taj element neće prikazivati na prozoru aplikacije.

Implementacija neprijatelja

Svaki neprijatelj mora pamtiti određene podatke poput svoga trenutnog stanja životnih bodova i trenutne pozicije na nivou kroz koji prolazi.

```
class Enemy {
  public:
    Enemy( path& put);       // konstruktor Enemy klase
    void walk();             // metoda kretanja neprijatelja
    long getID() { return _id; }
    ...

  private:
    static int newID;        // sakriveni djeljeni id neprijatelja
    long _id;                // id stvorenog neprijatelja
    SDL_Point _position;
int _health;

    int _WalkNodeIndex;      // index čvora cilja
    path _path;              // nasumično proslijeđen put
}
```

No osim tih podataka svaki neprijatelj posjeduje svoj jedinstveni ID dodijeljen neposredno nakon njegovog nastanka. Pomoću tog podatka tornjevi mogu usmjeriti svoj metak prema točno određenom neprijatelju koji je ušao u zonu koju taj toranj brani. Također, provjerom postojanja IDa sprječava se problem zalutalog metka, jer ukoliko više tornjeva uputi metak prema neprijatelju koji je upravo ubijen, tada se svi metci koji idu prema tom neprijatelju uništavaju. Neprijatelji se pokreću svakih 20 milisekundi pomoću javne *walk()* metode koja svakim izvođenjem smanjuje udaljenost između neprijatelja i cilja.

Zaključak

U okviru ovog rada izrađena je računalna igra obrane tornjeva u C++ programskom jeziku proširenim SDL 2.0 dinamičkom bibliotekom. Za razvoj aplikacije odabran je C++ jezik zato što podržava objektno orijentirani stil programiranja i kompatibilnost s C programskim jezikom zbog koje se u C++ programski kod mogu integrirati naredbe SDL biblioteke koje se koriste za čitanje i upravljanje događajima budući da je SDL biblioteka u potpunosti pisana u C jeziku. Nadalje, C++ jezik pruža i STL spremnike podataka različitih namjena od kojih su neki korišteni za spremanje objekata unutar igre.

SDL biblioteka nudi funkcionalnosti koje su implementirane na način da ostavljaju minimalni otisak (eng. *footprint*) na resurse računala. Ta činjenica ne iznenađuje s obzirom na činjenicu da je čitava biblioteka pisana u C programskom jeziku koji je namijenjen razvoju aplikacija visokih performansa. Prilikom implementacije igre, uočene su razlike u implementaciji funkcionalnosti korištenjem vektora i spremnika liste. Primjerice, vektor je sporiji prilikom brisanja s početka ili sredine spremnika a lista ne podržava [] operator pristupa.

SDL 2.0 biblioteka nudi mogućnost izrade stabilnih aplikacija izvrsnih performansi a sam programski jezik omogućava veliku fleksibilnost u izradi aplikacija različitih namjena a primarno za razvoj računalnih igara.

Reference

[1] Bloons TD 5. (6. rujna 2019.). Dostupno na: https://www.kongregate.com/games/ninjakiwi/bloons-td-5
[2] Clang frontend for LLVM. (7. rujna 2019.). Dostupno na: https://clang.llvm.org/
[3] Onslaught 2. (6. rujna 2019.). Dostupno na: https://www.newgrounds.com/portal/view/385302
[4] Simple DirectMedia Layer. (6. rujna 2019.). Dostupno na: https://www.libsdl.org/
[5] Steam. (7. rujna 2019.). Dostupno na: https://store.steampowered.com/
[6] Wikipedia: List of games using SDL. (7. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/List_of_games_using_SDL
[7] Wikipedia: Simple DirectMedia Layer. (7. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Simple_DirectMedia_Layer
[8] Wikipedia: Tower defense. (6. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Tower_defense

Implementacija osnovnih mehanika igre uloga u programskom alatu Unreal Engine 4

Ariana-Lea Zuber i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
ariana.zuber@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu opisane su osnovne mehanike igrača vezane uz žanr računalnih igara uloga (eng. *action role-playing games*). Na temelju tih mehanika, kreiran je i prototip računalne igre uloga u programskom alatu *Unreal Engine 4* za demonstraciju spomenutih mehanika. Biti će dan kratak opis žanra i njegovih glavnih predstavnika dok se u drugom dijelu rada opisuju glavni elementi kreirane igre. Fokus je na opisu osnovnih mehanika i određenih algoritama za njihovu implementaciju.

Ključne riječi: računalna igra, akcijska računalna igra uloga, Unreal Engine 4, C++, programiranje, algoritmi

Uvod

Žanr akcijske računalne igre uloga je podžanr temeljnog žanra igara uloga (eng. *role-playing games*) [6]. Temeljno obilježje igara uloga je to da igrač upravlja likom (ili grupom likova) u zamišljenom svijetu [7]. Mnoge igre ovog tipa svoje korijene imaju u društvenim igrama uloga (eng. *tabletop role-playing games*) i dijele sličnu terminologiju i mehanike igranja. Akcijske igre uloga naglasak stavljaju na borbu u stvarnom vremenu. Također često ima akcijsko-avanturističke elemente poput sustava za misije (eng. *quests*). Također su popularne u verzijama igara uloga za više igrača (eng. *massively multiplayer online role-playing games*). Neki od glavnih predstavnika su *The Elder Scroll* serijal, *The Witcher* serijal, *Fallout* serijal, *Dragon Age* serijal, *Mass Effect* serijal i dr. [2].

Uobičajeno, ove vrste igara su u tri dimenzije te imaju pogled iz trećeg lica preko ramena glavnog lika igre. Temeljna karakteristika je razvijanje jačine i sposobnosti glavnog lika a jednako tako kroz igru pojavljuju se neprijatelji koji su sve snažniji i pametniji. Kako se je razvijao ovaj žanr, tako je i realizam umjetne inteligencije neprijatelja postajao sve bolji i bolji: neprijatelji su postajali po stilu igranja sličniji igraču, razvijen je sustav osjetila stoga neprijatelji mogu „vidjeti“ i „čuti“ igrača. U novije vrijeme, kako su rasle performanse računala, tako su se pojavile i igre koje se temelje na konceptu

otvorenog svijeta gdje igrač ima veliku slobodu istraživanja svijeta i veliku raznolikost sadržaja. Otvoreni svjetovi omogućavaju nelinearan pristup igranju igre tako da svaki prolazak igre može biti nešto drugačiji od prethodnog [7].

U kreiranoj igri implementirat će se osnovne mehanike akcijske igre uloga. Igrač ima mogućnost izbora između dvije klase likova. Implementirane su osnovne mehanike životnih bodova, energije te mogućnost skupljanja iskustva (eng. *experience*). Igrač može nositi različitu opremu te je također implementiran sustav inventara (eng. *inventory*) u koji igrač može spremati objekte. Implementirana je i mogućnost kupovine i prodaje objekata za određenu valutu koju igrač može skupiti. Igrač ima mogućnost kretanja i borbe s neprijateljima.

Igrač za kretanje može koristiti miš po principu pokaži i klikni, ali više je uobičajeno da se likom upravlja strelicama na tipkovnici. Pomakom miša kamera lika igrača se rotira dok se s posebnim mehanikama upravlja tipkama na tipkovnici, poput otvaranje inventara, prikaza zadataka, prikaza opreme i sl.

Dizajn i implementacija igre

Kreirana igra je implementirana u programskom alatu *Unreal Engine 4* [3]. Za potrebe izrade igre korištena ne službena dokumentacija *Unreal Engine-a* i video poduke s *Virtus Learning Hub/Creative Tutorials* korisničkog računa na *YouTube* servisu [4][1]. 3D modeli korišteni u kreiranom projektu preuzeti su s Unreal Engine trgovine [5].

Na početku igre igrač može izabrati između dva lika s kojima može igrati.

Slika br. 14: Izbor klase

Čarobnjak i ratnik su dvije standardne klase koje se javljaju u igrama uloga stoga je igraču omogućeno igranje s jednim od ova dva lika. Kao što je već ranije navedeno, igračem se upravlja tipkovnicom dok se rotacija vrši putem miša. Odabirom određene klase, igrač ulazi u igru.

Osnovne moći klasa

Svaka klasa u igri ima svoje jedinstvene specifičnosti. Čarobnjak ima mogućnost brže regeneracije životnih bodova. Aktivacijom posebne moći, bodovi energije se smanjuju dok se životni bodovi povećavaju. Obično se ovakve moći mogu upotrijebiti na samom liku koji moć aktivira, u regiji na određenom radijusu ili primjerice na suigrača. Varijacija na temu bila bi nanošenje štete određenom neprijatelju ili svima u određenoj regiji.

Druga moć čarobnjaka je bacanje projektila. U vizualnoj implementaciji navedene mehanike koristi se *Particle System* unutar *Unreal Engine-a*. Prilikom aktivacije moći, na poziciji ruke čarobnjaka instancira se projektil koji se ispucava u zadanom smjeru, zadanom brzinom i određenim kutom. Projektil na sebi ima mehaniku detekcije kolizije stoga se prilikom kolizije s drugim objektima u igri uništava. Varijacije na temu ove mehanike su bilo koja vrsta napada na daljinu.

Ratnik ima mogućnost napada na blizinu, konkretno, ovdje se radi o napadu mačem neposredno ispred igrača. Oružje na sebi ima detekciju kolizije te u kontaktu s neprijateljskom jedinicom ozljeđuje neprijatelja za određenu vrijednost. Korištenje moći ili napadanje troši određenu količinu energije koja se regenerira s vremenom. Prirodno, što je jača šteta napada, to je veća i potrošnja energije.

Slika br. 15: Osnovne moći klasa - regeneracija, napad projektilom i napad oružjem

Osnovno sučelje igre

Što se tiče sučelja igre, ono se sastoji od nekoliko elemenata: grafički prikaz životnih bodova i energije, inventara, sučelja trgovine i prikaz moći/napada i napretka lika.

Slika br. 16: Elementi sučelja igre

U gornjem lijevom kutu nalazi se *avatar* lika ispod kojeg se nalazi razina iskustva lika. Zelenom linijom označeni su životni bodovi lika a plava linija označava količinu energije.

Inventar je na početku igre nevidljiv. Iznad životnih bodova nalazi se ime lika. Pritiskom na tipku „B“ moguće je inventar učiniti vidljivim i nevidljivim. Interakcija s trgovinom je moguća u neposrednoj blizini trgovine. Tada se u izborniku dućana vide elementi koje igrač može kupiti za određenu količinu novaca. Izbornik inventara i trgovine moguće je zatvoriti pritiskom na crveni gumb u gornjem lijevom kutu svakog prozora.

Na dnu sučelja nalazi se linija napretka ili iskustva. Kako igrač skuplja iskustvo, puni se linija iskustva. Kada se ona napuni, igrač napreduje na iduću razinu i na taj način jača svog lika. Svaka razina zahtjeva duplo više iskustva od prethodne razine za napredak na sljedeću razinu. Ovisno o klasi lika, navedene su moći igrača. Na prikazu moći moguće je vidjeli je li određena moć raspoloživa za korištenje ili ne. Putem sučelja se također pojavljuju određen poruke vezane uz trenutno stanje igrača poput poruke da je inventar pun i sl.

Protivnici igrača

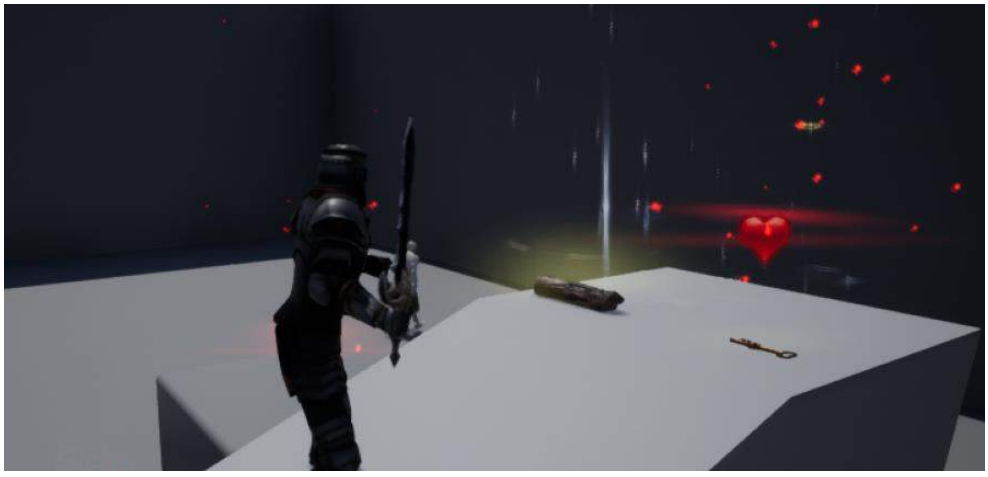
Protivnici igrača u svom inicijalnom stanju šeću terenom. Za navigaciju neprijatelja koristi se *NavMesh* komponenta. Ta komponenta omogućava kretanje po terenu te izbjegavanje prepreka. Protivnici imaju senzor vida te

ako igrač uđe u njihovo definirano vidno polje, protivnik će krenuti prema igraču kako bi igraču nanio štetu. Da bi protivnici napali igrača moraju se približiti igraču na neposrednu udaljenost. Ako neprijatelj ubije igrača, igra je gotova. Ako igrač savlada neprijatelja, tada neprijatelj ostavlja nagradu u obliku vrećice koja sadrži nasumičnu količinu novaca i ostale nasumično generirane objekte. Igrač prilaskom vrećici može pokupiti njen sadržaj.

Slika br. 17: Vreća koju neprijatelj ostavlja kao nagradu

Nagrade

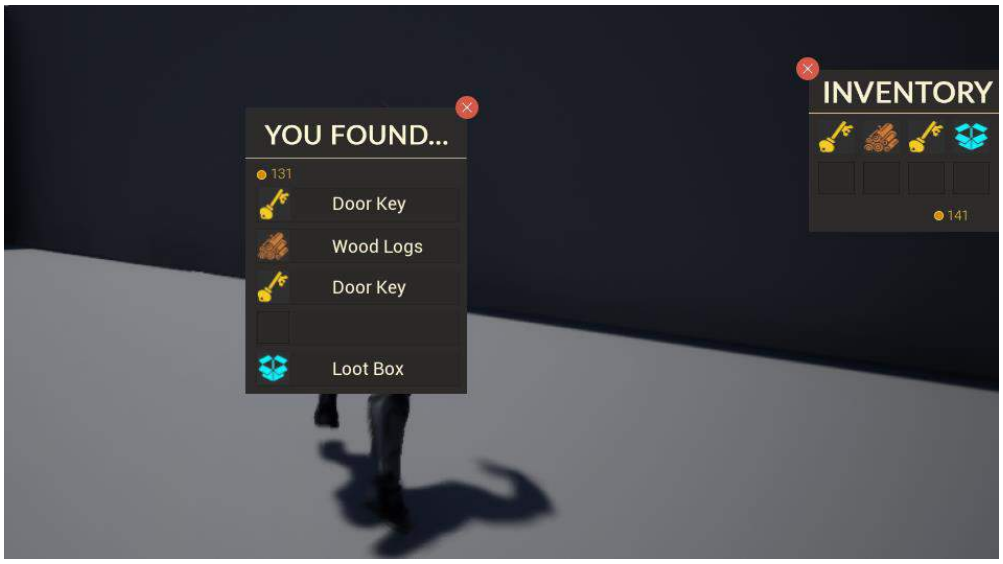
Nagrade se nalaze na unaprijed definiranim mjestima ili kao vrećica koja ispada neprijatelju nakon što je svladan. Kao nagrade, igrač može pokupiti drvo, ključ i apstraktni objekt koji vraća životne bodove a igraču je predstavljen sustavom čestica. Vizualni prikaz spomenutih objekata je vidljiv na slici broj 5.



Slika br. 18: Objekti koje je moguće pokupiti

Prilikom prikupljanja vrećice koju ostavljaju neprijatelji nakon savladavanja, nasumično se generiraju nagrade koji se nalaze u vrećici. Ovo je tipična

mehanika koja je česta u igrama uloga gdje se prolaskom kroz neku regiju nasumično skupljaju određeni resursi koje igrač kasnije može koristiti u trgovini, za izradu drugih predmeta i slično. Na slici broj 6 prikazan je prozor koji je korisniku vidljiv prilikom generiranja nagrada: prvo je prikazana količina generiranih novčića, ključeva i ostalih objekata. Nagrade je moguće pokupiti sve dok ima mjesta u inventaru. Ako je inventar pun, tada igrač o tome dobiva obavijest.



Slika br. 19: Izbornik sadržaja vreće

Na početku igre igrač u inventaru ima svega deset novčića. Novac se pribraja postojećem iznosu a ostali objekti se dodaju dok ima mjesta.

Zaključak

Akcijske igre uloga predstavljaju zanimljiv žanr računalnih igara jer imaju niz zanimljivih mehanika igranja. Mnoge od tih mehanika se javljaju i u drugim žanrovima računalnih igara. *Unreal Engine 4* znatno olakšava implementaciju mnogim segmenata računalne igre uloga jer ima ugrađene gotove sustave za implementaciju tipičnih mehanika poput kretanja, modeliranja ponašanja likova kojima ne upravlja igrač i sl. Osim podrške za izradu programske logike, podrška postoji i za druge segmente kao što je animiranje.

Nadogradnju osnovnih mehanika je moguće proširiti na mnogo načina: protivnicima se može omogućiti složenije ponašanje detaljnijim modeliranjem njihovog ponašanja, mogu se modelirati neprijatelji različite inteligencije, inventar se može nadograditi da objekti imaju svoju težinu ili veličinu i slično. Iz pogleda dizajna, balansiranje svih parametara težine igre je vrlo složena radnja: igra ne smije biti ni prelagana da ne bude dosadna niti preteška da

frustrira igrača. Također jedan smjer je dodavanje dodatnog sadržaja u obliku novih modela likova ili novih sposobnosti: nove moći i nove klase likova.

Reference

[1] Creating A Role Playing Game - Unreal Engine 4 Course. (1. rujna 2019.). Dostupno na: https://www.youtube.com/watch?v=pgIJqiZQ-2Y&list=PLL0cLF8gjBpqA8DcrhL_O9kD4jsUqhDR6

[2] The Best Action Role-Playing Games of All Time. (14. rujna 2019.). Dostupno na: https://www.ranker.com/list/all-action-role-playing-games-list/reference

[3] Unreal Engine 4. (15. rujna 2019.). Dostupno na: https://www.unrealengine.com/en-US/

[4] Unreal Engine 4 Documentation. (15. rujna 2019.). Dostupno na: https://docs.unrealengine.com/en-US/index.html

[5] Unreal Engine Marketplace. (1. rujna 2019.). Dostupno na: https://www.unrealengine.com/marketplace/en-US/assets?keywords=infinity%20blade

[6] Wikipedia: Action role-playing game. (14. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Action_role-playing_game

[7] Wikipedia: Role-playing video game. (14. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Role-playing_video_game

Izrada strateške igre u programskom alatu GameMaker Studio 2

Jakov Štulić i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
jstulic@foi.hr , mlkoneck@foi.hr

Sažetak

Tema ovog rada je izrada računalne igre u programskom alatu *GameMaker Studio 2*, posebnom alatu upravo za razvoj računalnih igara. Ukratko će se opisati aspekti razvoja prototipa strateške igre, napravljene po uzoru na igru *Heroes of Might and Magic*, igru pod imenom *Knjiga blaga*. To je igra na poteze koja koristi heksagonalnu rešetku za raspored jedinica na razini. U kreiranom prototipu kreirane su tri razine na kojima se demonstriraju mehanike igre ovog žanra.

Ključne riječi: računalna igra, strateška igra, GameMaker Studio 2, GML, programiranje, algoritmi

Uvod

Game Maker Studio 2 je programski alat za razvoj računalnih igara koji je razvilo poduzeće *YoYo Games* [7]. Na tržište izlazi 1999. godine pod nazivom *Animo* [2]. Izrađena igra u ovom alatu se može izvoziti na sve važnije platforme: *Windows*, *macOS*, *Android*, *iOS*, konzole *Playstation 4*, *Xbox One* i *Nintendo Switch* [2]. Ovaj alat ima podršku za vizualno skriptiranje tehnikom *Drag-and-drop* a također se skripte mogu izrađivati putem *GML* programskog jezika [3].

Iako je alat nešto jednostavniji od nekih drugih besplatnih programskih alata za razvoj računalnih igara, mnoge izvrsne igre su izrađene korištenjem ovog alata. Popularni naslovi su *Spelunky*, *Undertale*, *Hotline Miami*, *10 Second Ninja X*, *Cool, Serve, Delicious! 2!!* i mnoge druge [5]. Korisničko sučelje alata je vrlo jednostavno, dokumentacija je također jako dobro napisana tako da svatko tko želi razviti igru korištenjem ovog alata vrlo lako može krenuti u realizaciju svoje ideje.

Žanr strateških igara

Računalne strateške igre su igre čiji je fokus vješto promišljanje i planiranje kako bi se ostvario željeni cilj. Naglasak je na strategiji, taktičkim i logističkim izazovima. Često su uključeni i razni izazovi te koncept istraživanja [6]. Ovaj

žanr ima veliki broj podžanrova no dva najpopularnija su igranje na poteze (eng. *Turn-Based Strategy*) i strategija u stvarnom vremenu (eng. *Real-Time Strategy*). Kreirana igra je igra tipa strategije na poteze te će se u njoj implementirati tipične mehanike borbe ovog žanra. Prema A. Rollingsu i E. Adamsu, iza svih uspješnih strateških igara stoje tri važna svojstva: sukob, istraživanje i razmjena [1].

- Sukob – glavna komponenta koja nudi zanimljive mehanike borbe
- Istraživanje – nepoznata i nasumična priroda igre koja nudi kontinuirani izazov i nepredvidivost te zahtjeva donošenje novih odluka
- Razmjena – upravljanje resursima i korištenje istih kako bi se ostvarili željeni ciljevi

Kreirana igra

Kreirana igra se temelji na konceptu da naizmjenice igrač i protivnik pomiču svoje jedinice pri tome da svatko u jednom krugu pomiče sve svoje dostupne jedinice. Cilj borbe je naravno savladati sve jedinice neprijatelja. U slučaju da igrač ostane bez svih svojih jedinica, tada je izgubio borbu. U kreiranoj igri igrač dobiva natrag izgubljene jedinice na pojedinoj razini. Sa svakom razinom igre protivničke jedinice su teže sa svladati a dobivaju i neke nove sposobnosti koje igrač mora uzeti u obzir kako bi uspio nadmudriti protivnika. Igra koja je postala klasik u ovom žanru, a ujedno je bila glavna inspiracija za kreiranje ovog prototipa, je igra *Heroes of Might and Magic* [4].

Tematski, igra je zamišljena u izmišljenom svijetu gdje se za borbu koriste srednjovjekovna oružja i magije. Igrač upravlja s grupom ratnika koji su u potrazi za blagom koje se nalazi skriveno u špilji. Mnogi avanturisti su već pokušali doći do skrivenog blaga no do sada to još nikome nije pošlo za rukom. U ovu priču ulazi igrač koji će pokušati ostvariti taj cilj.

Za igranje igre, mora se koristiti miš kao ulazna jedinica a također se može a ne mora koristiti tipkovnica kao alternativa za određene akcije. S razinama igre, pozadina igre se mijenja i dočarava lokacije koje s razina postaju sve opasnije. Za kretanje boraca, korištena je heksagonalna rešetka. Format rešetke ostaje nepromijenjen od razine do razine no na pojedinim razinama postoje prepreke koje onemogućavaju kretanje na određenim ćelijama rešetke. Na slici br. 1 se može vidjeti dizajn prve razine igre gdje su vidljivi borci, heksagonalna rešetka te pozadina.

Dimenzije rešetke su 5 redaka puta 13 stupaca. Širina je duža s obzirom da se sukobljene strane nalaze na lijevoj i desnoj strani stoga se borba odvija primarno u tom smjeru. Početne pozicije su uvijek na najudaljenijim horizontalnim ćelijama.

Slika br. 1: Prva razina kreirane igre

Jedinice

Jedinice u kreiranoj igri se dijele u dvije osnovne skupine: jedinice kojima upravlja igrač i jedinice protivnika. Igrač ima tri vrste jedinica: *mačevalac*, *kopljanik* i *strijelac*. Kopljanik ima jaču obranu i slabiji napad dok mačevalac ima jači napad a nešto slabiju obranu. Strijelac ima slabu obranu no mogućnost pucanja na protivničke jedinice preko cijelog bojišta. Protivnik ima također tri vrste jedinica: *kostur*, *ljigavac* i *prizivač duhova*. Kostur je najslabija jedinica koja može napadati susjednu ćeliju. Ljigavac ima posebnu mehaniku da si regenerira životne bodove nakon svakog kruga. Prizivač duhova može oživjeti ubijenog kostura. Zajednička svojstva svih jedinicama su:

- Životni bodovi - odnosi se na broj štete koju jedinica može primiti prije nego što se uništi
- Minimalni napad – Minimalna količina štete koju jedinica može napraviti pri napadu (bez izračuna s obranom)
- Maksimalni napad - Maksimalna količina štete koju jedinica može napraviti pri napadu (bez izračuna s obranom)
- Obrana – Količina štete koju jedinica ignorira u slučaju da je napadnuta

Kretanje

Sve jedinice imaju mogućnost kretanja i to u radijusu od 2 heksagona u svim smjerovima. Jedinice se ne mogu kretati na pozicije gdje se nalazi druga

jedinica ili neka prepreka. Kako bi se kretanje olakšalo igraču, nakon što igrač odabere željenu jedinicu s kojom se želi kretati, osvjetljene su mu ćelija na koje se ta jedinica može pomaknuti. Desnim klikom miša se vrši akcija kretanja te je kretanje te jedinice onemogućeno do idućeg poteza.

Slika br. 2: Kretanje jedinica

Napadanje

Svaka jedinica također ima mogućnost napada. Sve jedinice, osim strijelca, mogu napadati protivnika na ćeliji koja je u neposrednoj blizini ćelije na kojoj se on nalazi. Strijelac može napadati protivnika na bilo kojoj ćeliji na nivou. Nakon što jedinica izvrši akciju napada, ta jedinica ne može izvršiti nikakvu akciju do idućeg kruga. Računanje napada štete se vrši tako da se uzme nasumična vrijednost između maksimalne i minimalne štete te jedinice te se oduzima vrijednost obrane jedinice koja je napadnuta.

Magije

Osim napadanja jedinicama, igraču su na raspolaganju i određene magije koje može koristiti jednom po potezu. Koliko magija igrač za vrijeme nivoa može koristiti je ograničeno resursom koji se zove mana. Igrač na početku igre dobiva 15 jedinica mane i ta mana se tokom igre ne regenerira stoga igrač mora pažljivo promisliti kada koristiti magije. Igraču su na raspolaganju tri vrste magija:

- Grom – magija koja čini štetu jednoj odabranoj jedinici te ovaj napad u potpunosti ignorira obranu napadnute jedinice

- Liječenje – magija koja vraća životne bodove odabranoj jedinici no ne može vratiti više životnih bodova od maksimuma životnih bodova jedinice
- Štit – magija koja podiže obranu odabranoj jedinici do kraja nivoa

Slika br. 3: Dostupne magije u igri

Zaključak

GameMaker Studio 2 je programski alat za izradu računalnih igara se pokazao kao izuzetno dobar alat za implementaciju osnovnih mehanika strateške igre na poteze. Jako je dobar za početnike jer omogućava vizualno skriptiranje koje je pregledno i intuitivno. Ipak, potrebno je proučiti dostupnu dokumentaciju kako bi se pohvatale osnove rada.

Ključna stvar kod strateških igara na poteze je parametrizacija i optimizacija težine igre. Igra mora biti dovoljno teška da bude interesantna no opet ne smije biti preteška kako bi stvarala frustraciju kod igrača. Uvođenje novih mehanika tijekom igranja igre je poželjna stvar koja čini igru zanimljivom. Također, trebalo bi imati u vidu svrhu svaku jedinicu i svaku magiju kod dizajniranja nivoa kako bi igrača natjerali na korištenje svih mogućih resursa u ostvarenju željenog cilja igre.

Reference

[1] A. Rollings i E. Adams (2003). *Andrew Rollings and Ernest Adams on Game Design*. SAD, San Francisco, New Riders.
[2] GameMaker Studio. (12. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/GameMaker_Studio
[3] GML Overview. (12. rujna 2019.). Dostupno na: https://docs.yoyogames.com/source/dadiospice/002_reference/001_gml%20language%20overview/
[4] Wikipedia: Heroes of Might and Magic. (12. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Heroes_of_Might_and_Magic
[5] Wikipedia: List of GameMaker Studio games. (12. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/List_of_GameMaker_Studio_games

[6] Wikipedia: Strategy video game. (12. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Strategy_video_game
[7] YoYo Games: GameMaker Studio 2. (12. rujna 2019.). Dostupno na: https://www.yoyogames.com/gamemaker

Izrada igre uloga u programskom alatu Unity

Karlo Vuljanko i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
kvuljanko@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu opisuje se proces izrade igre uloga u programskom alatu Unity. U radu će biti opisane glavne značajke i mehanike ovog žanra. Glavni fokus biti će na igraču i njegovim glavnim karakteristikama.

Ključne riječi: računalna igra, igra uloga, Unity, C#, programiranje

Uvod

Proces razvoja računalnih igara je izuzetno složen posao koji uključuje timove ljudi koji moraju imati specifična znanja i vještine iz mnogih različitih domena. Primjerice, programeri igara moraju imati znanja osnova programiranja, određena znanja iz područja matematike i fizike. Grafički dizajneri, ovisno o vrsti igre moraju imati sasvim drugi set vještina dok osoba koja radi glazbu i zvučne efekte mora dobro baratati znanjem glazbene kompozicije, audio produkcije i dr.

Korištenje programskih alata za izradu računalnih igara poput *Unity-ja*, *Unreal Engine-a, Godot-a* i sl. uvelike može pomoći u brzini i kompleksnosti izrade računalne igre [5]. Ključne funkcionalnosti pogona igre (eng. *game engine*) su proces renderiranja grafike, sustav fizike, detekcije kolizije, sustav za reprodukciju zvuka, skriptiranje, animiranje, modeliranje umjetne inteligencije i dr. [5]. Također, mogućnost izvoza igre na više platformi je značajna karakteristika koja može uvelike pomoći kod razvoja igara za više platformi. Korišteni alat za razvoj kreiranog prototipa igre u ovom radu je programski alat *Unity* [8]. Unity pruža mogućnost razvoja 2D i 3D igara kao i igara virtualne i prilagođene stvarnosti. Mnogi mali razvojni timovi odabiru upravo ovaj alat za razvoj svojih računalnih igara a i mnoge popularne igre su izrađene putem ovog programskog alata: Hearthstone, Wasteland 2, Assassin's Creed serijal, Deus Ex: The Fall i mnoge druge igre [1].

U ovom radu kreiran je prototip igre uloga u programskom alatu Unity. Igra uloga je žanr računalnih igara gdje igrač ima ulogu određenog lika u imaginarnom svijetu [6]. Glavni aspekti ovog žanra su bogata naracija, najčešće povezana uz složeni strukturirani proces donošenja odluka kroz koji

se razvija lik igrača. Mnoge igre se temelje na društvenim igrama uloga (eng. *tabletop role-playing games*) [7]. Te igre dijele sličnu terminologiju, mehanike igre i okolinu. Također jedna od ključnih karakteristika je interaktivan i otvoreni svijet (eng. *open world*). Najpopularniji podžanr ovog tipa igara je igra uloga za više igrača putem interneta (eng. *masive multiplayer online role playing game*). Primjerice, *World of Warcraft* je igra tog tipa, stara preko 15 godina a igru igra i dalje oko pet milijuna igrača [2].

Sučelje kreirane igre

Kod igara uloga, prikaz karakteristika igrača je izuzetno bitan aspekt igre. Kod mnogih igara ovog tipa se na sučelju nalazi jako puno informacija koje igraču govore o njegovom trenutnom stanju napretka lika. Najčešće karakteristike koje su prikazane su životi igrača, aktivne misije, informacija o opremi, vrsti oružja koje je aktivno i sl. U kreiranoj igri sučelje je rađeno po uzoru na nešto modernije igre gdje se primjenjuje princip minimalizma stoga se na korisničkom sučelju nalazi samo onaj najbitniji podatak a to su životi igrača. Sve ostale bitne informacije o igri i o liku igrača se nalaze u izborniku koji se može prikazati u svakom trenutku. Sučelje igre se u programskom alatu Unity implementira putem objekta tipa *Canvas*. *Canvas* objekt se u igri učitava kao dvodimenzionalna slika i postavlja se kao sloj iznad ostalih elemenata igre.

Slika br. 1: Kadar iz kreirane igre

Mehanike igrača

U ovom radu fokus je na implementaciji osnovnih mehanika igrača. Kreirana igra je trodimenzionalnog tipa te su osnovne mehanike igrača njegovo kretanje te interakcija s okolinom i drugim likovima u igri (eng. *non-playable character*). Osnovne karakteristike igrača su tipične karakteristike za ovaj žanr

igara: životi igrača, njegova snaga, jačina i oprema koju nosi. Te karakteristike određuju koliko ukupno štete igrač može primiti a također i količinu štete koju on nanosi svojim protivnicima. Klasično, povećanjem iskustva i nabavom bolje opreme igraču rastu mogućnosti preživljavanja i borbe.

Kao i kod velike većine igara ovog tipa, igrač na početku nema oružje niti dodatnu opremu već kroz napredak u igri igraču se omogućuje prikupljanje opreme i oružja. Na početku igre igrač se može kretati kroz kreirani svijet i vršiti osnovnu interakciju s okolinom i drugim likovima. U interakciji s objektima u igri igrač može pronaći opremu i napitke za regeneraciju života. U određenom trenutku igre, igraču u posjed dolazi oklop i oružje: štit i mač te nakon tog trenutka može napadati protivnike.

Kako bi se mogla vršiti interakcija s okolinom, potrebno je provjeriti nalazi li se igrač na minimalno dozvoljenoj udaljenosti za interakciju s određenim objektom. Ako se igrač nalazi na adekvatnoj udaljenosti, ispisuje se odgovarajuća poruka. U varijablu *myDistance* se upisuje udaljenost između igrača i objekta s kojim se želi vršiti interakcija. Nakon provjere je li ta udaljenost manja od zadane udaljenosti, prikazuje se tekst na ekranu te se vrši radnja na pritisak odgovarajuće ulazne kontrole.

```
myDistance = Vector3.Distance(portal.transform.position,
playerObject.transform.position);
    if (myDistance < 3)
    {
       onScreenText.GetComponent<Text>().text = "[E]  Interract";
       if (Input.GetButtonDown("Use"))
       {
          //Sample code
       }
```

Ključan aspekt kod većine računalnih igara je borbena interakcija. U borbi s drugim likovima igrač može izgubiti određenu količinu zdravlja. Ako količina životnih bodova padne na nulu, igrač je izgubio igru. Ova logika se može vidjeti u kodu ispod: funkcija *setHealth()* se poziva prilikom promjene životnih bodova igrača. Funkcija prima dva parametra: *healthModifier* i *type*. Prvi parametar definira količinu zdravstvenih bodova koja se mijenja dok drugi parametar određuje radi li se o oduzimanju ili dodavanju životnih bodova. Nakon modifikacije potrebno je provjeriti je li količina životnih bodova pala ispod nule. Ako jest, igra je gotova. Metoda *healthBar.fillAmount* služi ažuriranju informacije o životnim bodovima na sučelju igre.

```
public void setHealth(int healthModifier, int type)
  {
     if (healthModifier != 0) {
```

```
if (type == 0) {
   if (health + healthModifier < maxHealth)
   {health += healthModifier;}
                    else{health = maxHealth;}
} else {health -= healthModifier;}
if (health >= 0) {
   healthPercent = (float)health / (float)100;
   healthBar.fillAmount = healthPercent;
                    }
if (health <= 0) {
   healthBar.fillAmount = 0;
   endGame();
```

Upravljanje kamerom

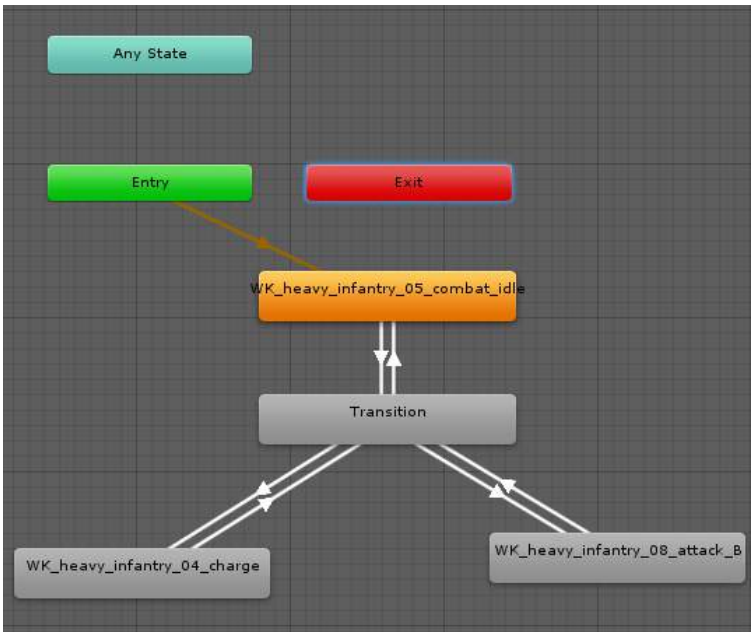
Mehanika kretanja igrača se direktno veže uz kretanje kamere putem koje igrač igra igru. Dobro pozicioniranje kamere je ključan aspekt na koji treba obratiti pozornost jer ako se kamera ne ponaša na željeni način to može uvelike pokvariti iskustvo igranja igre. Kamera u kreiranoj igri prati lik igrača te ima parametri visine i udaljenosti koji se dinamički mijenjaju, ovisno o sceni. Također, postoji i opcija manualnog rotiranja kamere putem desnog klika miša. Kada igrač pomiče miš te istodobno drži desnu tipku miša kamera se rotira u skladu s pomakom miša. Točan kut rotacije računa se pomoću *Quaternion.Euler* funkcije.

```
if (camButtonDown)
  {
   x += Input.GetAxis("Mouse X") * xSpeed * 0.02f;
   y += Input.GetAxis("Mouse Y") * ySpeed * 0.02f;
   y = Mathf.Clamp(y, -50.0f, 30.964f);
   Quaternion rotation = Quaternion.Euler(30.964f - y, x, 0);
   Vector3 position = rotation * new Vector3(0.0f, 0.0f, -walkDistance - 0.8f) +
target.position;
    _myTransform.rotation = rotation;
    _myTransform.position = position;
  }
```

Kretanje i animiranje igrača

Kao što je rečeno, kamera i kretanje lika su usko povezani. Za implementaciju kretanja korišten je *Unity Standars Asset* paket u kojem se nalazi kontroler za kretanje i animiranje lika. Dodavanjem animacija i parametrizacijom tog kontrolera namještaju se željene karakteristika izgleda i mogućnosti kretanja lika igrača. Ponekad je potrebno mijenjati parametre kretanja ovisno o situaciji. Model igrača se također mijenja ovisno o njegovoj opremi i oružju kojeg posjeduje.

Što se animiranja tiče, potrebno je odrediti koje sve pokrete igrač ima na raspolaganju te uvjete tranzicije iz jednog stanja u drugo. Igrač ima tri osnovna stanja: stanje mirovanja, stanje kretanja i stanje napadanja oružjem. Putem nekoliko varijabli se testira stanje igrača i na temelju toga se izvode željene animacije.



Slika br. 2: Kontroler za animiranje lika igrača

Prilikom dizajna igre, postoji mnogo detalja koje je potrebno definirati. Što se kretanja tiče, primjerice, može li se igrač kretati i napadati u isto vrijeme, što se dešava ako igrač trči pa pritisne tipku za napad i sl. U kreiranoj igri igrač ne može napadati i trčati u isto vrijeme.

```
else if (anim.GetBool("attacking") == false)
{
   anim.SetBool("running", true);
   anim.SetInteger("condition", 1);
   moveDir = new Vector3(0, 0, 1);
   moveDir *= speed;
   moveDir = transform.TransformDirection(moveDir);
}
```

Inventar

Inventar (*engl. Inventory*) je mjesto gdje igrač posprema objekte koje pronalazi tijekom igre: opremu, napitke, oružja i druge elemente. Inventar je neizostavna funkcionalnost svake igre uloga i predstavlja suštinski dio ovog žanra. Svaki objekt koji se može pokupiti je objekt klase *Item* dok je svako mjesto u inventaru objekt klase *InventorySlot*. Osnovne funkcionalnosti inventara su dodavanje, korištenje i uzimanje stvari.

```
public void AddItem (Item newItem)
{
        item = newItem;
        icon.sprite = item.icon;
        icon.enable = true;
        removeButton.interactable = true;
}
```

Zaključak

Unity kao razvojni alat nudi brojne mogućnosti za izradu kvalitetnih 2D i 3D igara. Također je vrlo pogodan alat za učenje jer postoji mnogo materijala za učenje u obliku dokumentacije, poduka i video poduka. Unity dosta olakšava proces izrade računalne igre jer su mnoge funkcionalnosti automatizirane ili se mogu realizirati kroz razna grafička sučelja. U ovom radu su opisane osnovne mehanike igre uloga koje predstavljaju osnovu na kojoj se mogu nadograđivati dodatne mehanike.

Reference

[1] Soomla: Top 10 Unity Games Ever Made. (10. rujna 2019.) Dostupno na: https://blog.soomla.com/2015/01/top-10-unity-games-ever-made.html

[2] Statista: stimated number of World of Warcraft subscribers from 2015 to 2023 (in millions). (10. rujna 2019.). Dostupno na: https://www.statista.com/statistics/276601/number-of-world-of-warcraft-subscribers-by-quarter/

[3] Unity: Animation controller. (10. rujna 2019.). Dostupno na: https://docs.unity3d.com/Manual/class-AnimatorController.html

[4] Unity: Post-processing overview. (10. rujna 2019.). Dostupno na: https://docs.unity3d.com/Manual/PostProcessingOverview.html

[5] Wikipedia: Game engine. (10. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Game_engine

[6] Wikipedia: Role-playing game. (10. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Role-playing_game

[7] Wikipedia: Role-playing video game. (10. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Role-playing_video_game

[8] Wikipedia: Unity. (10. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Unity_(game_engine)

Izrada inkrementalne igre za Android operacijski sustav pomoću programskog alata Unity

Stiven Drvoderić, Danijel Radošević i Mladen Konecki
Fakultet Organizacije i Informatike u Varaždinu
stiven.drvoderic98@gmail.com, darados@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu biti će opisane mehanike igre za mobilne uređaje inkrementalnog (eng. *Incremental*) tipa te će biti objašnjena njena implementacija u programskom alatu Unity. Glavna karakteristika ovog žanra je implementacija beskonačnog broja razina pomoću formula koje postepeno povećavaju težinu igre. Kreirani prototip igre sadrži tipične mehanike ovog žanra.

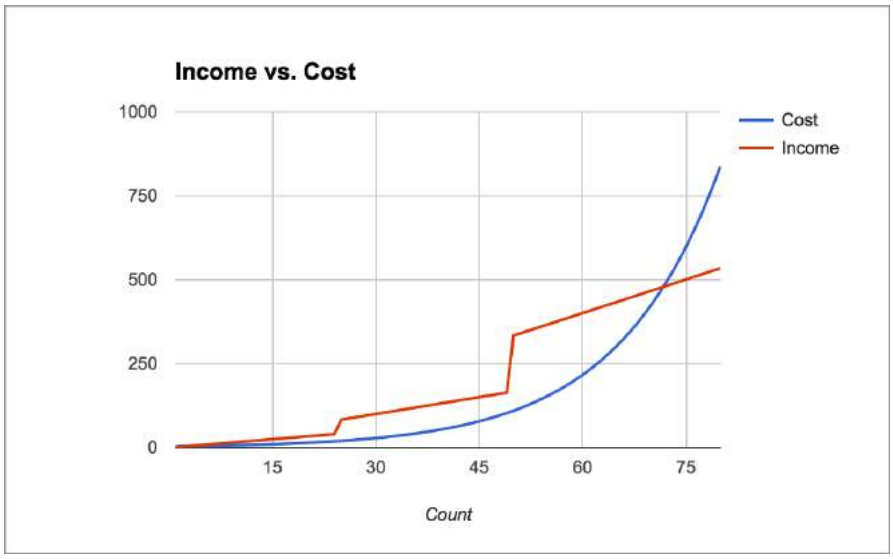
Ključne riječi: računalna igra, inkrementalna igra, mobilna igra, Unity, Android, C#, programiranje, algoritmi

Uvod

Inkrementalne igre su računalne igre gdje je glavna mehanika igranja neka jednostavna radnja poput uzastopnog klikanja po ekranu [8]. U nekim igrama ovog tipa klikanje postaje nepotrebno te nakon nekog vremena, igra igra samu sebe, zato se ova vrsta igara još naziva igra klikanja (eng. *clicker*) ili igra praznog hoda (eng. *idle*). Prva takva igra kreirana je 2002. godine i zvala se *Progress Quest* [2]. Ovaj žanr je stekao svoju popularnost 2013. godine kada je izašla uspješna igra pod nazivom *Cookie Clicker* [5]. Cilj igre je bio generirati što veći broj kolačića koji su se stvarali tako što je igrač mišem pritiskao ikonu kolačića. 2015. na *Steam* platformi je izašla igra *Clicker Heroes* koja je dodatno popularizirala ovaj žanr [8].

O popularnosti ovog žanra govori i podatak da su 2015. godine na internet stranici s igrama *Kongregate* od 5 najpopularnijih igara bile 3 igre upravo ovog tipa [7]. Ovaj podatak je predstavio urednik sadržaja stranice *Kongregate* Anthony Pecorella koji je ujedno i autor jedne od najpopularnijih inkrementalnih igara današnjice, AdVenture Capitalist [4].

Kako je glavna karakteristika tih igara da su beskonačne, za realizaciju tog koncepta igru je potrebno dizajnirati primjenom odgovarajućih matematičkih funkcija koje bi osigurale takvu funkcionalnost. Korištenjem dobro balansiranih formula, postiže se balans između rasta cijena u igri i generirane primarne valute. Kod izrade takvih formula najbitniji faktori su: primarna valuta, generator, valuta zamjene, množitelji i prestiž (eng. *Prestige*) [3].



Slika br. 1: Odnos cijene i prihoda u inkrementalnim igrama

Primarna valuta se koristi za kupovanje raznih pojačanja koja opet povećavaju zaradu. Generatori su objekti koji izvršavanjem akcija generiraju zaradu te se također njihovim jačanjem povećava zarada. Valuta zamjene u nekim slučajevima predstavlja vrijednost dobivenu od generatora koju je onda potrebno konvertirati u primarnu valutu. Množitelji su svi objekti koji u igri povećavaju zaradu ili neku drugu vrijednost. Prestiž je mehanika koja se otključava kada korisnik dođe do određene razine gdje se progres igre resetira u zamjenu za posebnu valutu koja se opet može zamijeniti za neki množitelj [3].

Opis kreirane igre

Kreirana igra implementirana je u programskom alatu Unity [6]. Unity je programski alat koji ima podršku za kreiranje 2D, 3D igara i igara virtualne stvarnosti. Također, igre kreirane u ovom alatu se mogu izvoziti na više od 25 platformi stoga je ovaj alat vrlo prikladan za realizaciju ovakvog projekta.

Kreirana igra ima naziv *Idle Heor Slayer* te onda implementira tipične mehanike inkrementalnog žanra igara. Primarna mehanika igre je klikanje po ekranu a kontekst je da igrač mora napadati neprijatelja sve dok ga ne ubije čime dobiva primarnu valutu koja je u ovoj igri zlato. Nakon što je neprijatelj ubijen, na ekranu se pojavljuje novi.

U igri je implementiran i koncept generiranja primarne valute. To je realizirano tako da igrač može kupiti partnera primarnom valutom te on tada napada neprijatelje i dok igrač ne igra igru, tj. Bez prisutnosti igrača. To je *idle* dio mehanike u kreiranoj igri.

Kao primarna valuta zamjene u igri se koristi šteta koju heroj ili partneri nanose neprijatelju. Nakon dovoljne količine nanesene štete, valuta zamjene se može konvertirati u zlato.

Množitelji u kreiranoj igri su implementirani u obliku ljubimaca koje igrač može kupiti u trgovini. Oni povećavaju neki od generatora. Također tu je i oprema koju igrač dobiva ubijanjem neprijatelja a također može otključati i dodatne vještine kupovinom istih u trgovini. Vještine se mogu uključiti na određeno vrijeme kako bi privremeno pojačale generatore, tj. Igrača ili partnere. Zadnje, tu si i artefakti koji trajno pojačavaju generatore.

Kada igrač u određenom trenutku dođe do razine neprijatelja i primijeti da je šteta koju mu nanosi vrlo mala, tada može aktivirati prestiž. Time se stanje igre resetira na početak a igraču trajno ostaje oprema i ljubimci. U zamjenu za prestiž igrač dobiva posebnu valutu s kojom može pribaviti posebne artefakte. S tim artefaktima igrač otključava novu dimenziju množitelja koja mu daje mogućnost povećanja trajanja vještina ili jačanje ostalih generatora.

Kreiranje generatora

Generatori se u igri mogu podijeliti u dvije kategorije: aktivne i neaktivne generatore. Aktivni generator se aktivira prilikom interakcije igrača s igrom, tj. klikanjem u igri. Neaktivni generatori su zaduženi za implementaciju partnera koji napadaju neprijatelje čak i kada igrač nije aktivan. Jednom kada se otključaju, početni napad partnera je jedan napad u sekundi.

Kako bi se ove mehanike mogle izvoditi bez potrebe za mrežnim pristupom, potrebno je kreirati bazu podataka u kojoj će se spremati podatci o trenutnom napretku igrača. Ta baza se mora nalaziti na sustavu na kojem se izvodi igra. U konkretnom slučaju, to je mobilni uređaj s Android operacijskim sustavom na kojem je odabrana SQLite baza podataka. U bazi se nalaze svi potrebni podatci i o ostalim elementima igre: generatorima, množiteljima i sl.

Nakon kreiranja baze podataka, moguće je implementirati generatore. Aktivni generator je heroj, tj. igrač. Logika igrača je implementirana putem C# skripte koja opisuje igrača i sadrži potrebne funkcije za interakciju s ostalim objektima. Kod pokretanja igre, dohvaćaju se podatci iz baze o igraču: razina heroja, maksimalna razina, trenutna razina, količina zlata i dr.

Inkrementalni aspekt osobina heroja se postiže pomoću funkcija za izračun cijene pojačanja heroja i funkcije za izračun ukupne štete.

```
public double IzracunajCijenuPojacanja(int brojZeljenihLevela) {
    return (double) (5*
          (System.Math.Pow(1.062,levelHeroja+brojZeljenihLevela)
          System.Math.Pow(1.062, levelHeroja)) / 0.062);}

public void PovecajLevel() {
     brojZlata = VelikiBroj.OduzmiOd(brojZlata,trenutnaCijenaPojacanja);
     levelHeroja += trenutniBrojPojacanjaLevela;
     trenutnaCijenaPojacanja =
     VelikiBroj.pretvoriBroj(IzracunajCijenuPojacanja
     (trenutniBrojPojacanjaLevela));
     GameObject.Find("TextPojacajLevelTrece").GetComponent<Text>().text
     = VelikiBroj.PrikazVelikogBroja(trenutnaCijenaPojacanja); }
```

U funkciji *IzracunajCijenuPojacanja* je vidljiva formula putem koje se vraća vrijednost primarne valute koja je potrebna kako bi se pojačao heroj. Druga funkcija najprije igraču uzima potrebno zlato te jača razinu heroja nakon čega računa novu cijenu idućeg pojačanja. Još jedan problem kod ovakvog tipa igara je što su brojčane vrijednosti koje se koriste vrlo velike stoga je potrebno kreirati strukturu pomoći koje se takve velike brojeve dohvaća.

Neaktivni generatori su također implementirani pomoću funkcije koja zbraja štetu svih partnera pomoću matematičke formule i zatim vraća vrijednost štete.

Interakcija generatora i neprijatelja

Interakcija između heroja i neprijatelja postiže se korištenjem *Collider* mogućnosti Unity alata. *Collider-om* se označava područje na koje korisnik može pritisnuti kako bi se registrirao udarac heroja. Područje za udarac je cijeli ekran. Zatim se kreira skripta koja provjerava sve pritiske na ekran te šalje poruku klasi heroja da izvrši udarac ako postoji pritisak koji je pogodio *collider*.

```
var stanjeEkrana = Input.touches;
if (Input.touchCount > 0) {
  foreach (var pritisak in stanjeEkrana) {
     Vector2 podrucjeZaPritisak =
     Camera.main.ScreenToWorldPoint(pritisak.position);

     if (pritisak.phase == TouchPhase.Began &&
     Physics2D.OverlapPoint(podrucjeZaPritisak) ==
     GetComponent<Collider2D>()) {
        animacija_Heroj.SetTrigger("JePritisnuto");
        GameObject.Find("Heroj").GetComponent<Heroj_Klasa>()
        .SendMessage("Napad",podrucjeZaPritisak);
        break;
     }
  }
```

```
}
```

Funkcija najprije provjerava postoje li pritisci na ekranu koji se nalaze spremljeni unutar *Input.Touches*. Ako postoje, provjerava se je li pritisnuta točka unutar *Collider-a* i ako je, šalje se poruka s imenom funkcije za napad heroja.

Druga interakcija je između partera igrača i neprijatelja. Ona se implementira korištenjem funkcije *IEnumerator*. Jačanje partnera vrši se na isti način kao i jačanje heroja s jednom razlikom, a ta razlika je ta da se podatci partnera spremaju odmah u bazu podataka. Pošto se svi podaci o partnerima nalaze u bazi podataka, izračun ukupne štete se vrši jednostavnim zbrajanjem štete svih partnera pomoću SQL upita koji traži sve partnere koji su trenutno aktivni. Izračun štete vrši se jedanput svake sekunde što se postiže *IEnumerator* funkcijom u nastavku.

```
private IEnumerator NapadPartnera() {
    while (true) {
      ukupnaSteta = izracunajUkupnuStetu();
      GameObject.Find("Neprijatelj").
      GetComponent<Neprijatelj_Klasa>()
      .SendMessage("PrimiUdarac", ukupnaSteta);
      yield return new WaitForSeconds(frekvencijaNapada);
    }
}
```

U kodu je vidljiva specifičnost *IEnumerator* funkcije, a to je čekanje ovisnosti o frekvenciji napada do idućeg izvođenja funkcije. U kombinaciji s beskonačnom petljom dobivamo željenu funkcionalnost.

Kreiranje množitelja

Kao što je već spomenuto, množitelji su svi objekti koji mijenjalju jačinu generatora i ima ih više vrsta. Množitelji su u inkrementalnim igrama sadržaj koji se neprestano nadodaje kako bi igrači mogli dostizati sve veće i veće razine u igri.

Prvi množitelj koji je igraču dostupan su vještine heroja. Vještine se implementiraju na način da se na određeno vrijeme promijene vrijednosti unutar pripadajućih klasa heroja, partnera ili neprijatelja.

Ostali množitelji se implementiraju na isti način, a to je dinamičko generiranje objekta pomoću informacija koje se nalaze u bazi podataka. Ako igrač želi vidjeti trenutno dostupnu opremu, potrebno je kliknuti na torbu i odabrati vrstu opreme. Putem SQL upita se dohvaćaju vrijednosti iz baze te se na ekranu prikazuju potrebni objekti.

Slika br 2: Prikaz opreme igrača

Ostali dijelovi igre

Od ostalih nespomenutih dijelova preostala je funkcionalnost prestiža koja se implementira izvršavanjem potrebnih upita nad bazom podataka kako bi se izmijenili podaci o množiteljima, generatorima i neprijatelju.

Trgovina služi za kupovanje množitelja, a za kupovinu se koristi posebna valuta koja se dobije prodajom nepotrebne opreme. Oprema se prodaje tako da korisnik pritisne na ikonu za prodaju. Pritisak se detektira pomoću *RayCast* funkcije te se izvođenjem funkcije može pročitati lokalna varijabla funkcije tipa *RayCastHit* koja sadrži informacije o pogođenom objektu pa se pomoću toga određuje koji objekt je potrebno obrisati.

Zaključak

Industrija računalnih igara je u neprestanom porastu te tržište mobilnih igara predstavlja najveći segment tog tržišta. Iako relativno mlado, tržište igara na mobilnim uređajima čini 47% ove industrije [1]. Ovaj podatak govori u prilog tome da se isplati investirati svoje vrijeme i znanje u učenje razvoja računalnih igara. Također, žanr inkrementalnih igara je sve popularniji stoga se isplati investirati znanje u ovoj domeni. No treba imati na umu da kako bi se uspjelo izdvojiti u odnosu na konkurenciju, valja biti inovativan i kreativan.

U ovom radu je kreiran prototip jedne inkrementalne igre koja implementira tipične mehanike svog žanra. Prednost ovog žanra je također jednostavnost i lakoća dodavanja novog sadržaja. Također, ovaj žanr je pogodan za monetizaciju jer se lako mogu unutar igre prodavati pojačanja za igrače koji žele većom brzinom napredovati u igri.

Reference

[1] 20 Mobile Gaming Statistics That Will Blow You Away | Mobile Gaming Industry Stats. (12. rujna 2019.). Dostupno na: https://influencermarketinghub.com/mobile-gaming-statistics/

[2] A. Le Conte: Idle Games, Everything You Need to know! (11. rujna 2019.). Dostupno na: https://mobilefreetoplay.com/idle-games-everything-you-need-to-know/

[3] A. Pecorella: The Math of Idle Games, Part I. (11. rujna 2019.). Dostupno na: https://blog.kongregate.com/the-math-of-idle-games-part-i/

[4] AdVenture Capitalist on Steam. (11. rujna 2019.). Dostupno na: https://store.steampowered.com/app/346900/AdVenture_Capitalist/

[5] S. Parkin: The rise of games you (mostly) don't play. (11. rujna 2019.). Dostupno na: https://www.gamasutra.com/view/news/237926/The_rise_of_games_you_mostly_dont_play.php

[6] Unity. (11. rujna 2019.). Dostupno na: https://unity.com/

[7] Why Players Love Idle Games & Why They're Great for Ads. (11. rujna 2019.). Dostupno na: https://www.chartboost.com/blog/mobile-games-that-play-themselves-great-for-rewarded-ads/

[8] Wikipedia: Incremental game. (11. rujna 2019.). Dostupno na: https://en.wikipedia.org/wiki/Incremental_game

Izrada strateške igre na poteze u programskom alatu Unity

Tomislav Jukica i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
tjukica@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu opisana je izrada strateške igre na poteze u programskom alatu Unity. U igri se implementiraju osnovne mehanike ovog žanra: kretanje igrača, pronalaženje najkraćeg puta, prepreka i osnovna mehanika pucanja. Same mehanike se temelje na strateškoj igri *XCOM: Uneny Unknown* prema kojoj je i izrađen praktični dio za demonstraciju.

Ključne riječi: računalna igra, strateška igra na poteze, Unity, C#, A* algoritam, programiranje

Uvod

Strateške igre predstavljaju jedan od temeljnih žanrova u području računalnih igara [7]. Ove igre se dijele u dva temeljna podžanra: strateške igre u stvarnom vremenu (eng. *real-time strategy*) i strateške igre na poteze (eng. *turn-based strategy*) [8]. Prva strateška igra na poteže je igra na ploči koja je svima dobro poznata, a zove se šah [4] i upravo je implementacija te igre bila prva strateška igra na poteze na računalu 1976. godine [6]. 1977. godine razvijena je prva ratna strateška igra na poteze pod imenom Empire [6]. Igra koja je postala klasik i definirala glavne mehanike ovog žanra je igra koju je razvio Julian Gollop kao nastavak igre Laser Squad a to je igra *UFO: Enemy Unknown* [9]. Nekoliko publikacija je izjavilo kako je to jedna od ponajboljih igara tog vremena. Ovaj žanr je ponovno doživio veliki uspjeh 2012. godine izlaskom igre *XCOM: Enemy Unknown*, modernom preradom starog spomenutog klasika iz 1994. godine [10]. Prema *Ranker-u*, *XCOM 2* je najbolja igra ovog žanra do današnjeg dana. Izuzetno uspješni predstavnici ovog žanra su: *Heros of Might and Magic* serijal, *Civilization* serijal, *Into the Breach* i drugi [1].

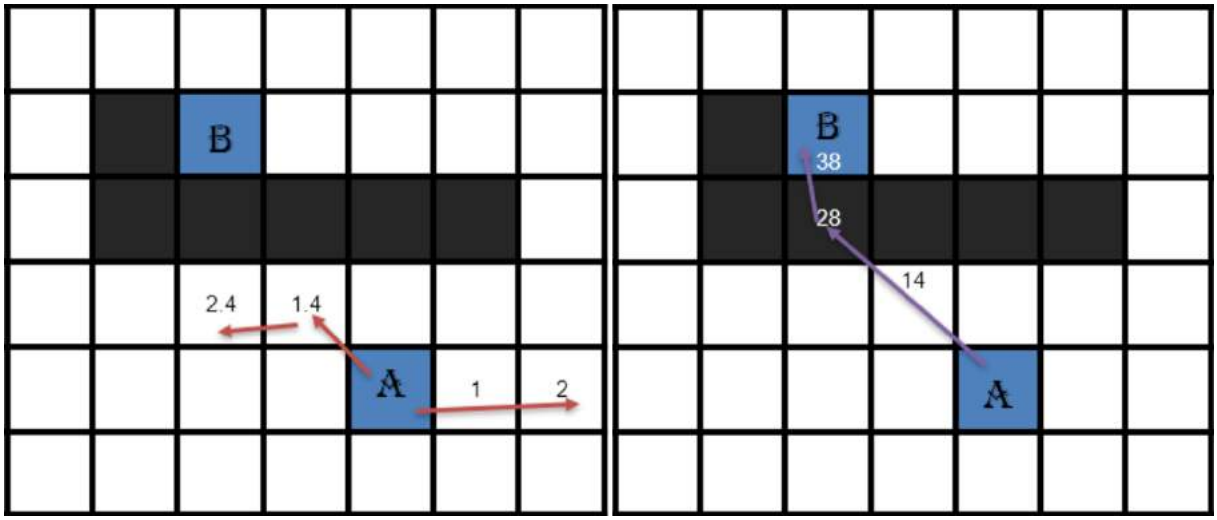
Osnovne mehanike igre

Osnovna mehanika igranja ovog žanra je kretanje igrača prostorom u potezima. Kako bi se implementirao ovaj koncept, kreirani nivo se dijeli u polja kojima se igrač može kretati. To polje se naziva mreža kretanja. Svako polje sadrži potrebna svojstva i potrebne informacije za igranje: osnovno svojstvo

su koordinate polja u odnosu na ostala polja. Također može li se tim poljem prolaziti ili se tu nalazi zid. Pojedina polja predstavljaju zaštitna polja gdje je igrač djelomično zaštićen od napada neprijatelja. Kako bi se omogućilo kretanje kreiranom mrežom, potrebno je primijeniti neki algoritam pronalaska najkraćeg puta nad tom mrežom. Postoji niz algoritama koji se bave ovom problematikom, svaki sa svojim prednostima i manama. Za implementaciju traženja najkraćeg puta u ovoj igri odabran je A* algoritam [3].

A* algoritam

A* algoritam je kreiran 1968. godine te je napravljen kao ekstenzija Dijsktrinog algoritma. A* algoritam postiže bolje performanse zbog korištenja heuristike [3]. Da bi se došlo s jednog mjesta na drugo, prvo se pojednostavljuje područje pretrage, odnosno radi se mreža s poljima. Nakon što je područje podijeljeno na polja, označava se početna točka (A) te ishodišna točka (B) te polja koja nisu prohodna (obojena crnom bojom).



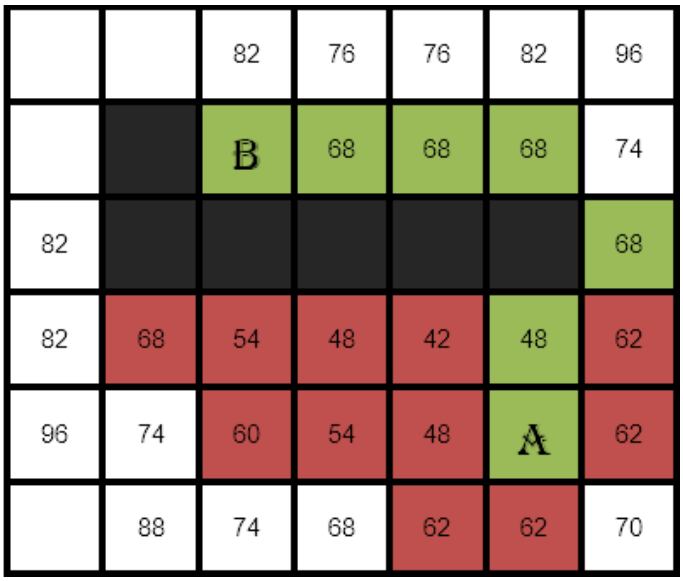
Slika br. 1: G i H cijene kretanja

Jednom kada je područje kretanja pojednostavljeno, za implementaciju potrebne su dvije liste:

- Otvorena lista – u nju se zapisuju sva polja po kojima se moguće kretati u procesu pronalaska najkraćeg puta
- Zatvorena lista – već promatrana polja za kretanje

Algoritam se izvodi tako da se prvo u otvorenu listu dodaje početna pozicija A i sva susjedna polja tog početnog polja. Tu je potrebno odlučiti je li dijagonalno kretanje poljem validno. U implementiranoj igri dijagonalno kretanje je dozvoljeno no treba imati na umu da se kod dijagonalnog kretanja prevaljuje duži put od horizontalnog ili vertikalnog kretanja. Kako bi se to uzelo u obzir, uzima se da je cijena horizontalnih i vertikalnog kretanja 1 dok se za

dijagonalno kretanje uzima faktor 1.4. Računaju se dvije cijene kretanja, G i H. G cijena je udaljenost od početne pozicije A dok je H cijena udaljenost nekog polja od odredišne pozicije B. H cijena je zračna udaljenost od točke A do točke B što u prikazanom slučaju na slici iznosi 3.8 (ili 38 radi jednostavnosti prikaza) [5]. Za realizaciju A* algoritma potrebno je uvesti novu cijenu, cijenu F, koja uzima u obzir G i H cijenu tako što ih jednostavno zbraja. Kretanje po polju se vrši tako da se uvijek kreće prema polju koje ima najmanju F cijenu. Nakon svakog koraka, ponovno se računaju potrebne vrijednosti te se taj proces izvodi sve dok se ne dođe do odredišne točke [2].



Slika br. 2: F cijene polja u radu A algoritma*

Prepreke

U kreiranoj igri, kao i kod igara *XCOM* serijala, postoje dvije vrste prepreka:

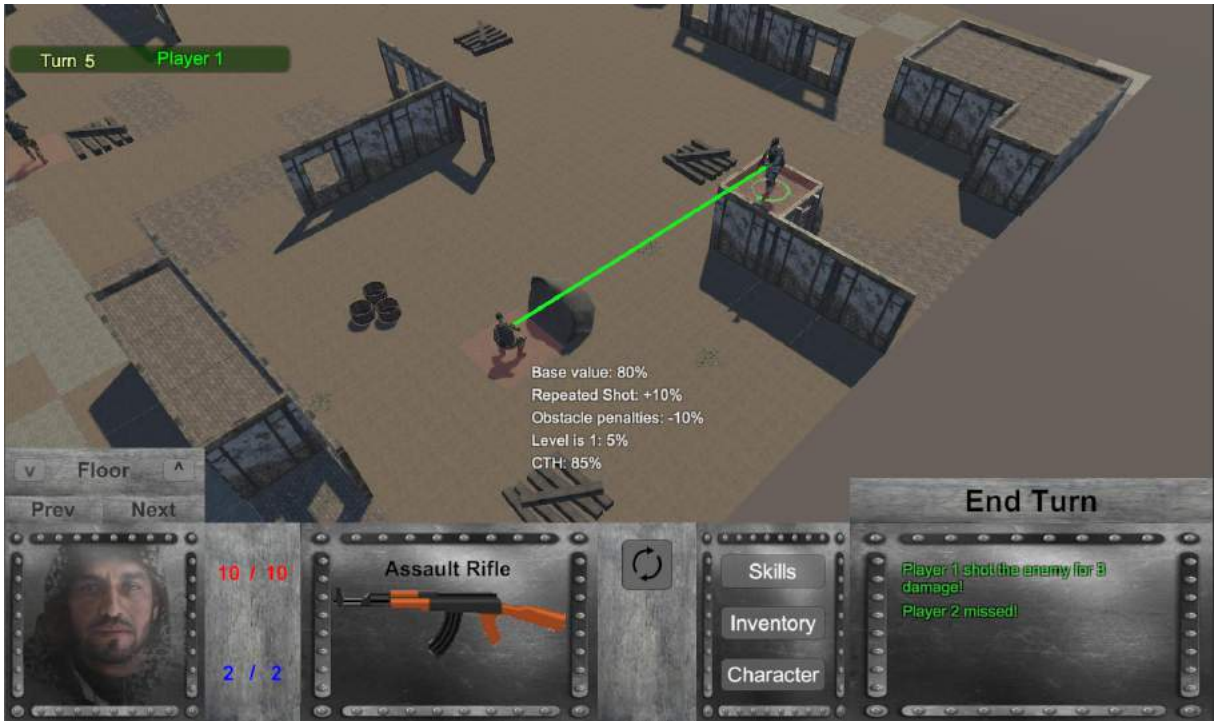
- Potpuna zaštita – to je vrsta prepreke koja potpuno štiti igrača od paljbe neprijatelja i kroz koju se ne može pucati
- Djelomična zaštita – to je vrsta prepreke koja djelomično štiti igrača od paljbe neprijatelj no igrač također iz te pozicije može pucati na neprijatelja

Zidovi su posebna vrsta prepreka koje ne zauzimaju polje već se nalaze na granici između polja. Također postoje dva osnova tipa zidova: zid koji djeluje poput potpune zaštite i zid koji ima funkciju djelomične zaštite. Primjerice, zid djelomične zaštite bi bio zid koji ima prozor u gornjem djelu konstrukcije. Kod izračuna može li se nanijeti šteta na određenom polju i koliko ona iznosi, koriste se informacije polja koje su zapisane u svakom pojedinom polju. U kreiranoj igri, prepreke umanjuju šansu za pogodak za 30%. U slučaju da postoji više prepreka u vidnoj liniji između neprijatelja i igrača, tada se uzimaju u obzir sve prepreke te se radi adekvatni izračun.

Slika br. 3: Izgled djelomičnih prepreka

Oružja

Za glavnu mehaniku borbe, oružja su presudan faktor. U kreiranom prototipu implementirana su tri osnovna tipa oružja: za blisku, srednju i daleku borbu. Prilikom računanja šanse za pogodak uzima se u obzir nekoliko faktora: vrsta oružja koje koristimo, udaljenost neprijatelja i prepreke koje se nalaze u vidnom polju između igrača i neprijatelja. Oružje za blisku borbu ima veliku šansu pogotka no efektivna udaljenost je svega 3 polja. S udaljenošću šansa za pogodak opada.



Slika br. 4: Prikaz šanse za pogodak

Također, jedan od faktora kod izračuna konačne štete je i dio tijela koji je pogođen. Tijelo likova u igri se dijeli u tri osnovna dijela: glava, trup i udovi. Pogodak u glavu izaziva najveću štetu i naziva se kritični udarac (eng. c*ritical hit*). Pogodak u trup čini standardnu štetu jer je trup najlakše pogoditi.

Pogodak u neki ud simulira situaciju gdje se nanosi manje štete od očekivanog te predstavlja napad djelomične štete. Svi ovi izračuni se temelje na osnovnim statističkim principima gdje se generiraju nasumične vrijednosti na temelju postotaka kako bi se odredio konačan ishod borbe.

Zaključak

Iako igra djeluje jednostavno, spomenute mehanike nisu jednostavne za implementaciju. S obzirom na veliku varijabilnost interakcije raznih aspekata, vrlo je velika šansa da će se u borbi pojaviti razni posebni slučajevi koje mogu dovesti do neželjenih rezultata. Svaku mehaniku bitno je pomno dobro dizajnirati i radi adekvatne implementacije igre i radi balansiranosti težine igre.

Reference

[1] Ranker: The Best Turn-Based Strategy Games of All Time. (30. kolovoza 2019.). Dostupno na: https://www.ranker.com/list/best-turn-based-strategy-games-all-time/reference

[2] Sebastian Lague: A* Pathfinding (E01: algorithm explanation) (YouTube video). (30. kolovoza 2019.). Dostupno na: https://www.youtube.com/watch?v=-L-WgKMFuhE

[3] Wikipedia: A* search algorithm. (30. kolovoza 2019.). Dostupno na: https://en.wikipedia.org/wiki/A*_search_algorithm

[4] Wikipedia: Chess. (30. kolovoza 2019.). Dostupno na: https://en.wikipedia.org/wiki/Chess

[5] Wikipedia: Heuristic. (30. kolovoza 2019.). Dostupno na: https://en.wikipedia.org/wiki/Heuristic

[6] Wikipedia: List of turn-based strategy video games. (30. kolovoza 2019.). Dostupno na: https://en.wikipedia.org/wiki/List_of_turn-based_strategy_video_games

[7] Wikipedia: Strategy game. (30. kolovoza 2019.). Dostupno na: https://en.wikipedia.org/wiki/Strategy_game

[8] Wikipedai: Turn-based strategy. (30. kolovoza 2019.). Dostupno na: https://en.wikipedia.org/wiki/Turn-based_strategy

[9] Wikipedia: UFO: Enemy Unknown. (30. kolovza 2019.). Dostupno na: https://en.wikipedia.org/wiki/UFO:_Enemy_Unknown

[10] XCOM: Enemy Unknown. (30. kolovoza 2019.). Dostupno na: https://xcom.fandom.com/wiki/XCOM:_Enemy_Unknown

Žanrovi računalnih igara

Antonio Brnada i Mario Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
antbrnada@foi.hr, mario.konecki@foi.hr

Sažetak

Razvoj računalnih igara traje već preko 50 godina. Svakim danom oduševljenje ljudi računalnim igrama kumulativno raste, uključujući sve uzraste. Granice u određivanju kojem žanru ili žanrovima igra pripada vrlo su fleksibilne jer je za svaku igru potrebno izvršiti višekriterijsku analizu kako bi se došlo do željenog rezultata svrstavanja.

Brz razvoj tehnologije stvara dobru podlogu za razvoj novih žanrova, a uz tako brz razvoj kombinacije žanrova postaju sve zastupljenije jer se povećava kompleksnost samih igara.

Ključne riječi: žanrovi, računalne igre, klasifikacija žanrova, primjeri žanrova

Uvod

Početak razvoja računalnih igara bio je skroman, te je bilo teško tvrditi postojanje većeg broja žanrova. Gledajući rezultate danas, moguće je vidjeti kako računalne igre postaju milijunski poslovi, vođeni ne samo razvojem već i istraživanjem, analizama, ciljanim skupinama, marketingom itd. Takav ubrzani razvoj doveo je do velikog broja ideja i diferencija između računalnih igara. To nas dovodi do podataka koji su od značaja za područje žanrova računalnih igara.

Klasifikacija računalne igre u određeni žanr najčešće se vrši po osnovi mehanike igre, a vizualni dijelovi, kao i prostor i vrijeme u kojem se radnja izvodi, su manje bitni za određivanje žanra igre. Klasifikacija igara ponajviše je bitna krajnjim korisnicima – igračima, ali i izdavačima računalnih igara.

Tako organizirani sustav služi pri odluci o razvoju sljedeće igre. Praćenje trendova među korisnicima može dovesti do ideje, ovisno o popularnosti žarna u nekom periodu. Korisniku će s druge strane žarnovi pomoći donijeti odluku pri odabiru nove igre kojoj bi posvetio svoje vrijeme.

Definicija računalne igre i žanrova računalnih igara

Pojam računalne igre često je izjednačen s pojmom videoigre, te takva definicija glasi: „Računalne igre (videoigre), primjenski računalni programi za zabavu. U pravilu su to interaktivne igre koje se odvijaju na osobnim računalima, specijaliziranim računalima (tzv. igraće konzole), prijenosnim (džepnim) igraćim konzolama, te na dlanovnicima, mobitelima i sličnomu“ [5].

Kako bi se što ispravnije predstavilo odgovarajuće primjere igara, potrebno je odrediti granice prema kojima igra pripada kategoriji računalnih igara. U sljedećih nekoliko rečenica bit će prikazana detaljnija razlika između pojmova računalne igre i videoigre.

„Pojam računalne igre obično se poistovjećuje s pojmom videoigre. Zapravo, među njima ne postoji značajnija razlika, osim što se računalne igre igraju pomoću računala, a videoigre pomoću konzole (npr. Playstation) koja je priključena na televizor“ [1].

Time dolazimo do pojma žanrova računalnih igara. Žanrovi se određuju po osnovi mehanike igre, ali nekad postoje i izuzeci, npr. računalna igra čija je mehanika igre kombinacija dvaju žanrova [4].

Suvremena klasifikacija žanrova računalnih igara

Valve Corporation je danas jedan od najvećih digitalnih distributera računalnih igara. Njihov rad krenuo je 2003. s njihovom platformom pod nazivom *Steam* koja danas ima više od 67 milijuna korisnika svaki mjesec. Korisnici mogu birati između više od 10 000 dostupnih igara koje su najvećim dijelom računalne igre [2].

Jedna od važnih usluga koju *Steam* platforma mora pružati svojim korisnicima je sortiranje i filtriranje ukupnog sadržaja (igri) po žanrovima. Već je navedeno kako granice klasifikacije nisu striktne i kako postoje razna odstupanja koja otežavaju točnu klasifikaciju žanrova. Daljnja razrada klasifikacije dovodi do takozvanih oznaka (engl. *Tags*) koje platforma *Steam* koristi za detaljniju specifikaciju igre, a koja se temelji na proširenju nekog od početnih žanrova.

Deset najvažnijih žanrova prema *Steam-u* su:

- Akcijske igre (engl. *Action)*,
- Avanturističke igre (engl. *Adventure*),
- Jednostavne igre / Usputne igre (engl. *Casual*),
- Indie igre (engl. *Indie*),

- Internetska igra s velikim brojem igrača (engl. *Massively Multiplayer Online - MMO)*
- Trkaće igre (engl. *Racing*),
- Igra igranja uloga (engl. *Role-playing game – RPG*)
- Simulacije (engl. *Simulation*),
- Sportske igre (engl. *Sports*),
- Strateške igre (engl. *Strategy*)

Osim navedenog, koristi se gotovo 400 oznaka kako bi se moglo doći do što specifičnijih rezultata prilikom pretraga [8].

Battle-Royale je primjer novog žanra koji se pojavio u 2017. i koji je uvelike utjecao na razvoj igara. Generalizacijom ovog žanra došlo bi se do žanra akcijskih igara, ali zbog kompleksnosti igre dosta često se predstavlja kao zasebna vrsta žanra.

Akcijske igre

Akcijske igre imaju jedan od najstarijih početaka među računalnim igrama i predstavljaju jedan od žanrova iz kojeg je proizašlo najviše pod-žanrova, poput *FPS* igri, *Battle Royale*, igri na 3D platformama i drugih. Mehanika igre je malo drugačija kod svakog pod-žanra, ali ono što im je zajedničko je izazivanje korisnikovih refleksa, koordinacije i brzine reagiranja [7].

Slika br. 1: Prikaz igre „Playerunknown's Battlegrounds“

Kao primjer bit će prikazana jedna *Battle Royale* igra koja je svojim izlaskom 2017. pridobila jako veliki broj korisnika. *Playerunknown's Battlegrounds*

(slika 1), kraće PUBG, izdana od strane poduzeća PUBG Corporation, je igra u kojoj sudjeluje po 100 igrača kojima je cilj preživjeti na jednom otoku, te ubiti sve protivnike. Potrebno je sakupljati razne stvari u okolini koje igraču omogućavaju preživljavanje i koje vode igrača do bolje opreme i prednosti nad protivnikom. Platforma *Steam* ovu igru klasificira u tri žanra – akcija, avantura i igra s masovnim brojem igrača preko Interneta. Ipak prema glavnim žanrovima, ova igra i dalje prvobitno pripada akcijskim igrama [8].

Avanturističke igre

Avanturističke igre uglavnom prate glavnog protagonista koji prolazi kroz razne zapreke i zagonetke, istražujući okolinu oko sebe kako bi napredovao do sljedećeg cilja. Često je prisutan i pripovjedač koji detaljnije opisuje zadatke i priču. Avanture mogu biti i realne, ali mogu uključivati i znanstvenu-fantastiku, razne misterije, čak i neke mitove.

Ovaj žanr bio je vrlo popularan u ranijim godinama, dok se danas njegova zastupljenost smanjuje i kompenzira kroz kombinaciju s drugim žanrovima kao što je npr. akcijsko-avanturistička igra [7].

A way out (slika 2) je akcijsko-avanturistička igra razvijena od strane poduzeća *Electronic Arts*. Za igranje su potrebna dva igrača koja mogu surađivati lokalno ili online. Priča prati dvojicu prijatelja koji moraju pronaći svoj put iz zatvora. Potrebno je donositi odluke koje utječu na daljnji ishod igre, istraživati i rješavati probleme kako bi se došlo do sljedeće faze igre [3].

Slika br. 2: Prikaz igre „A way out“

Internetska igra s velikim brojem igrača (MMO)

MMO žanr je svoj veliki uspjeh doživio nakon popularizacije Interneta jer je omogućavao velikom broju igrača da igraju istovremeno u nekom virtualnom svijetu. Rijetko postoji situacija u kojoj se neka računalna igra može svrstati samo u MMO žanr, u većini slučajeva igra je obilježena pomoću pod-žanrova poput *Battle Royale* koji sadrži segmente akcije i MMO žanra. Vrlo popularan pod-žanr koji je nastao iz ovog žanra je igra igranja uloga putem Interneta s velikim brojem igrača (eng. *Massive Multiplayer Online Role-playing Game – MMORPG*).

Kao primjer može se navesti igra *Rust* koja sadrži elemente preživljavanja, akcijskih i avanturističkih igara i ono najvažnije, igra ne bi bila moguća bez glavnih elemenata MMO žanra. *Rust* (slika 3) je predstavljen 2018. od strane poduzeća Facepunch Studios i danas je među 10 najigranijih igara na platformi *Steam*.

Cilj igre je preživljavanje u svijetu s velikim brojem ljudi. Potrebno je paziti na osnovne životne potrebe poput gladi i žeđi, graditi skloništa, kreirati nove stvari, loviti životinje i sukobljavati se s drugim igračima. U takvom virtualnom svijetu moguće je razviti i oblike suradnje s drugim igračima, te udružiti snage u svrhu preživljavanja [8].

Rust je jedan od primjera suvremenih računalnih igara koje koriste elemente iz više žanrova da bi se dobio željeni rezultat igre. Samim time teško je jednoznačno odrediti kojem žanru igra pripada.

Slika br. 3: Prikaz igre „Rust“

Budućnost žanrova računalnih igara

Uvođenje novih tehnologija na tržište računalnih igara dovodi do novih oblika razvoja igara, a samim time i novih žanrova. Prije nekoliko godina predstavljen je koncept virtualne stvarnosti (engl. *Virtual Reality – VR*). Ova tehnologija predstavljena je na konzolama, ali i na osobnim računalima.

Razvoj VR igara za sada se zasniva na već postojećim žanrovima, kreiraju se VR simulacije, VR akcijske igre, VR strateške igre, uz korištenje raznih motiva. Ipak, razmišlja se o tome da VR igre postanu zaseban žanr unatoč tome što mehanike VR igara ne moraju biti iste ili slične [7].

Može se vidjeti preokret u razvoju koji je donijelo uvođenje žanra *Battle Royale* u industriju računalnih igara. Veliki broj novih igara predstavljen je baš u tom žanru, mnoge postojeće igre su modifikacijama i prepravcima uvele mogućnost *Battle Royale* perspektive. Nekoliko godina prije izlaska ovog žanra na tržište nitko nije predviđao takav uspjeh.

Prema Hodentu, teško je pretpostavljati šta nas očekuje u budućnosti, ali on vjeruje kako će postojati nove platforme koje još ne možemo pojmiti [6].

Zaključak

Prilikom klasifikacije žanrova može se primijetiti kako ne postoji unificirana verzija popisa žanrova, već se taj popis mijenja kroz vrijeme. Neki žanrovi nestaju, a neki tek dobivaju na popularnosti. Sve više se javlja potreba za standardiziranom listom žanrova koja bi pomogla ne samo igračima već i mnogim digitalnim platformama poput *Steam-a*.

Računalne igre su danas često kombinacija dvaju ili više žanrova, te ih je kao takve teško okarakterizirati, osim ako se promatra dominantniji žanr. Kroz budućnost razvoja igara igrače očekuje još više ispreplitanja žanrova koji u konačnici mogu dovesti i do potpuno novih žanrova kako bi se opisao točno takav oblik igre. Tehnologije poput VR-a potpomažu ovakav razvoj događaja.

Korisnicima tj. igračima i dalje ostaje zadovoljstvo i užitak igranja računalnih igara, pri čemu svatko može pronaći igru prema svojih preferencijama. Na kraju, svi su igrači na neki način različiti i sve dok ta razlika postoji kreirat će se mnoštvo žanrova kako bi se u konačnici našla odgovarajuća igra za svakoga.

Reference

[1] Bilić, V., Gjukić, D., Kirinić, G. (2010). Mogući učinci igranja računalnih igrica i videoigara na djecu i adolescente. Napredak: časopis za pedagogijsku teoriju i praksu, 151(2), pp. 95-213.

[2] CNET u suradnji s John Lewisom (bez dat.) The History of PC Gaming Preuzeto 29.05.2019. s http://paidcontent.cnet.co.uk/the-history-of-pc-gaming/?ftag=ACQ-07-10abc7d

[3] Electronic Arts (2018). A way out. Preuzeto 30.8.2019. s https://www.ea.com/games/a-way-out

[4] Fengfeng Ke, (2009). A Qualitative Meta-Analysis of Computer Games as Learning Tools Preuzeto 25.08.2019. s https://www.igi-global.com/chapter/qualitative-meta-analysis-computer-games/20076

[5] Računalne igre. (bez dat.). U Hrvatska enciklopedija. Preuzeto 25.08.2019. s http://www.enciklopedija.hr/natuknica.aspx?id=68642

[6] Stafford P. (2017). What will the game industry look like in five years? Preuzeto 30.08.2019. s https://www.polygon.com/features/2017/11/14/16533054/the-game-industry-five-years-2022

[7] Technavio (2018). Top 10 most popular game genres in the world 2018. Preuzeto 28.08.2019. s https://blog.technavio.com/blog/top-10-most-popular-game-genres

[8] Valve Corporation (2019). Steam. Preuzeto 30.8.2019. s https://store.steampowered.com

E-sport: usporedba s tradicionalnim sportom i mogućnost razvoja ove vrste sporta u studentskoj populaciji

Sanja Ćurković, Mario Konecki i Damir Vučić
Ured za sport, Sveučilište u Zagrebu
Fakultet organizacije i informatike, Sveučilište u Zagrebu
Fakultet organizacije i informatike, Sveučilište u Zagrebu
sanja.curkovic@gmail.com, mario.konecki@foi.hr, damir.vucic@foi.hr

Sažetak

E-sport polako ali sigurno postaje sve prepoznatiji kao prava i službena sportska disciplina. Sve veći broj igrača prati ili sudjeluje na raznim e-sport natjecanjima. U ovom radu dan je osvrt na definiciju e-sporta, razvoj e-sporta, usporedbu e-sporta s tradicionalnim sportom, a dan je i primjer dobre prakse u području organiziranje e-sporta na Sveučilištu u Zagrebu.

Ključne riječi: e-sport, natjecanja, turniri, studenti, Sveučilište u Zagrebu, Ured za sport Sveučilišta u Zagrebu

Uvod

Razvoj Interneta i informatičke tehnologije (IT) ubrzao je popularnost interaktivne digitalne komunikacije. Popularnost videoigara i e-sporta raste nevjerojatnom brzinom, osobito nakon pojave streaming platformi poput Panda TV-a, YouTubea i Twitcha. Veliki međunarodni turniri koji se organiziraju zadnjih 20-ak godina iz godine u godinu privlače sve više igrača i gledatelja. E-Sport je relativno novo područje u kulturi igranja, no ono postaje jedan od najvažnijih i najpopularnijih dijelova zajednice videoigara, osobito među adolescentima i mladima.

E-Sport se odnosi na organizirana natjecanja u videoigrama, najčešće u kontekstu organiziranih turnira [11]. Obuhvaća niz platformi, od osobnih računala do igraćih konzola i niz žanrova, uključujući igre sportske tematike i igre u stvarnom vremenu (RTS) koje ujedno predstavljaju dva najpopularnija žanra e-sporta [19]. Ovo područje je uznapredovalo tijekom posljednjeg desetljeća te je sve veći broj natjecanja i događanja koja privlače gledatelje širom svijeta [13].

Tržište obiluje sve većim brojem e-sportskih natjecanja i povezanih događaja čija popularnost vrtoglavo raste. Međutim, treba istaknuti da je ovo područje u

većini Zapadnih zemalja u ekspanziji, a već je pronašlo svoje mjesto u kulturi nekih istočnoazijskih zemalja [15]. Primjerice, u Južnoj Koreji e-sportom upravlja nacionalna udruga (Korean eSports Asssocation; KeSPA) koja certificira profesionalne igrače, daje rang liste i organizira natjecanja [12], pri čemu se može vidjeti da tradicionalni sport i e-sport djeluju u sinergiji i uvažavanju. Velike sličnosti e-sporta s tradicionalnim sportom dovele su do toga da je Olimpijsko vijeće Azije (OCA) objavilo da će e-sport postati službeni sport s medaljama na Azijskim igrama 2022. godine, što predstavlja važno priznanje e-sportu [6].

S obzirom na sve veći broj e-sport natjecanja i povezanih događanja, sve je veći interes medija, korporativnih sponzora ali i znanstvenika za ovo područje. Posljednjih godina e-sport je sve popularniji u akademskoj zajednici kako u studentskoj populaciji tako i među znanstvenicima, što je vidljivo objavom sve većeg broja radova na teme koje su povezane s e-sportom. Cilj ovog rada je prikazati usporedbu e-sporta s tradicionalnim sportovima te prikazati primjerom kako je e-sport pronašao svoje mjesto u sveučilišnom sportu.

Sličnosti i razlike između e-sporta i tradicionalnih sportova

Bez obzira na to što je u fokusu istraživača, e-sport ipak predstavlja manje istraženo područje. Još uvijek postoji nedosljednost u definiranju e-sporta, premda je generalno dogovoreno da se ovaj izraz odnosi na strukturirano, računalno upravljano i natjecateljsko višestruko igranje igara koje uključuje i publiku [5]. Iako se razlikuje od tradicionalnih ekipnih sportova u realnom prostoru, e-sport se oslanja na timski rad koji uključuje koncentraciju, koordinaciju, komunikaciju i povezanost.

Bez obzira na to što pojedini igrači mogu postati poznati i privući veliku publiku, ulagači koji traže mogućnosti sponzorstva često se okreću timovima. Igre u e-sportu, temeljene na tradicionalnom sportu, također su stekle popularnost posljednjih godina jer se veći broj izdavača videoigara udružuje s tradicionalnim sportskim franšizama u razvoju sezonskih turnira i e-sport reprezentacija.

Da bi igrači postigli vrhunsku izvedbu moraju biti odlično pripremljeni (jednako kao i u tradicionalnim sportovima) kako bi mogli izvoditi ozbiljne i precizne pokrete ruku, obrađivati velike količine informacija i učinkovito surađivati s članovima svog tima. Za e-sport timove timski rad se zasniva na zadatku - slično kao i u tradicionalnom organizacijskom okruženju. S druge strane, e-sport timski rad događa se u visoko konkurentnoj, stresnoj i intenzivnoj virtualnoj okolini koja zahtijeva brzo donošenje odluka i odgovora povezanih s fizičkim aktivnostima orijentiranim na akciju. Stoga e-sport nudi vrijednu i jedinstvenu priliku za daljnje istraživanje društvenih i

organizacijskih procesa formiranja tima i koordinacije tima. Timovi se smatraju korisnim za izvršavanje složenih zadataka zbog svoje sposobnosti dopuštanja članovima tima da razmjenjuju radno opterećenje, prate radna ponašanja drugih, potiču i doprinose ekspertizi o podskupovima [16]. Ipak, glavni izazov u procesu stvaranja učinkovitog tima odnosi se na odabir članova tima koji mogu pomoći postizanju visoke razine izvedbe [17].

Temeljem sličnog koncepta slažu se i ekipe u sportskim igrama, uvažavajući vještine svakog pojedinca koji se usavršava na jednoj ili više igračkih pozicija, no mora biti u sinergiji sa svim ostalim igračima. Općenito gledano, u ekipnom sportu naglašeno je pitanje motivacije i timskog rada. To su faktori koje treba uzeti u obzir pri raspravljanju o karakteristikama natjecateljskog e-sporta.

Postojeće definicije suvremenog sporta naglašavaju njegove kondicijske, natjecateljske i institucionalizirane dimenzije [7]. Pitanje koliko e-sport ispunjava ove kriterije i može li se zvati sportom u fokusu je brojnih istraživanja i izložen je brojnim stručnim raspravama [8; 9; 10; 11; 12]. Najveće zamjerke idu u smjeru činjenice da u e-sportu izostaje kondicijska komponenta [12].

Utvrđena značajka koja razlikuje igru od sporta je upravo kondicijska spremnost [3; 20]. Neki znanstvenici navode da e-sportu nedostaje upravo taj segment kondicijske spremnosti kako bi se mogao smatrati pravim sportom [11]. Ipak, zagovornici e-sporta sugeriraju da e-sport dijeli mnoga od glavnih obilježja tradicionalnog sporta. Zagovornici tvrde da e-sport uključuje međuljudsku konkurenciju, obuku i razvoj vještina, pridržavanje pravila, postizanje ciljeva, koordinaciju i spretnost [4; 10].

Rasprava o e-sportu kao sportu daje polazište za dodatna istraživanja i potrebu novog pristupa za unaprjeđivanje diskusije na ovu temu. Istraživači pokušavaju utvrditi u kojoj se mjeri uočava sličnost s tradicionalnim sportom s obzirom na prirodu aktivnosti i funkcioniranje poslovnog modela. U raspravama o e-sportu se ponekad koriste i argumenti da sličnosti s tradicionalnim oblikom sporta ne postoje samo u prirodi natjecanja, već i u obliku organiziranih događanja, od prosudbi, komentara, publike uživo, emitiranja, do novčanih nagrada za igrače. Rasprava o tome treba li e-sport klasificirati kao profesionalni sport traje već neko vrijeme [11]. Nesuglasice se uglavnom odnose na nedostatak pokreta velikih mišićnih skupina kao i kondicijske pripremljenosti te uspostave institucionalne povezanosti.

S druge strane, može se postaviti pitanje što je onda s drugim sportovima, primjerice šahom, automobilizmom, moto utrkama, biljarom, pikadom i sličnim sportovima koji također nemaju kondicijsku dimenziju izraženu kao neki drugi sportovi u kojima je ta komponenta naglašena. U svakom slučaju još

će se raspravljati na ovu temu, no činjenice o popularnosti e-sporta doprinose rušenju barijera i predrasuda o e-sportu. Kreiranje igara koje sve više sliče tradicionalnom sportu pridonose rušenju barijera realnog i virtualnog prostora. Tako igre sa sportskom tematikom oponašaju tradicionalni sport u njegovom kondicijskom (fizičkom) djelu, dok su RTS igre reprezentacija borbenih ili vojnih borbi s izraženim vizualnim i prostornim koordinacijskim parametrima, koji su prisutni i u tradicionalnim sportovima [2].

Slično kao i u tradicionalnom sportu, e-sport sadrži usporedne mjere za procjenu igračeve izvedbe u igri (Seo, 2013) te se tako mogu uspoređivati s tradicionalnim sportom kao što je nogomet i postizanje golova u nogometnoj utakmici [19]. U RTS e-sportu, pojedinačni igrač kontrolira čitavu virtualnu vojsku koja se promatra iz zračne perspektive s igračima koji su usredotočeni na poraz svojih protivnika [1] putem niza mogućih ciljeva, najčešće uništavajući strukture i jedinice protivnika. Za ovakve akcije je potrebna iznimna prostorno-vizualna orijentacija, koordinacija ruka-oko, situacijsko rješavanje zadataka i čitav niz tjelesnih i kognitivnih funkcija koje su potrebne i u tradicionalnim sportovima.

Razvoj e-sporta na Sveučilištu u Zagrebu

Zagrebački sveučilišni sportski savez i Ured za sport Sveučilišta u Zagrebu, uz suradnju sa Studentskim zborom Sveučilišta u Zagrebu po prvi puta je pokrenuo e-sport akademsko natjecanje u Republici Hrvatskoj. Prateći svjetske trendove, razvoj e-sporta je sada na Sveučilištu u Zagreb uz bok nekolicini sveučilišta u Europi koja su također pokrenula svoja službena e-sport natjecanja. Uz navedeno, otvara se i mogućnost uvođenja novih znanstvenih disciplina koje su popratne u razvoju e-sport industrije, kao što su Psihologija e-sporta, Modeli poslovanja u e-sportu, Tehnologije e-sporta, Programiranje računalnih igara i slično. U procesu popularizacije ovog sporta pokrenule bi se tribine i konferencije vezane za teme e-sporta kao i škole „igranja“ na kojima bi sudjelovali svi studenti Sveučilišta u Zagrebu koji bi željeli unaprijediti svoje vještine ili naučiti nove vještine koje do sada nisu imali.

Temeljem svega ranije navedenog, Ured za sport Sveučilišta u Zagrebu i zagrebački sveučilišni sportski savez uvrstio je e-sport u svoj kalendar natjecanja u akademskoj godini 2017./2018. pruživši priliku da se kao eksperimentalni sport u sustavu UniSportZG razvije u studentskoj populaciji. E-sport tako i formalno stiže na Sveučilište u Zagrebu. Iako on tu postoji već dugi niz godina, studenti su dobili priliku natjecati se u službenoj ligi. Upravo su studenti veliki dio e-sport industrije kao igrači, ali i kao publika. Ovakav projekt jedan je od ključnih preduvjeta važnih za razvoj e-sporta u Hrvatskoj, kao i za podizanje svijesti o istom. Neke od najboljih uzdanica domaćeg

e-sporta nastajale su upravo u skučenim sobama studentskih domova, a sada se ovi isti studenti mogu natjecati sa svojim kolegama u za to posebno opremljenom prostoru.

Slika br. 1: E-sport na Sveučilištu u Zagrebu

Ured za sport Sveučilišta u Zagrebu i Zagrebački sveučilišni sportski savez u okviru sustava UniSportZG planira organizira i studentski sport u Gradu Zagrebu i Zagrebačkoj županiji. Kroz sustav UniSportZG koji obuhvaća gotovo šest tisuća studenata organiziraju se natjecanja u 30 različitih sportova što Zagrebački sveučilišni sportski savez i studente uključene u e-sport stavlja na prvo mjesto ne samo u Hrvatskoj i regiji, nego i u Europi po brojnosti i rezultatima. Uz dosadašnje lige u sportovima kao što su futsal, košarka i rukomet, održava se i e-sport s čak četiri igre: League of Legends, Counter-Strike: Global Offensive, Pro Evolution Soccer 2018 i Hearthstone.

Osim kao natjecatelji, svi zainteresirani studenti, mogu sudjelovati u organizaciji samog natjecanja kako bi stekli dodatna znanja i iskustvo za potencijalni rad u e-sportu ili industriji videoigara. Do sada su održana dva Sveučilišna studentska prvenstva, u akademskoj godini 2017./2018. i 2018./2019. Prvo natjecanje održano je u Stablu znanja u Studentskom centru koje je za tu prigodu bilo posebno opremljeno. Osim za natjecanja, posebno opremljen prostor studenti su mogli koristiti i za vježbu i trening. Natjecanje se održavalo subotom i nedjeljom. Prvenstvo se sastojalo od četiri kategorije,

tri timske i jedne pojedinačne. U League of Legendsu i CS:GO-u natjecali su se timovi s pet igrača, PES 2018 igrao se u parovima, a Hearthstone pojedinačno.
U akademskoj godini 2018./2019. godini održano je drugo po redu službeno natjecanje u e-sportu. Partner u provedbi natjecanja bio je Hangar 18 d.o.o. koji je osigurao infrastrukturu za provođenje natjecanja. Sudjelovalo je ukupno 211 studentica i studenata (PES 21, FIFA 379, Hearthstone 30, CS 40 LOL 83) s 18 različitih visokih učilišta. Natjecanje je organizirano u pet igara: League of Legends, Hearthstone, Counterstrike, Pro Evolution Soccer i FIFA 19. Pro Evolution Soccer i FIFA 19 su se odigravali na konzolama Playstation 4, dok su se ostale igre odigravale na računalima.

Slika br. 2: E-sport na Sveučilištu u Zagrebu

Pobjednik u Pro Evolution Soccer natjecanju bile je ekipa FER-a, dok je drugo mjesto zauzela ekipa FOI-ja. Treće plasirana ekipa bila je ekipa Medicinskog fakulteta, dok je na četvrtom mjestu završilo Tehničko veleučilište Zagreb (TVZ). Ukupno je nastupilo 9 ekipa. Pobjednik u FIFA 19 natjecanju bila je ekipa Prehrambeno-biotehnološkog fakulteta (PBF), dok je drugo mjesto zauzela ekipa Prirodoslovno-matematičkog fakulteta (PMF). Treće plasirana bila je ekipa Veterinarskog fakulteta, dok je četvrti završio Fakultet strojarstva i brodogradnje (FSB). Ukupno je nastupilo 14 ekipa. Hearthstone je osvojila ekipa FER-a, drugo plasirana je bila ekipa FSB-a, dok je treće mjesto zauzeo PMF. Četvrto mjesto zauzeo je Građevinski fakultet. Ukupno je nastupilo 9 ekipa.

League of Legends natjecanje je osvojila ekipa FOI-ja koja je u finalu pobijedila ekipu TVZ-a, dok je treće mjesto zauzeo PMF, koji je porazio FER. Ukupno je sudjelovalo 14 ekipa. Counter strike: Global offensive natjecanje je osvojila ekipa FER-a, dok je drugo mjesto zauzeo FSB. Treće mjesto je zauzeo TVZ, kojem je ekipa FOI-ja predala susret. Ukupno je sudjelovalo 8 ekipa. Sva natjecanja u igrama League of Legends, Counterstrike i Hearthston popraćena su streamingom na Twitch kanalu UniSportZg, na kojemu je zabilježeno preko 4000 pregleda za vrijeme trajanja sezone.

Zaključak

Temeljem ranije iznesenih činjenica o e-sportu i tradicionalnim sportovima može se zaključiti da sličnosti ne postoje samo u prirodi natjecanja već i u obliku organiziranih događanja, od prosudbi, komentara, publike uživo, emitiranja, do novčanih nagrada za igrače. S druge strane, pojedina istraživanja ukazuju na nedostatke ističući da e-sportu nedostaju kondicijski angažman i odgovarajuća struktura upravljanja.

U ovom radu dan je osvrt na e-sport s kineziološkog aspekta i prikazan je primjer dobre prakse na Sveučilištu u Zagrebu u organizaciji, provođenju i promoviranju e-sporta u studentskoj populaciji. Ovakvim primjerima uvelike se može pomoći razvoju e-sporta u Republici Hrvatskoj, osobito ako i druga Sveučilišta pokrenu ovakva natjecanja. Time bi se stvorila organizacijska struktura koja bi se formirala u nacionalnu akademsku organizaciju e-sporta.

Reference

[1] Buchanan-Oliver, M., & Seo, Y. (2012). Play as co-created narrative in computer game consumption: The hero's journey in Warcraft III. Journal of Consumer Behaviour, 11(6), pp. 423–431.

[2] Burk, D. L. (2013). Owning e-Sports: Proprietary rights in professional computer gaming. University of Pennsylvania Law Review, 161(6), pp. 1535–1578.

[3] Coakley, J. (2008). Studying intercollegiate sports: High stakes, low rewards. Journal of Intercollegiate Sport, 1(1), pp. 14–28.

[4] Crawford, G., & Gosling, V. (2009). More than a game: Sportsthemed video games & player narratives. Sociology of Sport Journal, 26(1), pp. 50–66.

[5] Freeman, Guo, and Donghee Yvette Wohn (2017). Social Support in eSports: Building Emotional andEsteem Support from Instrumental Support Interactions in A Highly Competitive Environment. In CHI PLAY '17.Proceedings of the 2017 Annual Symposium on Computer-Human Interaction inPlay,Amsterdam, Netherlands, 15–18 October 2017. New York: ACM Press, pp. 435-447.

[6] Graham, Bryan. (2017). eSports to Be A Medal Event at 2022 Asian Games. News Article. TheGuardian https://www.theguardian.com/sport/2017/apr/18/esports-to-be-medal-sport-at-2022-asian-games. Posjećeno 1. rujna 2019.
[7] Guttmann, A. (2004). From ritual to record: The nature of modern sports. New York: Columbia University Press.
[8] Hallmann, K., & Giel, T. (2018). eSports–Competitive sports or recreational activity? Sport Management Review, 21, pp. 14-20. http://doi.org/10.1016/j.smr.2017.07.011
[9] Hilvoorde, I. & Pot, N. (2016) Embodiment and fundamental motor skills in eSports. Sport, Ethics and Philosophy, 10(1), pp. 14–27.
[10] Holt, J. (2016). Virtual domains for sports and games. Sport, Ethics and Philosophy, 10(1), pp. 1–9.
[11] Jenny, S. E., Manning, R. D., Keiper, M. C., & Olrich, T. W. (2017). Virtual(ly) athletes: Where eSports fit within the definition of "Sport." Quest, 69(1), pp. 1–18.
[12] Jonasson, K., & Thiborg, J. (2010). Electronic sport and its impact on future sport. Sport in Society, 13(2), pp. 287–299.
[13] Kresse, Christian. (2016). eSports in 2015 by the Numbers: Attendance Figures, Investments andPrize Money. eSports Marketing Blog.http://esports-marketing-blog.com/esports-in-2015-atten-dance-figures-investments-prize-money/#.WRxgG1Xyupo.
[14] Korean Culture and Information Service. (2016). How South Korea became a sporting powerhouse. Korea.net. Retrieved from http://www.korea.net/AboutKorea/Sports/ How-South-Korea-Became-Sporting-Powerhouse
[15] Li, R. (2016). Good luck have fun: The rise of eSports. New York: Skyhorse Publishing.
[16] Mathieu, John E.; Tonia S. Heffner; Gerald F. Goodwin; Eduardo Salas; and Janis A. Cannon-Bowers.(2000). The Influence of Shared Mental Models on Team Process and Performance. Journal of Applied Psychology, vol.85, no.2, April 2000, pp. 273-283.
[17] Reagans, Ray; Ezra Zuckerman; and Bill McEvily. (2004). How to Make the Team: Social NetworksVs. Demography as Criteria for Designing Effective Teams. Administrative Science Quarterly,vol.49, no. 1, March 2004, pp. 101-133.
[18] Seo, Y. (2013). Electronic sports: A new marketing landscape of the experience economy. Journal of Marketing Management, 29(13–14), pp. 1542–1560.
[19] Seo, Y., & Jung, S. (2014). Beyond solitary play in computer games: The social practices of eSports. Journal of Consumer Culture, 16(3), 635–655. http://doi.org/10.1177/1469540514553711
[20] Suits, B. (2007). The elements of sport. In Ethics of sport (pp. 9–19). Champaign, IL: Human Kinetics.

Kombinatorne igre

Zvonimir Galić, Mirko Čubrilo i Mario Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
zgalic@foi.hr, mcubrilo@foi.hr, mario.konecki@foi.hr

Sažetak

Teorija kombinatornih igara je grana matematike i teoretskog računarstva koja proučava slijedne igre s potpunom informacijom. Kombinatorne igre su igre za dva ili više igrača koje moraju završiti nakon konačnog broja poteza. Među najpoznatijim kombinatornim igrama su šah, dama, Hackenbush, Nim, Kuglice i kutije (engl. Dots and Boxes) itd. U ovom radu dani su opisi nekih od predstavnika kombinatornih igara, uz navođenje pravila ovih igara, kao i odgovarajućih taktike nužnih za pobjedu.

Ključne riječi: kombinatorne igre, Hackenbush, Nim, Kuglice i kutije

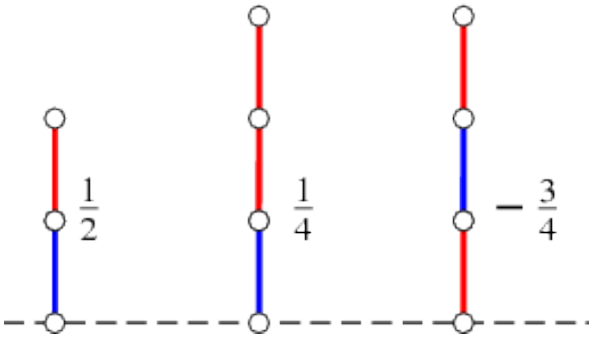
Uvod

Slijedne igre su igre u kojima igrači naizmjence odigravaju svoje poteze. U slijedne igre spadaju šah, Nim, Čovječe ne ljuti se, remi itd. Kartaške igre i igre s kockicama ipak ne spadaju u kombinatorne igre jer nisu igre s potpunom informacijom. Potpuna informacija znači da oba igrača znaju sve svoje i protivnikove poteze u igri. U igri šah oba igrača vide sve figure i sve prošle poteze u igri, ali u kartaškim igrama nedostaje znanje o kartama drugih igrača i nepodijeljenim kartama da bi i one bile igre s potpunom informacijom. Kod većine kombinatornih igara pobjednik se određuje zadnjim potezom u igri. Normalan način određivanja pobjednika znači da je pobjednik onaj tko napravi zadnji potez u igri, odnosno igrač koji ne može napraviti potez je gubitnik. Mizeran način određivanja pobjednika znači suprotno, onaj koji zadnji napravi potez – gubi.

Hackenbush

Igra Hackenbush je kombinatorna igra za dva igrača koja se uglavnom igra na papiru ili ploči za pisanje. Pravila igre su jednostavna. Igrači naizmjence brišu linije određene boje, jedan igrač briše crvene, a drugi igrač briše plave linije. Prvi igrač koji ostane bez linija svoje boje je gubitnik. Sve crvene i plave linije su spojene s tlom, koje je uglavnom duga, crna, isprekidana crta na dnu papira ili ploče. Ako se neki dio lika odvoji od te crte tada se on automatski briše iz

igre. To znači da se pobjednik ne može uvijek odrediti samo brojanjem linija. Recimo da Lijevi igrač igra s plavim linijama, a Desni s crvenim.

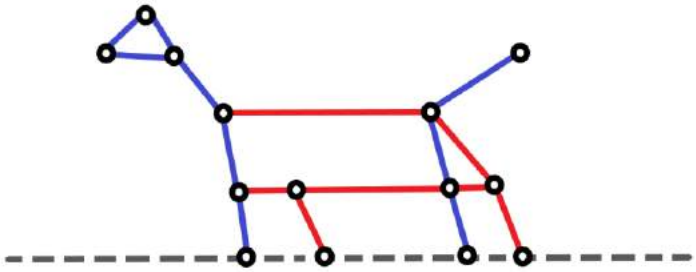


Slika br. 1: Hackenbush

U prvom primjeru i Lijevi i Desni igrač imaju po jednu liniju, ali to ne znači da je igra izjednačena. Ako Lijevi igrač igra prvi tada on mora izbrisati plavu liniju, što automatski briše i crvenu liniju te Lijevi igrač pobjeđuje. Ako Desni igrač igra prvi tada on briše samo crvenu liniju. Lijevi igrač odigra svoj potez, ostavlja Desnog igrača s praznom pločom te Lijevi igrač opet pobjeđuje. To znači da sve linije u ovoj igri ne vrijede isto. Plava linija je povezana s isprekidanom linijom te je Desni igrač nikako ne može ukloniti iz igre. To znači da ona vrijedi jedan potez. Crvena je linija je s crnom linijom povezana preko plave linije te je Lijevi igrač može izbrisati svojim potezom. Stoga vrijedi manje nego plava linija, odnosno vrijedi pola poteza. Vrijednost igre se računa na ovaj način: 1 – ½ = +½. Ako je vrijednost igre pozitivna to znači da prednost ima Lijevi igrač, ako je negativna onda prednost ima Desni igrač. Ako je vrijednost igre nula, onda je igra izjednačena. Nulte igre pobjeđuje onaj koji je drugi na potezu, jer oba igrača imaju isti broj poteza.

U drugom primjeru (slika 1) su dvije crvene linije i samo jedna plava. Ako Lijevi igrač igra prvi, svojim potezom će izbrisati sve linije te ostaviti Desnog igrača bez poteza. Ako Desni igrač igra prvi, neovisno o tome koju liniju briše, Lijevi igrač će moći napraviti svoj potez te opet ostaviti praznu ploču. Što znači da gornja crvena linija ne vrijedi pola poteza kao donja, već u ovom slučaju vrijedi samo jednu četvrtinu poteza. U tom primjeru igra ima vrijednost: +1 – ½ - ¼ = +¼.

U trećem primjeru Lijevi igrač ima samo jednu liniju koja uklanja jednu crvenu liniju iz igre. Na drugu crvenu liniju Lijevi igrač ne može utjecati. Ako Lijevi igrač igra prvi, Desni igrač će imati još jedan potez te će pobijediti. Ako Desni igrač igra prvi tada može izbrisati gornju crvenu liniju te realizirati suprotnu igru u odnosu na prvi primjer, odnosno +½ -1 = -½. Ako izbriše donju liniju tada ostavlja praznu ploču te opet pobjeđuje. U ovom primjeru donja crvena linija vrijedi -1, a gornja –¼, dok plava linija vrijedi +½. Ukupno, vrijednost igre je +½ -1 –½ = -¾.

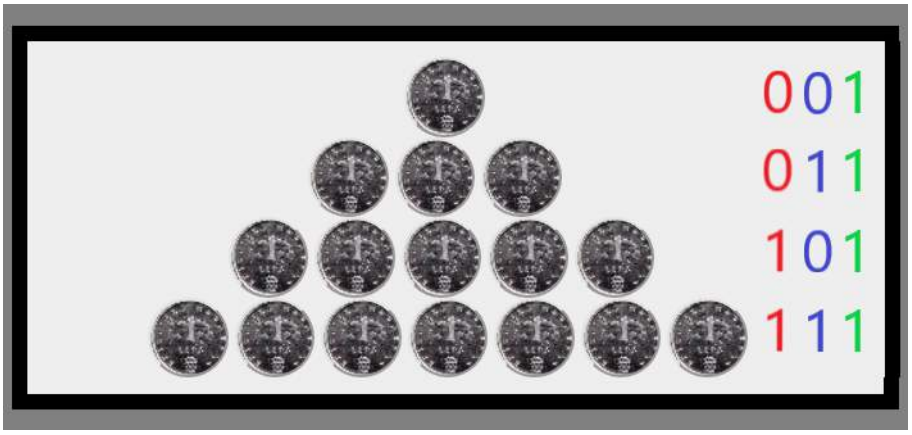


Slika br. 2: Složena igra Hackenbush

Na slici 2 je prikazan „pas“ sastavljen od crvenih i plavih linija. Ako se malo bolje pogleda, može se vidjeti da igrači ne mogu jedan drugome izbrisati linije. U ovom slučaju se vrijednost igre može izračunati brojanjem linija. Lijevi plavi dio lika se sastoji od 6 linija, a desni plavi dio lika se sastoji od 3 linije, dok se crveni dio lika sastoji od 7 linija. Ako Plavi igrač sam sebi ne izbriše više linija jednim potezom, uvijek će pobijediti. Vrijednost ove igre je: 9 – 7 = +2.

Nim

Igra Nim je jedna on najstarijih kombinatornih igara uopće. Porijeklom je vjerojatno iz Kine, a u Europi se prvi put spominje u izvorima koji datiraju s početka 16. stoljeća. Igra se s više hrpa nekog predmeta (šibice, novčići, žetoni, igraće karte itd.). Pobjednik se može utvrditi na normalan ili mizeran način. Svaki igrač u svom potezu uzima jedan ili više novčića, uz uvjet da u jednom potezu smije uzeti novčiće iz samo jednog reda.



Slika br. 3: Igra Nim

Na slici 3 vidi se jedna od varijanti igre Nim. Novčići su poslagani u 4 hrpe, odnosno retka. Ako se broj novčića zapiše u binarnom zapisu, može se vidjeti da se u svakom stupcu zapisa nalazi paran broj jedinica. Zelene znamenke u zapisu se mogu nazvati jedinice, plave dvojke, a crvene znamenke četvorke.

Igrač koji je na potezu u ovakvoj situaciji gubi igru. Koliko god novčića uzme, zadatak drugog igrača je da ponovo igru vrati u gubitničku poziciju, vodeći se prethodno navedenim pravilom: broj jedinica, dvojki i četvorki mora biti paran.

Ako Lijevi igrač uzme dvije kovanice s treće hrpe, na stolu će ostati sljedeći raspored novčića: 1-3-3-7, odnosno binarno zapisano:

0 0 1
0 1 1
0 1 1
1 1 1

Vidljivo je da ostaje neparan broj dvojki i četvorki. Da se igra vrati u gubitničku poziciju za Lijevog igrača, Desni igrač mora uzeti 6 novčića sa zadnje hrpe. Nakon drugog poteza stanje na stolu izgleda ovako: 1-3-3-1, odnosno:

0 0 1
0 1 1
0 1 1
0 0 1

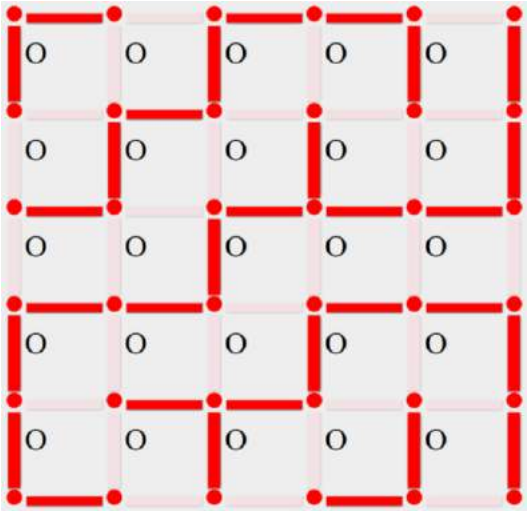
Treći potez neka bude uzimanje sve tri kovanice iz drugog reda. Igra se vraća u gubitničku poziciju uzimanjem svih kovanica iz trećeg reda. Nakon 4 poteza igra ostaje sa samo po jednom kovanicom u dva reda. Očito je da sljedeći igrač na redu gubi, u navedenom slučaju to je Lijevi igrač. Uzima predzadnju kovanicu te Desni igrač uzima zadnju i pobjeđuje ostavljajući prazan stol.

Kuglice i kutije

Kuglice i kutije (engl. Dots and Boxes) je jako poznata igra koja se uglavnom igra na listu papira s kvadratićima. Igra započinje s određenim brojem točaka koje su postavljene u redove i stupce, u obliku tablice. Igrači moraju naizmjence povezivati susjedne točke. Svaki igrač u jednom potezu ima pravo povući samo jednu crtu, odnosno povezati dvije točke. Kada se povezivanjem točaka zatvori lik u obliku kvadratića ili ćelije na tablici igrač koji je zadnji povukao potez, odnosno zatvorio kvadratić, dobiva jedan bod te zapisuje svoje inicijale u isti kvadratić. Svaki put kada igrač dobije bod, mora povući još jedan potez. Igra završava kada se sve točke spoje sa svojim susjedima, odnosno nacrtaju svi kvadratići. Nakon toga igrači prebrojavaju svoje bodove te je pobjednik igrač s više bodova. Za razliku od igre Nim, pobjednik ove igre se ne određuje zadnjim potezom, ali postoje određene taktike koje igraču daju prednost nad protivnikom.

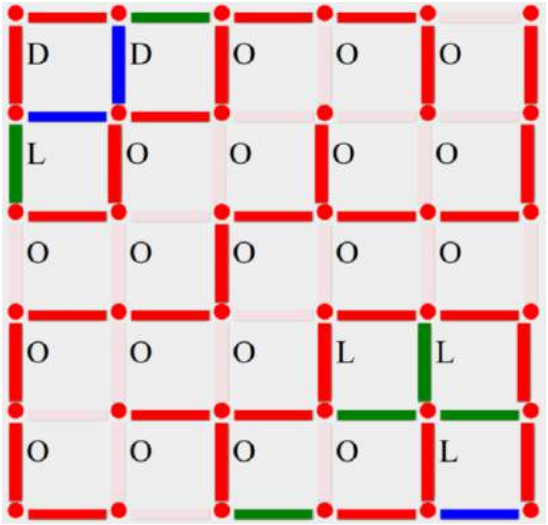
Pobjeda u ovoj igri je povezana s dugim lancima kvadratića. Dugi lanac se sastoji od tri ili više kvadratića koje jedan igrač može zatvoriti u jednom

potezu. U igrama s neparnim brojem kutija prvi igrač mora napraviti paran broj dugih lanaca, a drugi igrač mora napraviti neparan broj dugih lanaca za pobjedu. U igrama s parnim brojem kutija, prvi igrač mora napraviti neparan broj dugih lanaca, a drugi igrač mora napraviti paran broj lanaca za pobjedu.



Slika br. 4: Kuglice i kutije 5x5

Na slici 4 nakon 31 poteza postoje 4 duga lanca. Desni (drugi) igrač je na potezu te mora započeti jedan dugi lanac. Ako otvara najkraći lanac, odnosno onaj od tri kvadratića, Lijevi igrač mora uzeti samo jedan bod te druga dva ostaviti svom protivniku. To će natjerati Drugog igrača da otvori sljedeći dugi lanac te Lijevi igrač do kraja mora ponavljati istu taktiku: uzeti sve bodove osim zadnja dva u lancu.



Slika br. 5: Igra Kuglice i kutije 5x5 nakon dva poteza Lijevog igrača

Na slici 5 se vidi kako je Lijevi igrač uzeo jedan bod iz najkraćeg lanca te tri od pet bodova iz drugog lanca. Desni igrač može uzeti samo po dva boda iz svakog lanca. Vodeći se ovom taktikom Lijevi igrač pobjeđuje rezultatom 16-9.

Zaključak

Igranje kombinatornih igara je relativno jednostavno, ali za pobjedu u njima je potrebno imati određenu taktiku. Čak i ako su poznata sva pravila i taktike, ponekad je nemoguće pobijediti. Na početku igre je potrebno pogledati situaciju u kojoj se igrač nalazi, te je onda lako provjeriti isplati li se uopće igrati ili ne.

Reference

[1] Berlekamp E. R. (2000.) - The Dots and Boxes Game, AK Peters, Ltd.
[2] Berlekamp E. R., Conway J. H.i Guy R.K. (2003.) - Winning Ways for Your Mathematical Plays, Vol. 3, AK Peters, Ltd.
[3] Berlekamp E. R., Conway J. H.i Guy R.K. (2003.) - Winning Ways for Your Mathematical Plays, Vol. 2, AK Peters, Ltd.
[4] Berlekamp E. R., Conway J. H.i Guy R.K. (2001.) - Winning Ways for Your Mathematical Plays, Vol. 1, AK Peters, Ltd.
[5] Conway J.H. (2001.) – On numbers and Games, 2nd Edition, AK Peters, Ltd.
[6] Brilliant.org (bez dat.) Combinatorial Games – Definition, Preuzeto 02.08.2019. s https://brilliant.org/wiki/combinatorial-games-definition/
[7] Mathologer (07.06.2015.) NIM, or, always WIN with math [Video file]. Preuzeto 01.08.2019. s https://www.youtube.com/watch?v=niMjxNtiuu8

Izrada igre "Potapanje brodova" u C++-u

Mislav Matijević
Fakultet organizacije i informatike, Sveučilište u Zagrebu
mmatijevi@foi.hr

Sažetak

Ovaj rad ukratko predstavlja izradu i logiku igre „Potapanje brodova“ napisanu u jeziku C++ između 20. i 25. lipnja 2019. godine. U radu je naveden koncept same igre „Potapanje brodova“, izvedba realizacije koda videoigre, opis umjetne inteligencije računalnog suparnika te opis moda s dva igrača.

Ključne riječi: Potapanje brodova, igra, C++, umjetna inteligencija

Uvod

Videoigre su izvrstan spoj programiranja i zabave (ako već samo programiranje nije dovoljno zabavno). Otvaraju mogućnost programeru da osmisli i ostvari umjetnu inteligenciju, nauči računalo kako razmišljati u igri te kako pobijediti čovjeka. U današnje vrijeme igre su iznimno hardverski zahtjevne, podrazumijevaju kompleksne 3D prostore i optimiziranje istih simultano dok se igrač njima kreće, detaljne modele, priče kojih se i filmovi posrame te likove koji odaju dojam stvarnih osoba. Za stvaranje takvih igara potrebni su visoki budžeti i ogromni timovi.

No, jedna osoba itekako može stvoriti vlastitu igru. Ističe se alat *Construct* [1] kojim se vrlo jednostavno može stvoriti funkcionalna 2D igrica. Pogonski sklop za video igre *Unity* [2] dopušta korisniku da uz poznavanje jezika C# stvori čak i 3D okruženja, a i mnoge video igre dopuštaju zajednici igrača da za njih stvaraju i modificiraju terene, modele, misije i ostale elemente videoigara, odnosno stvaraju *modove*.

Jedan zadatak sa završne vježbe kolegija *Programiranje 1* poslužio mi je kao inspiracija da stvorim vlastitu igru u programu C++. Iako se već dugo bavim izradom misija i terena u raznim videoigrama, ovakav zadatak bio mi je izazov jer je podrazumijevao izradu igre skoro iz nule, koja će ispis imati praktički u DOS sučelju. Ipak, ovo je svakako bio jedan od najzabavnijih programa kojim sam se bavio, a smatram da može izvrsno poslužiti mojim kolegama studentima ne samo kao C++ kod, već i kao zanimljiv primjer i možda motivacija za izradu nečeg sličnoga, kao dokaz da se C++ ne mora nužno koristiti za rješavanje fakultetskih zadataka.

Koncept igre

Igra „Potapanje brodova“ društvena je igra nastala 1982. [3]. Igra se u dva igrača, od kojih oba postavljaju brodove na svojoj ploči od 10x10 polja. Najčešće su vrste brodova po jedan brod koji zauzima 4 polja u bilo kojem pravocrtnom smjeru, a potom dva broda po 3 polja, tri broda po 2 polja i konačno četiri broda po samo jedno polje. Pri postavljanju brodovi se ne smiju dodirivati. Igra se odvija u potezima tijekom kojih svaki igrač „na slijepo“ gađa neprijateljsko polje izgovarajući koordinate polja koje gađa (npr. „1-5“ bilo bi polje u prvom retku i petom stupcu). Ako igrač promaši, odnosno pogodi „more“, potez nastavlja drugi igrač. Osim svojih ploča na koje igrači postavljaju brodove, svaki igrač također ima i praznu ploču na kojoj označava pogotke/promašaje na tuđem terenu. Ako je igrač pogodio brod, suparnik mu to priopći te igrač označava pogodak. Ako je brod razoren (sva polja su mu uništena), igrač na svojoj ploči označava koji je brod uništio. Budući da se brodovi ne smiju dodirivati, igrač zna da ako je uništio neprijateljski brod ne treba gađati na polja oko tog broda.

Izrada igre

Prvi korak pri izradi ove igre bio je osmisliti izgled ploče. Ploča je dvodimenzionalno polje koje sadržava varijable tipa „char“, a one su jedan od sljedećih znakova:

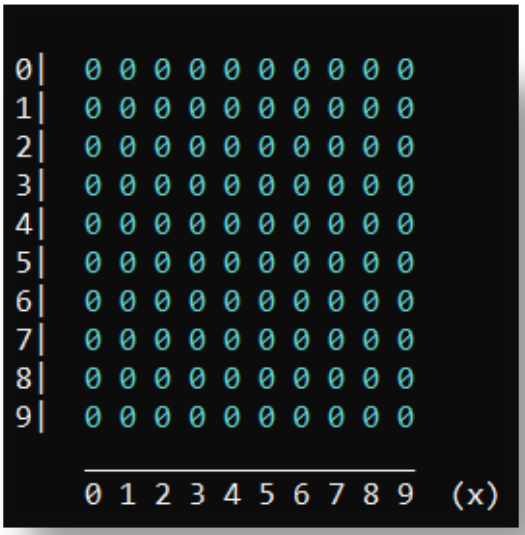
- '0' polje je nepogođeno, odnosno „more“
- '-' polje je pogođeno, ali nije bilo broda na njemu
- 'X' na neprijateljskoj ploči oznaka za pogođeni, neuništeni brod
- '1', '2', '3' ili '4' označava vrstu broda na tom polju

U kasnijoj verziji dodane su boje, pa je svaka vrsta polja dobila i svoju boju pri ispisu (plava za „more“, crvena za pogođeni/uništeni brod)

Kao drugi korak trebalo je definirati kako se postavljaju brodovi. Igrač može birati ili da mu računalo automatski razmjesti brodove, ili da ih on sam ručno postavi. Sve se svodi na odabir koordinata prvog polja broda i smjera po kojemu će ostatak polja biti smješteno. Bilo je nužno namjestiti mnogo pravila: brod ne smije izlaziti iz okvira „ploče“, brod ne smije dodirivati drugi brod te brod ne smije dodirivati/prolaziti kroz nijedno polje koje dodiruje neki drugi brod. Ova pravila ostvarena su kao funkcije koje vraćaju vrijednosti tipa „bool“ temeljem toga jesu li svi uvjeti zadovoljeni.

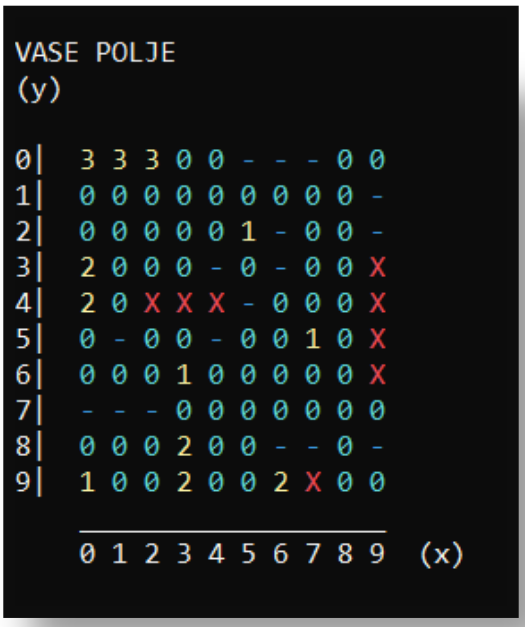
Pomoću funkcije s dvije *for* petlje ispisuju se polja – svako sa svojim odgovarajućim znakom i bojom. Ova funkcija poziva se prije svakog sljedećeg poteza.

Ispis prazne „ploče“ izgleda ovako:



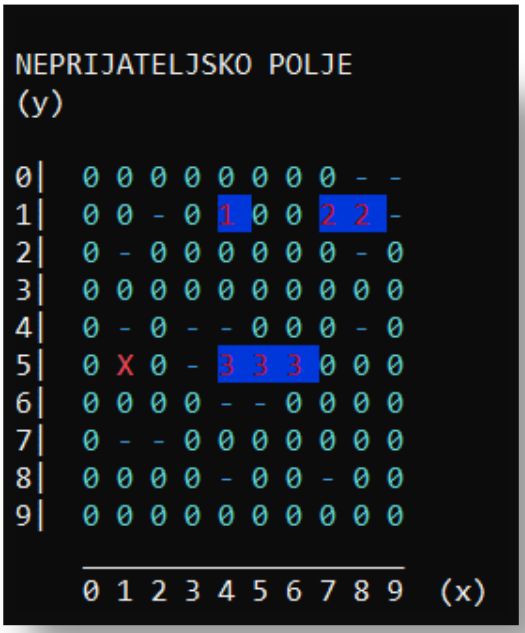
Slika br. 1: Prazna „ploča“

Izgled igračeve ploče tijekom igre izgleda ovako:



Slika br. 2: Igračeva ploča

Izgled neprijateljske ploče tijekom igre izgleda ovako:



Slika br. 3: Neprijateljska ploča

Nakon uspješnog ostvarenja ispisa polja, trebalo je definirati potez. Potez je velika funkcija koja omogućava ili igraču ili računalu da pogodi koordinatu. Tko igra prvi potez u pravilu bira računalo, osim na najtežoj razini igre kada računalo uvijek igra prvo. Priča s igračem je jednostavna, igrač upiše (brojčane) koordinate koje želi gađati, a potom se ispiše odgovarajuća poruka ako je igrač pogodio, odnosno promašio neprijateljski brod. Problem nastaje kod poteza računala.

Umjetna inteligencija

U prvoj, najprimitivnijoj verziji igre, računalo bi tijekom svakog svog poteza pseudoslučajnim odabirom odredilo koordinate koje želi gađati. No, ovo je dovelo igrača u nadmoćnu poziciju nad svojim računalnim suparnikom. Naime, kada igrač pogodi brod, osim ako je brod od jednoga polja, igrač zna da i tijekom sljedećeg poteza treba nastaviti gađati negdje oko tog pogotka sve dok ne uništi brod do kraja. Također, nakon uništenja broda, igrač svakako više neće gađati u polja koja dodiruju uništeni brod jer pravila igre ne dopuštaju doticanje brodova. Bilo je nužno naučiti računalo da pazi na isto. Za početak, računalo također ima praznu „ploču“ na kojoj prati pogotke igračevih brodova – to je jedina „ploča“ koja se u igri ne ispisuje na zaslon. Računalo je moguće postaviti na tri razine težine:

1. Najlakša razina; računalo slučajno gađa polja, ne reagira na pogotke

2. Srednja razina; računalo traži položaj broda nakon uspješnog pogotka
3. Najteža razina; računalo ima 30% šanse da pogodi igračev brod

Računalo na uspješan pogodak na srednjoj i teškoj razini reagira prvo gađanjem u smjeru „sjevera“ (prema gore) u odnosu na uspješno pogođeno polje. Nakon toga, ako je ostvaren promašaj, nastavlja gađanjem u smjeru „juga“ (prema dolje), zatim lijevo i konačno, u slučaju da su svi pogodci promašaji, gađa desno. Ako je ostvaren pogodak, i sljedeći će potez gađati u istom smjeru (jer je to očito smjer u kojemu je postavljen brod). Tu se stvara problem: što ako je računalo počelo gađati brod na sredini? To znači da će kada gađanjem dođe do prvog praznog polja / do kraja broda pomisliti da ga je uništilo. Zato je bilo nužno uvesti provjeru je li brod uništen (jesu li se X-ići kao oznake pogođenih polja u računalovoj „nevidljivoj ploči“ neprijateljskih brodova pretvorili u brojeve kao oznake veličine potopljenog broda ili ne, odnosno ili brod nije potopljen). Također nužno je bilo uvesti varijablu koja prati koje je polje računalo prvo pogodilo te se u to polje vratiti i nastaviti gađati u suprotnom smjeru. Kroz ovu logiku ostvarila se umjetna inteligencija u ovoj igri i računalo zna kako gađati i uništiti igračev brod.

Mod igre za dva igrača

U modu igre za dva igrača naglasak je prebačen s logike igre na tajnost polja. Igrači prvo unesu svoja imena, tako da se računalo imenom obraća svakom igraču. Ovo je bitno, jer je za privatnost polja nužno da se zna kada koji igrač smije gledati konzolu na kojoj je ispisana igra. Za razliku od načina igre protiv računala, igrači svoju vlastitu ploču vide samo kada postavljaju brodove. Tijekom svakog poteza upitani su žele li pogledati svoju ploču i tako je osigurano da se polje svakoga igrača ispisuje samo onda kada on to želi i kada je siguran da suigrač ne gleda u konzolu. Također, ekran se nakon svakoga poteza osvježava, tako da sav prijašnji sadržaj stalno nestaje (nije slučaj u modu igre protiv računala kada igrač može na konzoli gledati povijest igre). Jedino su bodovi vidljivi svima. Tako je omogućeno diskretno postavljanje brodova svakome igraču, kao i pregled vlastitih ploča.

Zaključak

Umjetna inteligencija u videoigrama nije manifestirana samo u visokobudžetnim projektima. Smatram da je izrada videoigre u programskom jeziku jedan od najboljih načina da se s tim programskim jezikom bolje upoznamo. Analiziranje vlastitih misli pri nekoj igri te zapisivanje te logike u obliku programskog koda koji će „uputiti“ računao kako da razmišlja tijekom igre vrlo je zanimljiv proces i svaki trenutak pisanja igre „Potapanje brodova“ mi je bio izuzetno zanimljiv. Vjerujem da ovakvi primjeri mogu studentima

pomoći da nauče programirati, ali i kao motivacija za ulazak u svijet razvoja videoigara.

Reference

[1] Construct 3 [Na Internetu] Dostupno na: https://www.construct.net/en [Pristupano 17.9.2019.]

[2] Unity [Na Internetu] Dostupno na: https://unity.com/ [Pristupano 17.9.2019.]

[3] Battleship (puzzle) [Na Internetu] Dostupno na: https://en.wikipedia.org/wiki/Battleship_(puzzle) [Pristupano 17.9.2019.]

STRUČNA KONFERENCIJA

RAČUNALNE IGRE 2019

4. listopada 2019.