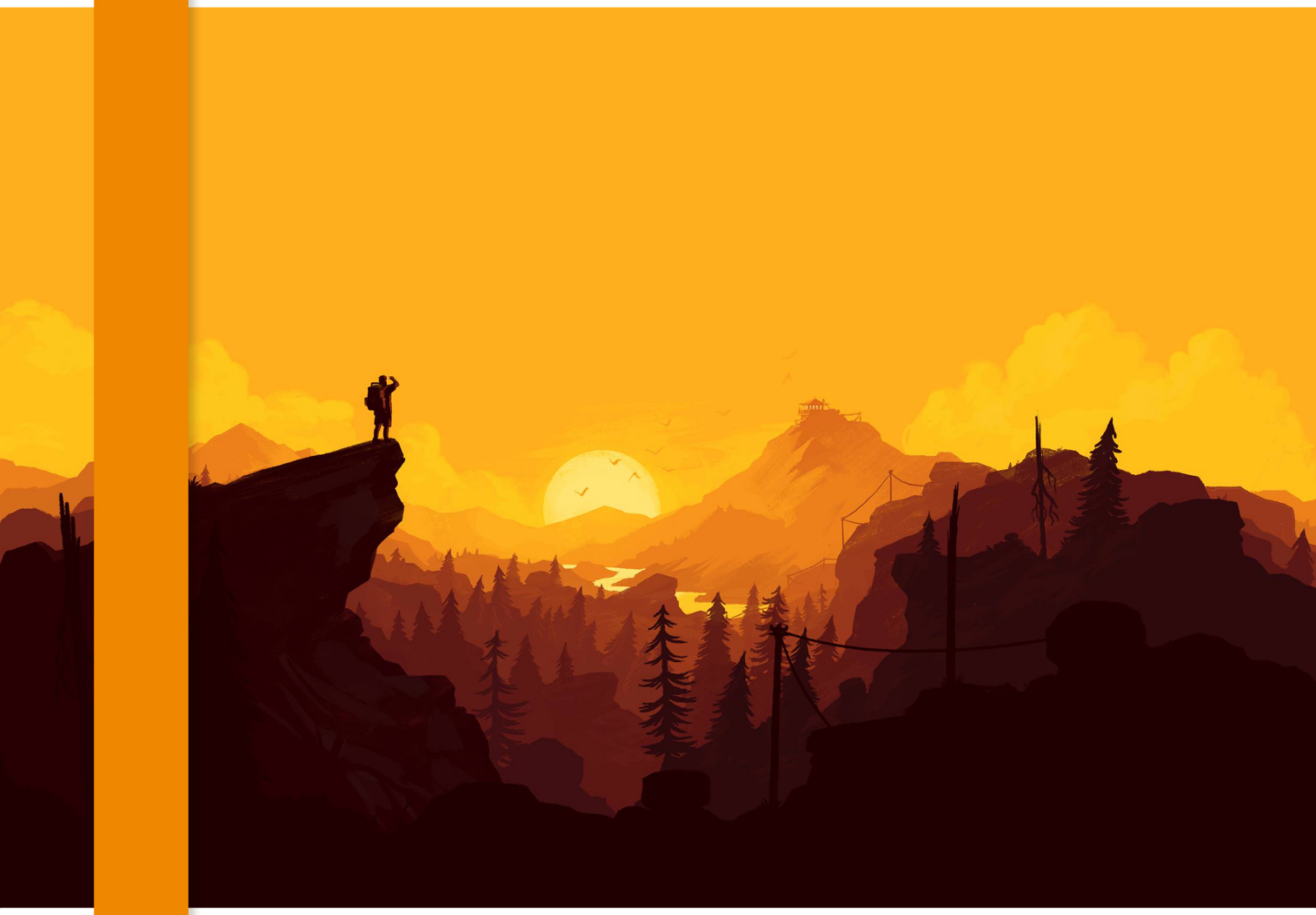
FAKULTET ORGANIZACIJE I INFORMATIKE



ZBORNIK RADOVA

RAČUNALNE IGRE 2021

STRUČNA KONFERENCIJA

UREDNICI

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

ISSN 2623-7679

IMPRESUM

IZDAVAČ Konferencija Računalne igre 2021

NAKLADNIK Fakultet organizacije i informatike
Pavlinska 2, 42000 Varaždin

KONTAKT racunalne-igre@foi.hr

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

doc. dr. sc. Mladen Konecki
Sveučilište u Zagrebu

Andrej Kovačević
Exordium Games

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

DATUM ODRŽAVANJA KONFERENCIJE 31. prosinca 2021.

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

PROGRAMSKI ODBOR

izv. prof. dr. sc. Mario Konecki
predsjednik, Sveučilište u Zagrebu

prof. dr. sc. Damir Boras
rektor, Sveučilište u Zagrebu

prof. dr. sc. Snježana Prijić Samaržija
rektorica, Sveučilište u Rijeci

prof. dr. sc. Dragan Ljutić
rektor, Sveučilište u Splitu

prof. dr. sc. Vlado Guberac
rektor, Sveučilište u Osijeku

prof. dr. sc. Dijana Vican
rektorica, Sveučilište u Zadru

prof. dr. sc. Nikša Burum
rektor, Sveučilište u Dubrovniku

prof. dr. sc. Alfio Barbieri
rektor, Sveučilište u Puli

prof. dr. sc. Marin Milković
rektor, Sveučilište Sjever

prof. dr. sc. Ivanka Stričević
prorektorica, Sveučilište u Zadru

prof. dr. sc. Drago Žagar
dekan, Sveučilište u Osijeku

prof. dr. sc. Vlatko Cvrtila
dekan, VERN'

izv. prof. art. Davor Švaić
prodekan, Sveučilište u Zagrebu

izv. prof. dr. sc. Ante Bilić-Prcić
prodekan, Sveučilište u Zagrebu

prof. dr. sc. Patrizia Poščić
Sveučilište u Rijeci

izv. prof. dr. sc. Giorgio Sinković
dekan, Sveučilište u Puli

doc. dr. sc. Časlav Livada
Sveučilište u Osijeku

prof. dr. sc. Marijan Cingula
Sveučilište u Zagrebu

doc. dr. sc. Tin Turković
Sveučilište u Zagrebu

Andrej Kovačević
direktor, Exordium Games

Darjan Vlahov
pročelnik, Sisačko-moslavačka županija

Mario Čelan
direktor, SiMoRa

dr. sc. Goran Đambić
Algebra

Lovro Nola, direktor
Machina

doc. dr. sc. Mladen Konecki
Sveučilište u Zagrebu

prof. dr. sc. Kornelije Rabuzin
Sveučilište u Zagrebu

prof. dr. sc. Mirko Čubrilo
Sveučilište u Zagrebu

prof. dr. sc. Mirko Maleković
Sveučilište u Zagrebu

prof. dr. sc. Alen Lovrenčić
Sveučilište u Zagrebu

prof. dr. sc. Danijel Radošević
Sveučilište u Zagrebu

izv. prof. dr. sc. Zlatko Stapić
Sveučilište u Zagrebu

izv. prof. dr. sc. Igor Pihir
Sveučilište u Zagrebu

izv. prof. dr. sc. Dijana Plantak Vukovac
Sveučilište u Zagrebu

izv. prof. dr. sc. Dijana Oreški
Sveučilište u Zagrebu

doc. dr. sc. Irena Kedmenec
Sveučilište u Zagrebu

ORGANIZACIJSKI ODBOR

doc. dr. sc. Mladen Konecki
predsjednik odbora, Sveučilište u Zagrebu

izv. prof. dr. sc. Mario Konecki
Sveučilište u Zagrebu

Andrej Kovačević
direktor, Exordium Games

Darjan Vlahov
pročelnik, Sisačko-moslavačka županija

Mario Čelan
direktor, SiMoRa

Luka Milić, mag. ing. comp.
Sveučilište u Zagrebu

dr. sc. Alan Mutka
Rochester Institute of Technology

doc. dr. sc. Tin Turković
Sveučilište u Zagrebu

doc. dr. sc. Sanja Ćurković
Sveučilište u Zagrebu

Damir Vučić, prof.
Sveučilište u Zagrebu

Ivan Perkov, mag. soc.
Sveučilište u Zagrebu

izv. prof. dr. sc. Dijana Plantak Vukovac
Sveučilište u Zagrebu

doc. dr. sc. Nikola Ivković
Sveučilište u Zagrebu

doc. dr. sc. Ivan Malbašić
Sveučilište u Zagrebu

ISSN 2623-7679

STRUČNA KONFERENCIJA

RAČUNALNE IGRE 2021

31. prosinca 2021.

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 2002 Universitas Studiorum Jadertina

Sveučilište u Dubrovniku Universitas Studiorum Ragusina

Universitas Studiorum Polensis Georgii Dobrila

University North
Sveučilište Sjever
MMXV

ALGEBRA VISOKO UČILIŠTE

MACHINA GAME DEV ACADEMY

VERN'

Grad Varaždin

Grad Bjelovar

Grad Daruvar

GENERALNI POKROVITELJ

SISAČKO-MOSLAVAČKA ŽUPANIJA

STRUČNA KONFERENCIJA

RAČUNALNE IGRE 2021

31. prosinca 2021.

ORGANIZATORI I SUORGANIZATORI

foi SVEUČILIŠTE U ZAGREBU FAKULTET ORGANIZACIJE I INFORMATIKE VARAŽDIN

CECIIS Central European Conference on Information and Intelligent Systems

Sveučilište u Zagrebu

Sveučilište u Rijeci

Sveučilište u Splitu

Sveučilište Josipa Jurja Strossmayera u Osijeku

Sveučilište u Zadru

Sveučilište u Dubrovniku

Universitas Studiorum Polensis Georgii Dobrila

Sveučilište Sjever

ALGEBRA VISOKO UČILIŠTE

MACHINA GAME DEV ACADEMY

VERN'

Grad Varaždin

Grad Bjelovar

Grad Daruvar

GENERALNI POKROVITELJ

SISAČKO-MOSLAVAČKA ŽUPANIJA

SADRŽAJ

1 **Razvoj trodimenzionalne računalne igre u programskom alatu Unity**
Domagoj Hamzić, Mario Konecki i Alen Lovrenčić

9 **Proces razvoja računalne igre s 3D elementima u razvojnom okruženju Unity**
Matija Ivanić i Mario Konecki

16 **Umrežene igre sljedeće generacije**
Josip Petanjek, Danijel Radošević i Mladen Konecki

24 **Izrada strateške igre u stvarnom vremenu u programskom alatu Unity**
Deni Baban i Mladen Konecki

31 **Izrada „shoot 'em up" igre u programskom alatu Unity**
Erik Baukovac i Mladen Konecki

37 **Izrada 2D platformera u programskom alatu Unity**
Petar Mihalić

43 **Automatizirani robotski proces praćenja IR objekta unutar određenih granica**
Ivan Kolarić i Mario Konecki

50 **Razvoj proceduralno-generirane računalne igre simulacijskog tipa u razvojnom okruženju Unity**
Alan Vranić i Mario Konecki

59 **Uloga računalnih igara u obrazovanju**
Ivan Blažek

70 **Popularnost žanrova računalnih igara**
Julijan Brlić

80 **Izrada akcijske 3D računalne igre**
Lino Zadravec, Mario Konecki i Alen Lovrenčić

86 **Dizajniranje računalnih igara**
Tomislav Jukica i Mladen Konecki

95 **Izrada igre preživljavanja u programskom alatu Unity**
Dominik Štefičar i Mladen Konecki

101 **Izrada igre šaha u Unity okruženju**
Mario Nižić, Danijel Radošević i Mladen Konecki

107 **Izrada 2D platformske videoigre u Godotu**
Dominik Kavur, Danijel Radošević i Mladen Konecki

115 **Izrada „pixel art" igre uloga u programskom alatu Unity**
Danijel Tokić i Mladen Konecki

122 **Izrada videoigre preživljavanja u alatu Unreal Engine 4**
Emil Mihalić, Danijel Radošević i Mladen Konecki

Razvoj trodimenzionalne računalne igre u programskom alatu Unity

Domagoj Hamzić, Mario Konecki, Alen Lovrenčić
Fakultet organizacije i informatike, Sveučilište u Zagrebu
domagoj.hamzic@foi.hr, mario.konecki@foi.hr, alen.lovrencic@foi.hr

Sažetak

Ovaj rad baziran je na osnovama kreiranja trodimenzionalne računalne igre te kreiranju grafike tj. objekata u alatu za trodimenzionalno modeliranje. Rad je rađen pomoću dva alata, Unity i Blender. Unity je služio za kreiranje računalne igre, poslovne logike dok je Blender korišten za kreiranje trodimenzionalnih objekata koji su uneseni u samu igru. Svi objekti koji imaju fizičko ponašanje i animaciju su vlastoručno kreirani dok su statični objekti preuzeti te su navedeni izvori preuzimanja istih.

Ključne riječi: 3D računalna igra, zagonetke, Escape room, Unity, Blender, programiranje

Igra je inspirirana društvenom igrom u kojoj grupa igrača ulazi u tematski uređen prostor i pred njima je niz različitih zagonetki, mozgalica, ključeva, šifri koje je potrebno dekodirati kako bi unutar zadanog vremena uspješno obavili zadanu misiju te izašli iz sobe. Ova igra je za razliku od društvene, namijenjena za jednu osobu te se svaki pokušaj evidentira kako bi se mogao usporediti sa drugim pokušajima.

Kreiranje modela

Modeli koji imaju animaciju u igri, kreirani su u besplatnom programskom alatu Blender koji se koristi za izradu računalne grafike, animiranih filmova, vizualnih efekata i sl. Proces izrade modela može se rastaviti na tri dijela:

- Oblik modela
- Teksturiranje modela
- Animiranje modela

Izrađeni modeli su se izvozili iz Blendera u .fbx formatu koji se vrlo lako uvozi u Unity gdje se krije poslovna logika igre.

Poslovna logika

Cilj igre je bijeg iz zatvorenog prostora pomoću logičkog rješavanja zagonetki. Postoje dvije razine koje su tematski različite, ali imaju isti cilj. Mehanika

kretanja glavnog junaka se vrši pomoću tipka A, S, D, W, a pomoću miša se izvršava akcije rotacije pogleda. Prikupljanje objekata izvršava se pomoću tipke E (igrač je upućen direktno u igri kako manipulirati sa određenim objektima). Korisnik tijekom igre može pauzirati igru pomoću tipke P (vrijeme se zaustavlja) te ima opciju nastavljanja, ponavljanja ili završetka igre. Ukoliko korisnik želi provjeriti koji mu se objekti nalaze u Inventory sistemu, potrebno je pritisnuti tipku I. Svaki objekt u Inventory ima jedinstvenu tipku (1-9) pomoću koje se može koristiti izabrani objekt.

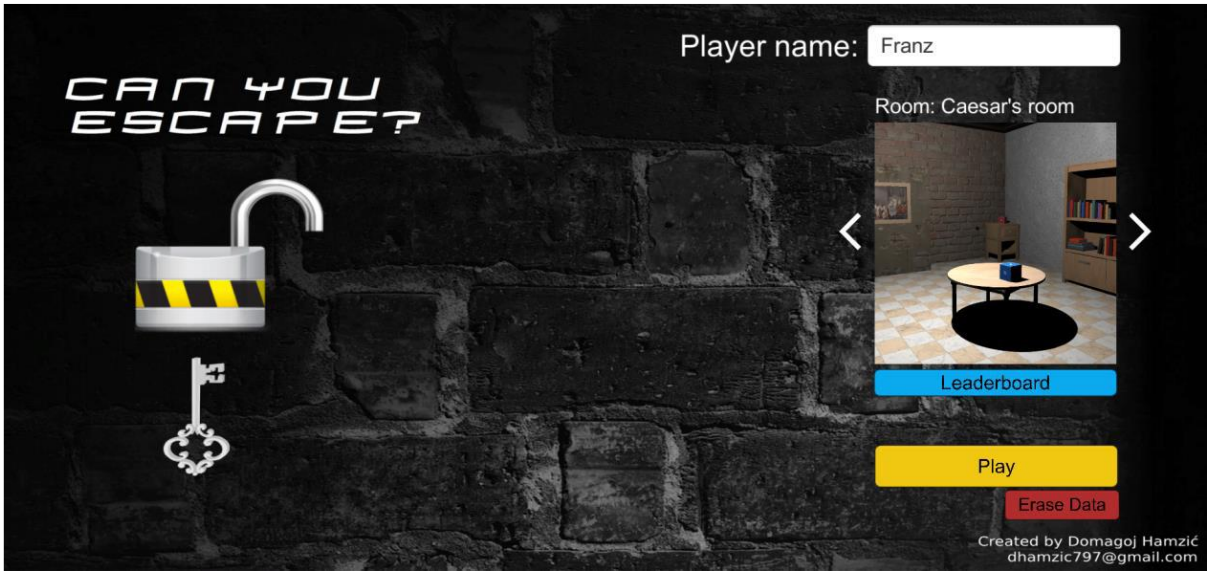
Upute igranja

Slijedi objašnjenje kako pobjeći iz soba koje su kreirane. Za potrebe izrade diplomskog rada, izrađene su dvije sobe:

- Caesar's room
- Morse's room

Korisniku se prvo prikaže glavni meni. Glavni meni sastoji se od prostora za unos imena igrača, odabira sobe i tri tipke:

- Leaderboard
  - Rezultati pojedinačne sobe
- Play
  - Igranje igre, sljedeća scena je igra
- Erase Dana
  - Brisanje svih postignutih rezultata (PlayerPrefs.DeleteAll())



Slika 1. Glavni izbornik [autorski rad]

Caesar's Room

Za razliku od Morse's room, Caesar's room nema ograničeno vrijeme izlaza iz sobe. Ukoliko korisnik želi pauzirati igricu, potrebno je pritisnuti tipku „P“ na tipkovnici. Ukoliko igrač želi koristiti objekte koje je prikupio u Inventory-u, potrebno je pritisnuti tipku „I“. Slijedi objašnjenje savršeno odigrane igre.



Slika 2. Caesar's room [autorski rad]

Ispred se nalazi ormar i na njemu ključ kojeg je potrebno uzeti. Ključ otključava zaobljeni ormarić. Ranije je rečeno da svaki objekt sa kojim se može vršiti interakcija, prikazuje opis korištenja istog.



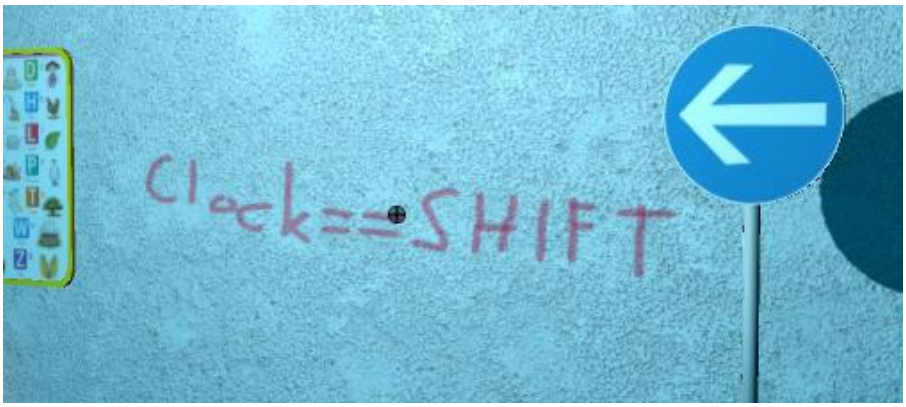
Slika 3. Ormarić [autorski rad]

U prvoj ladici se nalazi knjiga koja sadrži sljedeći niz znakova: „HVFDSH“ koje trenutno ništa ne govore. U drugoj ladici se nalazi papirić sa oznakom koja je identična kao i slika na prekidaču za svjetlo.



Slika 4. Hydra [autorski rad]

U trećoj ladici nalazi se ključ koji se kasnije koristi. Prekidač pali UV lampu koja otkriva skrivenu poruku na istočnom zidu. Isto tako pored poruke se nalazi znak sa obaveznim skretanjem u lijevo. Sat na zidu prikazuje 3h. Riječ je o Caesar shift zagonetki. Svaki znak u nizu znakova „HVFDSH“ potrebno je pomaknuti za 3 polja u lijevo. „H“ postaje „E“, „V“ postaje „S“, „F“ postaje „C“, „D“ postaje „A“, „S“ postaje „P“ i „H“ postaje „E“. Rješenje algoritma je „ESCAPE“. Ukoliko postoje igrači koji imaju problema sa abecedom, lijevo od skrivene poruke „Clock==SHIFT“ nalazi se slika sa engleskom abecedom.



Slika 5. Caesar Shift [autorski rad]

U sobi se nalaze kocke sa slovima te se svako slovo nalazi u nizu znakova „ESCAPE“. Potrebno je kocke poredati po cilindrima tako da čine niz znakova „ESCAPE“.



Slika 6. ESCAP_ [autorski rad]

Nedostaje jedna kocka, točnije zadnja sa slovom „E“. Potrebno ju je pronaći. Kocka je skrivena iza slike Cezara. Potrebno je baciti sliku.

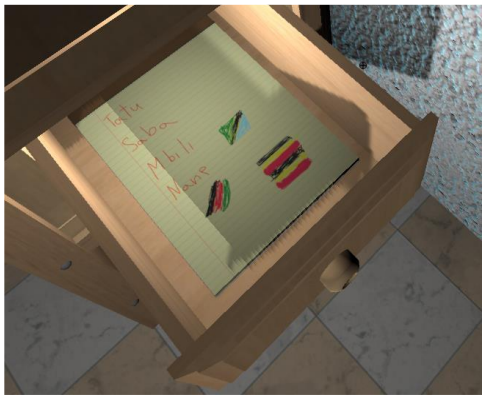


Slika 7. Ceazrova slika [autorski rad]

Iza slike se nalazi sef koji traži lozinku od četiri znaka. Desno od sefa nalazi se ormarić koji još uvijek nije istražen. U ladici se nalazi papirić sa nepoznatim znakovima.

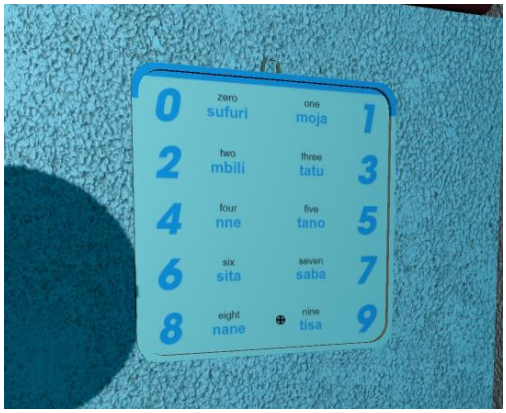


Slika 8. Sef [autorski rad]



Slika 9. Svahili papirći [autroski rad]

U sobi je pronađena slika koja nije istražena. Na slici se nalaze riječi, identične kao i na papiriću te je svaka riječ mapirana sa brojem. „Tatu“ je „3“, „Saba“ je „7“, „Mbili“ je „2“ i „Nane“ je „8“. Šifra za otvaranje sefa je sljedeći niz brojeva: 3728.



Slika 8. Svahili slika [autorski rad]

Unutar sefa nalazi se tražena kocka koju je potrebno staviti na ispravan cilindar. Nakon postavljanja kocke, ormar sa knjigama se otvara te otkriva skrivena vrata koja je potrebno otključati sa ključem koji je ranije prikupljen iz treće ladice zaobljenog ormarića uz istočni zid. Nakon izlaska iz sobe, igraču se otvaraju informacije o odigranoj igri.



Slika 9. Rezultat [autorski rad]

Morse's Room

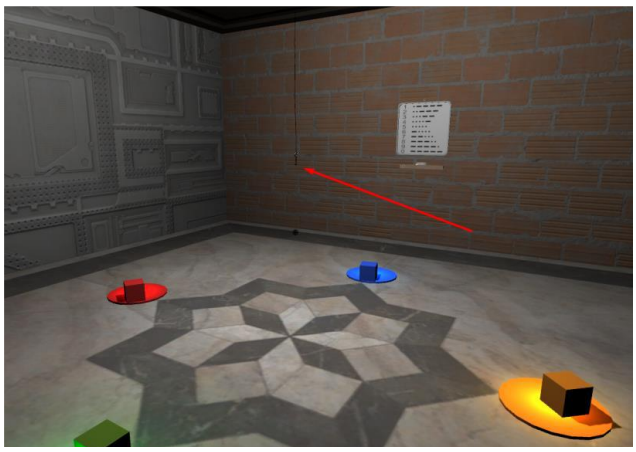
Za razliku od prethodno opisane sobe, izlazak iz ove sobe je vremenski ograničen. Ukoliko igrač ne stigne izaći na vrijeme, igra se zaustavlja tj. prekida.



Slika 10. Morse's Room [autorski rad]

Potrebno je postaviti svaku kocku na boju cilindra koja je jednaka boji kocke. Kocke su smještene na ladici koja je igraču trenutno nedohvatljiva. Da bi dohvatio kocke, potrebno je stisnuti prekidač koji se nalazi desno od police sa kockama. Prekidač pomiče policu sa koje kao rezultat, ispadaju kocke. Nakon toga je potrebno postaviti kocke na ispravnu boju. Nakon što se kocke postave

na odgovarajući cilindra, uže sa ključem se spušta te je moguće uzeti ključ i spremiti ga u Inventory.



Slika 11. Postavljanje kocka [autorski rad]



Slika 12. Morseov kod [autroski rad]

Ključ otključava ladicu koja se nalazi ispod slike Morseovog koda. Nakon pritiska tipke (Simbol Play), igraču se pusti zvuk Morseovog koda koji predstavlja lozinku za otključavanje ladice koja se nalazi u desnom gornjem kutu istočnog zida. Potrebno je pomoću slike Morseovog koda i zvuka, otkriti o kojoj je 4-znamenkastoj lozinki riječ. Ispravna lozinka je 9621. Dobivenu lozinku treba ukucati u polje za unos koje se nalazi na ranije navedenoj ladici. Unutar ladice se nalazi daljinski upravljač koji zahtjeva unos od tri broja. Na zidu se nalaze tri sata od kojih jedan prikazuje 3h, drugi 8h i treći 5h. To su brojevi koje je potrebno unesti na daljinski upravljač i nakon toga pritisnuti tipku „OPEN“. Nakon unosa tražene kombinacije brojeva i pritiska tipke „OPEN“, iz poda se uzdiže cilindar sa ključem na vrhu. Ključ otključava vrata koja predstavljaju kraj igre.



Slika 13. Ladica [autroski rad]



Slika 15. Ključ [autorski rad]



Slika 14. Daljinski [autroski rad]

Zaključak

Alat Unity idealan je za početnike jer pruža jednostavno sučelje uz mnogo ugrađenih Unity mehanizama koji olakšavaju izradu poslovne logike igre. Alat Blender je također vrlo jednostavan za izradu modela te je vrlo kompatibilan sa Unity-em tako da je izvoz tj. uvoz modela također riješen vrlo elegantno i bezbolno.

Što se tiče mojeg osobnog mišljenja, trenutno izrađena igra ima mnogo prostora za napredak, možda i koja ideja više što se tiče zagonetki, pogotovo veća mašta prilikom izrade priče igre te proširenje igre na više korisnika odjednom (Multiplayer). Za izradu ozbiljnije igre, ipak je potrebno više ljudi, minimalno troje gdje bi jedna osoba bila zadužena za izradu modela/animacija, druga za poslovnu logiku i treća za kreiranje priče igre.

Reference

[1] D. Hamzić, „Trodimenzionalno modeliranje objekta za stanovanje i uređenje okoliša u Blenderu“, info:eu-repo/semantics/bachelorThesis, University of Zagreb. Faculty of Organization and Informatics. Department of Computing and Technology, 2019. Pristupljeno: kol. 25, 2021. [Na internetu]. Dostupno na: https://urn.nsk.hr/urn:nbn:hr:211:231339

[2] „Master Procedural Maze & Dungeon Generation“, Udemy. https://www.udemy.com/course/procedural-maze-dungeon-generation/ (pristupljeno ruj. 01, 2021).

[3] „Build A Multiplayer Kart Racing Game In Unity V.2019“, Udemy. https://www.udemy.com/course/kart-racing/ (pristupljeno ruj. 01, 2021).

Proces razvoja računalne igre s 3D elementima u razvojnom okruženju Unity

Matija Ivanić i Mario Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
mivanic@foi.hr, mkonecki@foi.unizg.hr

Sažetak

Tema ovog rada je proces razvoja računalne video igre. Biti će detaljno objašnjeni koraci u izradi igre. Ovaj rad sastojat će se od funkcionalnog prototipa računalne igre koja će biti implementirana u Unity razvojnom okruženju, te opisan korak po korak u njezinoj izradi. Bit će detaljno opisan i prikazan proces razvoja igre, kako bi nekome tko čita ovaj rad bio lakše započeti vlastiti razvoj video igre.

Ključne riječi: Video Igra; Unity; Razvoj, 3D Game Kit; Agenti; Avanturistička igra; Programiranje; 3D, Animacije

Uvod

Danas su video igre ne zaobilazna stvar. Svi danas u nekoj mjeri igraju video igre, da li one bile na računalima, konzolama ili na mobitelima. Stoga igre više nisu ''tabu'' tema, o kojoj se malo piše. Novac koji se vrti u industriji video igara danas je već nego filmska i glazbena industrija zajedno. To nešto govori o tome koliko je danas ova stvar popularna i koliko je rasprostranjena u svijetu.

Uz veliku popularnost video igara, tako je porasla popularnost za samostalnu izradu video igre. Tako danas postoji velik broj alata koji omogućavaju neku vrstu izrade video igre. A među svim tim alatima, postoje oni koji su više popularni i koji daju veće mogućnosti korisnicima tijekom izrade. Jedan od takvih alata sigurno je i Unity razvojno okruženje. Uz sve njegove mogućnosti koje ću kasnije detaljno opisati, vrijedi spomenuti kako je besplatan za korištenje, te time dostupan svima.

Unity

Da bismo mogli bolje shvatiti što je to Unity, potrebno ga je prvo detaljno opisati. Unity je razvojno okruženje (engl. Engine), koji se prvenstveno koristi u svrhu izrade video igara [1]. Područje koje pokriva Unity danas je veliko, odnosno raznoliko, te nije namijenjeno samo video igrama iako mu je to glavno područje. Tako se danas Unity može koristiti u svrhe automatizaciji,

transportu u proizvodnji, koristiti za izradu filmova, raznih animacija i kinematike, izrade raznih arhitekturnih i inženjerskih stvari [9]. kinematike, izrade raznih arhitekturnih i inženjerskih stvari [9]. Tu možemo vidjeti kako su područja u kojoj se primjenjuje je veoma široko. Ipak preko 90% Unity se koristi za izradu video igara, simulacije, sučelja za pametne telefone, mobilne i desktop aplikacije.[1][2]

Neznačajnija karakteristika koju Unity pruža je svakako podrška za dvodimenzionalne (2D) i trodimenzionalne (3D) igre za računala, konzole, web i mobilne uređaje. Tako za desktop platformi, developeri mogu kreirati igre za Windows, Windows Store, Mac OS X, te Linux, gdje Ubuntu ima specijalnu podršku, no radi i za ostale platforme. Kod Windows operacijskog sustava, developer može birati između 32-bitnog i 64-bitnog sustava. Što se tiče weba, Unity ima podršku za Unity Web Player, Google Native Client i Flash [10]. Unity developerima pruža tri načina programiranja njihovih aplikacija, tj. pisanja skripata, odnosno skriptiranje. Tako imamo Unity JavaScript, poznatiji kao UnityScript, C#, koji je daleko najpopularniji, te se najviše koristi, te Boo.

Unity sučelje

Unity sučelje je veoma „User Friendly“. To znači ima veoma moderno sučelje koje je jednostavno korisniku za korištenje. Organizacija samog sučelja je veoma logička i jednostavna, te sve opcije su korisniku na dohvat ruke. Jedna od bitnih stavki kod svakog programa je njegova brzina izvršavanja, a u tome je Unity svakako među najbržim alatima. Svako izvršavanje, svaka animacija unutar alata je veoma fluidna, te se dobiva osjećaj da je program izrazito brz i traži malo resursa za izvršavanje svojih zadaća.

Sučelje Unity-ija se sastoji od različitih prozora, od kojih svaki ima svoju namjenu. Neka od glavnih sučelja su:

- Prozor igre – Prikaz same igre
- Prozor scena – Prozor u kojem se gradi okruženje igre, vizualizaciju igre.
- Prozor hijerarhija – Sadrži popis svih objekata koji se nalaze u igri
- Prozor projekt – Prozor koji raspolaže sa svom imovinom u igri
- Prozor inspektor – Dateljizacija objekata
- Prozor konzola – Prikaz upozorenja, informacija i pogrešaka
- Prozor Assets Store – Direktna poveznica na web stranicu
- Prozor animator – Služi za kreiranje animacija

Proces razvoja igre - ''Anne and Mysterious Planet''

Sam proces razvoja video igre nije težak, ali svakako nije jednostavan. Početnicima predstavlja relativno velik izazov iz razloga što treba shvatiti kako neke funkcije unutar Unity-a rade. Baš iz tog razloga pokušat ću biti što temeljiti u objašnjavanju, kako bi svi mogli shvatiti, te kasnije samostalno razviti svoju video igru.

Za kreiranje ove igre koristit ću već unaprijed neka gotova rješenja, kako bi sama igra bila vizualno puno ljepša. No uz te dijelove koristit ću i svoja vlastita rješenja. Ovaj način izrade video igre ima svojih prednosti i mana.

Ukoliko koristima neka besplatna gotova rješenja, tada sam proces izrade igre je puno lakši, brži i ukoliko nismo dovoljno dobri u animaciji, puno vizualno ljepši. Jedna od većih prednosti ovakvog načina izrade video igre jest brža mogućnost učenja kreiranja vlastite igre, jer puno stvari dobijemo gotovih iz kojih možemo puno toga novog naučiti. Kako se pišu skripte, kako kreirati interaktivni dio igre, kako kretanje radi likova radi, kako je riješena umjetna inteligencija neprijatelja, te još mnoge druge stvari. Iz tih razloga započeti kreiranje video igre iz gotovih dijelova je svakako poželjno. No nedostatak takve vrsti izrade jest ono što je prednost samostalne izrade. A to je da nismo ograničeni na stvari koje su nam dostupne, već možemo kreirati vlastite zamisli. Samostalno odlučujemo o tome kakve će postavke biti u igri, kako ćemo kreirati animacije i tehničke postavke igre. Drugim riječima dobijemo slobodu u kreiranju igre, ali nam je za to potrebno puno više znanja. Naravno možemo kombinirati oboje, te tako izvući najbolje iz obje strane, što je meni osobno i najbolje, jer spajamo najbolje s obje strane.

3D Game Kit

Od mnogo dostupnih stvari, odabrao sam The Explorer: 3D Game Kit kreiran od strane Unity Technologies. 3D Game Kit je skup, odnosno zbirka mehanike, alata, sustava i sredstava za povezivanje igranja bez pisanja koda ili u najlošijem slučaju pisanja veoma malo koda. [3]

On se odnosi na trodimenzionalni svijet koji obuhvaća glavnog lika, neprijatelje, okolinu, te animacije i mehaniku za to. Postoji još i 2D Game Kit koji je veoma sličan ovome, ali je namijenjen za 2D igre, kao što i sam naziv govori. Ovaj paket je dostupan da službenim Unity stranicama, ali zbog lakšeg preuzimanja i stavljanja u vlastiti projekt, preporuča se da se skine direktno preko Assets Store prozora u Unity alatu. Da bismo skinuli paket potrebno je kliknuti na download gumb, te se nakon toga prijaviti u Unity sistem. Kako je ovo dosta velik paket, zauzima 2.2 Gb prostora, tako je potrebno neko vrijeme da se on skine. Nakon što se skinuo, potrebno je sve u uvesti u Vaš projekt.

Klikom na gumb ''Import'', paket će se uvesti u vaš projekt. Vrijedi napomenuti kako će uvažanje ovog paketa izbrisati sve prethodno što se nalazi u projektu, stoga treba biti oprezan. Ovaj proces je dosta dugi, te mu treba vremena

Likovi

Likovi su podijeljeni na dva dijela. Glavni igrači i neprijatelji. Glavni igrač je samo jedan. Glavnog igrača predstavlja ženska osoba imenom Anne. U igri postoji samo jedan lik za odabir. Kako izgleda Anne, može se vidjeti na sljedećoj slici 24. Anne u igri može trčati, skočiti i napadati. Svi njezini pokreti su animirati. To znači da ima animacija za ''Idle'', trčanja, skoka, doskoka, pada s veće visina, te napad. Ona se upravlja pomoću tipkovnice i miša. Anne nisam kreirao samostalno, već je ona besplatno dostupna preko 3D Game Kit-a. To znači da s njom dolaze animacije i skripte za njezino upravljanje. Iako to sve dolazi u paketu, potrebno ih je kreirati. To znači da su dostupni materijali koji su potrebni za kreiranje animacije i slično.



Slika 1. Anne, glavni lik igre

Druga vrsta likova u igri su neprijatelji. Nijedna igra ne bi bila zanimljiva da ih nema. Tako za ovu igru postoje ukupno 3 vrste neprijatelja. Dva manja neprijatelja, od kojeg jedan ima mogućnost napada na daljinu, ispaljivanjem opasne sluzi, dok drugi direktno napada. Treći tip neprijatelja je veliki alfa neprijatelj. Svi neprijatelji su animirani, te imaju mogućnost napada. Kako igra odmiče, odnosno kako ju igra prelazi, tako neprijatelji postaju sve teži za ubijanje. Rastu broji udaraca koji su potrebni da se uništi neprijatelja. Tako na početku su neprijatelji koji se mogu ubiti samo sa jednim udarcem, dok kasnije za to je potrebno i po nekoliko udaraca. Kao što je glavni lik uzet s 3D Game Kit-a, tako su i neprijatelji preuzeti. Kako su to besplatno dostupni materijali, nema potrebe ih ne upotrijebiti. To je još jedna prednost 3D Game Kit-a

Prvi neprijatelj, slika 2, ima mogućnost samo direktnog napada, dok drugi, slika 3, ima mogućnost direktnog napada, ali i mogućnost napada na daljinu pomoću bacanja sluzi. Treća vrsta neprijatelja, slika 4, je neprijatelj koji ima mogućnost napada kao i neprijatelj druge vrste, ali ima dodatno štit koji ga može obraniti, te ga se može napasti jedino s leđa.



Slika 2. Neprijatelj koji direktno napada



Slika 3. Neprijatelj koji može napadati na daljinu



Slika 4. Alfa neprijatelj

Sadržaj u igri

Interaktivni sadržaj svakako doprinosi veliki utjecaj na zanimljivost igre. Stoga je veoma bitno ga imati u svojoj igri. U ovoj igri kao interaktivni sadržaj koristim „cutscene“, kutije koje se mogu razbiti, kutije koje mogu igraći vratiti zdravlje ili mu dodati dodatni život, vrata, koja se mogu otvoriti ali samo pronalaskom određenih kristala, kontrolne točke, te putujućih platformi.

Gameplay igre

Radnja ove igre dešava se na misterioznom planetu na kojeg sleti glavna likinja video igre, te mora pronaći drugu letjelicu, s kojim je stigao njezin kolega, jer se njezina letjelica pokvarila, pri slijetanju na planet. Tijekom njezine potrage, susret će se sa raznim preprekama koje će morati savladati ukoliko želi doći do cilja, odnosno pronaći partnerovu letjelicu. Igra pripada u 3D avanturističku igru igranu i trećem licu. Glavnom liku, bit će dostupno oružje s kojim će se moći obraniti i neprijatelje koji se nalaze u igri, ali i za destrukciju raznih kutija.

Kod svake video igre, najvažnija je dobra ideja. Jer ukoliko imamo dobru ideju kako će video igra izgledati, te kako će on funkcionirat, riješili smo puno

problema koji se mogu kasnije desiti. Stoga je vrlo dobro razmisliti o tome kako bi igra trebala izgleda, tko bi sve trebao biti u njoj, te koji joj je smisao i cilj. Ukoliko je dobra ideja, zamisao igre, tada se ne moramo brinuti i tome da li će ona kasnije biti uspješna ili ne. Naravno da realizacija igre puno ovisi o tome, ali velik utjecaj svakako ima ideja i zamisao, te koliko je ona zabavna za igranje korisnicima, iz razloga što kod samostalnog kreiranja igre, mi ne možemo vizualno parirati velikim budžetnim video igrama koje se nalaze na tržištu, ali možemo parirati sa originalnošću i zabavnošću. Stoga je ovo veoma bitna stavak, te treba uzeti dovoljno vremena kako bi bili sigurni da je ideja prava, te će se drugima svidjeti.



Slika 5. Scena iz igre

Zaključak

Unity razvojno okruženje zajedno sa Visual Studio-em pruža pregršt mogućnosti developerima i korisnicima koji razvijaju video igre. Video igre su nešto nezaobilazno, te ih treba shvaćati ozbiljno. Zbog toga se industrija video igara rapidno razvija, a posljedica toga je da postoji niz dodataka i paketa koji se mogu koristiti u izradi bilo koje vrsti video igre, te bi bilo neozbiljno ih ne iskoristiti, pogotovo što su velika većina besplatna.

Korištenjem već gotovih, besplatnih dijelova i paketa, možemo puno ubrzati sami razvoj video igre, a istovremeno dobiti vizualno predivnu igru. Stoga bi bilo veoma nezahvalno ne ih iskoristiti. Iako razni paketi i dodaci puno pomažu, još uvijek ne mogu puno toga pokriti, te je potrebno dosta

samostalnog rada da se to dovede do zavidne razine u kojem će igrači uživati u igri .

Za kraj rekao bih da kreiranje video igre može biti veoma zabavno, ali istovremeno veoma izazovno i iscrpljujuće. Pogotovo ako ste početnik ili netko tko to ne radi redovito. Programska rješenja veoma napreduju, te je potrebno pratiti u kontinuitetu razvoj igara, jer se brzo može zaostati. Zbog razno raznih paketa, pogotovo vizualnih, danas svako tko ima volje i minimalnog znanja, može s vremenom kreirati neku vrstu video.

Reference

[1] Unity Techologies, Public Relations. Preuzeto 03.07.2021. s https://unity.com
[2] GamesBeat, venturebeat.com, Unity files for IPO (2020), preuzeto 03.07.2021 s https://venturebeat.com/2020/08/24/unity-files-for-ipo-reveals-163-million-loss-for-2019-and-1-5-million-monthly-users/
[3] Unity, Learn.unity.com (2021), preuzeto 07.07.2021. s https://learn.unity.com/project/3d-game-kit
[4] Kenneth Andersen, The Ultimate C# and Unity course for beginners (2021), dostupno s https://www.udemy.compp

Umrežene igre sljedeće generacije

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 daje se pregled izrade i dizajna umrežene igre sljedeće generacije korištenjem alata Unreal Engine i Spatial OS. Govorit će se o računalnim igrama kao mediju i stanju trenutne generacije razvoja računalnih igara. U radu će se govoriti o umreženju simulacije, uspostavi višeserverske arhitekture i dinamičnim upravljanjem interesa. Također će se obraditi i napredne tehnike umreženja i uspostava uništivog okruženja. Za sve koncepte kreiran je prototip koji demonstrira navedene koncepte.

Ključne riječi: Unreal Engine, Spatial OS, računalna igra, umreženo igranje, višeserverska arhitektura, dinamično upravljanje interesima

Uvod

Chris Crawford, utemeljitelj GDC-a, često zvan Socratesom računalnih igara, u svom „zmajevskom“ govoru priča o temeljnoj ideji igara [1]. One su jedinstven medij zbog svoje interaktivnosti. Kako bi se ona mogla adekvatno postići on zaziva na paritet s pravim svijetom (vizualni i auditivni elementi) te simuliranje ljudskog ponašanja. Pošto je simuliranje ljudskog ponašanja tehnički neizvedivo, ne možemo se oslanjati na njega kao podlogu za istinski interaktivni medij. Jedino pravo (a i očito) rješenje je umreženje ljudi, tj. igrača. Ograničenje broja mogućih interakcijskih situacija u računalnoj igri mijenja se od ograničenog broja implementiranih mehanika na praktički neograničeni broj ideja ljudskih bića. Interakcija zato igra važnu ulogu u umreženim igrama. Logički slijedi da će okružje s više igrača imati više interakcija i bit će bogatija raznolikošću u tim interakcijama. Zbog toga je bitno da se broj igrača u umreženoj igri proširi do tehnološki moguće granice.

Ograničenja trenutne generacije

Igre trenutne generacije ograničene su svojom arhitekturom i procesorskim performansama te su igre često dizajnirane upravo oko tih ograničenja. Fortnite, kao glavni predstavnik trenutne generacije računalnih igara, umrežuje 100 igrača i 50000 repliciranih objekata [2]. Fortnite je vrlo detaljna simulacija s umjetnom inteligencijom, uništivim objektima i slično, ali je

tehnološki gledano, ograničena. Na primjer, ne postoje fizički simulirani objekti, svijet je zapravo veoma statičan.

Umreženje simulacije

Fizika je iznimno bitan, temeljni dio svake simulacije koji predstavlja stanje svijeta. Umreženje tog stanja je zapravo temeljni problem koji se mora riješiti. Rješenje mora osigurati da svi objekti na svim umreženim računalima imaju isto stanje. Kako bi to osigurali, fizika mora biti serverski autoritarna. Nadalje, bez obzira na odaziv, moramo održati trenutnu kontrolu. Rješenje se zasniva na činjenici da je simulacija sustav koji ima neke unose te nakon obrade kao rezultat dobivamo stanje svijeta.

Deterministički diskretni korak – slanje unosa

Ovo je koncept u kojem se svakoj simulaciji šalju njeni unosi. Dakle, ako svaka simulacija u isto vrijeme unese iste unose, rezultat bi svugdje trebao biti isti [3]. Ovakvo rješenje je, tehnički gledano, iznimno zahtjevno za izvesti:

- Simulacija mora biti deterministička: isti unosi uvijek daju iste izlaze.
- Simulacija se mora događati u diskretnim koracima, odnosno u vremenski fiksnim otkucajima.
- Kako bi se mogli simulirati sve unose, moramo ih primiti, odnosno čekati unos igrača s najsporijom vezom. Za to je potreban sustav premotavanja stanja simulacije.

Ograničenje detaljnosti simulacije je zapravo u tome koliko je pokreta moguće obraditi i umrežiti odjednom. Na primjer, igra s 32 moguća unosa, odnosno 32 bita binarnih informacija: svaki unos može imati vrijednost 0 ili 1. Ako pretpostavimo 10 gigabitni serverski „budžet“, te 60 pokreta u sekundi, dolazimo do maksimalnog broja od 2282 igrača. Ovu tehniku većinom koriste strateške igre.

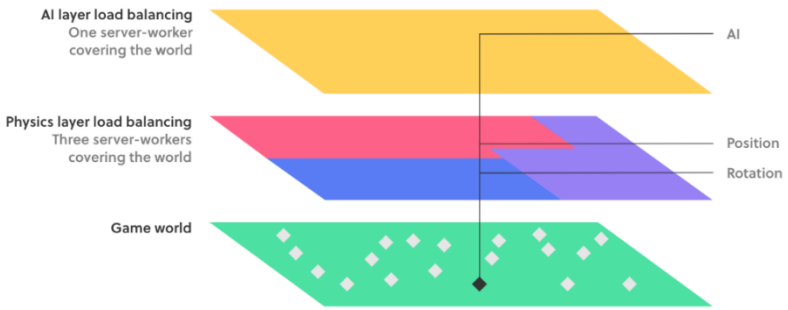
Sinkronizacija stanja – slanje izlaza

Ovo je koncept u kojem šaljemo stanja simulacije, tj. fizičke pozicije svih objekata simulacija [4]. Server provodi simulaciju dok klijenti samo ažuriraju pozicije te provode interpolaciju. Pozicijski podaci su iznimno veliki, te je veliki dio implementacije ove tehnike posvećen smanjivanju količine podataka koji šaljemo. Same pozicijske podatke moguće je smanjiti na 83 bita. Daljnje optimizacije uključuju selektivnu sinkronizaciju pomoću akumulatora prioriteta, koji ažurira pozicije objekata ovisno o njihovoj važnosti za pojedinog klijenta, na primjer, ovisno o udaljenosti. Ovu tehniku koristi većina igara, kao što je Fortnite, te će se koristiti i u ovom radu.

Višeserverska arhitektura

Kada govorimo o višeserverskoj arhitekturi, mislimo na proces obrade jedne simulacije od strane više računala: ovo se još naziva distribuirana simulacija. U ovom radu koristi će se javno dostupna tehnologija Spatial OS [5], jer ima već stabilnu i produkcijski spremnu inačicu. Simulacija se u Spatial OS percipira kroz tri koncepta: svijet, entiteti i komponente. Svijet je izvor kanonske istine za cijelu simulaciju. Sve podatke koje želimo dijeliti između radnika (servera ili klijenata) moramo spremiti u entitete, koji sadrže komponente.

Kako bi se simulacija distribuirala koriste se koncepti slojevitosti i prostornog uravnoteženja opterećenja. Slojevitost je koncept u kojem se grupe komponenata organiziraju po radnicima koji simuliraju svijet [6]. Prostorno uravnoteženje opterećenja je podjela jednog sloja po njegovom prostornom, odnosno geografskom načelu, dakle jedan radnik brine se samo za komponente na određenom dijelu svijeta [7]. Na sljedećoj slici prikazan je primjer gdje na vrhu imamo sloj koji je odgovoran za komponentu upravljača umjetnom inteligencijom, te jedan sloj koji je zadužen za fizičku simulaciju.



Slika br. 1: Slojevitost [6]

Dinamično upravljanje interesom klijenta igre

Pošto želimo optimizirati količinu poslanih podataka svakom klijentu, moramo dinamički upravljati njegovim interesom. Na primjer, bitniji su nam neprijateljski igrači u našem vidljivom polju, nego igrači našeg tima daleko iza nas.

Za ove potrebe koristiti će se komponenta za upravljanje interesom, te će se ona modificirati kada se kontekst promijeni. Ovo se radi dobavljanjem drivera koji nam daje vezu na radnika gdje možemo direktno ažurirati njegov interes.

Interes ćemo modificirati na temelju vidljivog polja (engl. Field Of View), udaljenosti te s obzirom na tim. Kako bi igrači mogli raspoznati između više timova, potrebno je razviti sustav označavanja koji pridodaje entitetu komponentu koja označava njegov tim.

Kako bi se procijenio mogući broj igrača, prvo je potrebno utvrditi koliko interakcija svaki klijent ima po sekundi, što je prikazano u sljedećoj tablici:

Tablica br. 1: Broj interakcija po polju

Udaljenost (eng. Distance)	Vidljivo polje (eng. FOV) - interakcija	Blizina (eng. Vicinity) - interakcija
100m	320	256
350m	480	384
700m	320	256
2000m	256	256
6000m	256	0
19000m	512	0

Kada se primijeni definirani sustav označavanja, broj interakcija se prepolovi na 2472 interakcije po klijentu. S obzirom da je „budžet“ za broj interakcija Spatial OS-a oko 6 milijuna, to nas dovodi do 2427 igrača.

Konfiguracija sloja za umjetnu inteligenciju

Umjetna inteligencija je odličan kandidat za postavljanje na svoj sloj pošto je vrlo skup za procesiranje ali latencijski tolerantan [8]. Sloj je potrebno dodati u konfiguraciju te na njega dodati odgovarajuće klase: upravljač umjetne inteligencije (sustav koji donosi odluke) te menadžer umjetne inteligencije, koji upravlja funkcionalnošću istih. Primijetite da se na ovaj sloj ne dodaje lik umjetne inteligencije (engl. Character) zato što se želi omogućiti igračima preuzimanje kontrole nad likovima kojima upravlja umjetna inteligencija no bez promjene sloja.



Slika br. 2: AI u igri

Vozila

Vozila su izazovan slučaj umrežene fizike, zato jer imaju kompleksno kretanje koje se teško umrežuje. Srž problema je da nam je potrebna trenutna kontrola u okruženju sa odazivom. Rješenje je sustav premotavanja. Dakle, klijent serveru šalje svoje pokrete, oboje izvedu te pokrete, te server klijentu šalje natrag poziciju na kojoj bi on trebao biti u tom trenutku. Klijent gleda u prošlost svojih pozicija te ako se one ne podudaraju, on premota svoje pokrete natrag do tog trenutka, te ponovi sve pokrete koje je napravio nakon tog trenutka. Rezultat je trenutna neposredna kontrola te precizna pozicija. Ovakva implementacija je u ovom radu izvedena nekoliko puta, s „INetworkPredictionInterface“ i „PhysX“ sustavom fizike, s ručno izrađenim algoritmom [9] te naposljetku s „Chaos“ sustavom fizike koji ima ugrađeno premotavanje.

Umrežena simulacija fizike objekata

Kod velikih svjetova s puno fizičkih objekata dolazi do problema s kolizijskim oblicima. Kako bi se uštedjelo na procesorskim performansama, izračuni kolizije su često niske rezolucije, što dovodi do trzanja (engl. Jitter) ili nemirnih objekata. Ovaj problem se rješava jednostavnim algoritmom koji svaki trenutak provjerava je li se objekt značajno pomaknuo (npr. 1 cm) od prošlog trenutka. Ako je, njegova pozicija se prenosi svim klijentima. Ovaj algoritam se nadalje može nadograditi tako da se nakon detektiranja trajnog putovanja uključi umreženje pozicije, jer ono podliježe sustavu akumulatora, što je efikasnije.

Deformacija terena

Deformacija terena se zasniva na tehnologiji proceduralnih oblika. To je skup vrhova i bridova koji se mogu modificirati tijekom izvođenja igre. Teren se sprema kao polje od 100 vektora, te se zahvaljujući serijalizaciji svakom klijentu šalju samo promjene koje su njemu potrebne. Osim toga se algoritam ograničava na način da kolizija s terenom utječe na samo 8 najbližih vektora te se oni mogu kretati samo prema dolje, sa smanjenom preciznošću.

Izgradnja objekata

Izgradnja objekata pruža korisnicima izvor kreativnosti i slobode u oblikovanju svijeta. Sustav je osmišljen tako da se klijentu prvo daje pregled objekata – igrač vidi kako će objekt izgledati sagrađeno, te se igraču daje mogućnost postavljanja više objekata odjednom tako da prate postavljenu liniju. Za ovo je potrebno izgraditi menadžere za vizualni pregled s korisničke

strane te menadžera za postavljanje na serverskoj strani. Naposljetku je potrebno razviti alat koji izgradi sam objekt te i sam izgradivi objekt.

Uništivi objekti

Uništivi objekti su ključni za stvaranje dinamičnog svijeta. U ovom radu koristi se „PhysX APEX“ sustav za stvaranje uništivih objekata, uz pomoć proširenja „DENT“ [10] za njegovo umreženje. Treba zamijetiti da je sustav vrlo ograničen jer nije namijenjen za umreženje te se preporuča sljedeća konfiguracija: ne simulirati krhotine nakon što su odvojene od glavnog objekta i sve krhotine trebaju biti direktno povezane na temeljnu.



Slika br. 3: Sinkronizacija uništivih objekata u igri

Simulacija projektila i balistike

Projektili se mogu izvesti na dva načina: linijskim tragom i fizičkim projektilima. Linijski trag nije povoljan za kreirano okruženje jer može doći do slučaja da ne postoji interes nad metom, te ona neće biti vidljiva što znači da linijski trag neće nikada pogoditi tu metu. Fizički projektili su puno bolja opcija jer je moguće pratiti njihovo putovanje kroz prostor te primjereno obavijestiti klijente o njihovom postojanju ako nad njime imaju interes.

Balistički sustav je simulacija probijanja i štete koju bi pogodak projektila uzrokovao nekom objektu. Algoritam se zasniva na događaju preklapanja kolizije. Dakle, ako do preklapanja dođe, projektilu jednostavno smanjimo brzinu dok objektu nadodamo štetu. Izračun količine probijanja može biti veoma kompleksan te se zato u ovom radu koristi pojednostavljena verzija koja ovisi o materijalu objekta, metku te njegovoj brzini i težini.

Funkcionalnost igre i meča

Jednom kada je kreirana dobra podloga za dobre performanse i set značajki za stvaranje zanimljivog i „bogatog“ umreženog svijeta, potrebno je dizajnirati funkcionalnosti igre koje u fokus stavljaju interakciju među igračima. Postoje dvije dominantne vrste umreženih igara: one koje se temelje na mečevima, te one koje se temelje na trajnim svjetovima.

Dizajn igara sa trajnim svjetovima je iznimno zahtjevan te će vjerojatno zahtijevati iznimno dobru skalabilnost, veliki broj servera te iznimno veliki svijet. Ovo je prvobitno vrlo skupo za održavanje te je teško osmisliti monetizacijski model koji bi to podržavao. Osim toga, gotovo je nemoguće stvoriti dovoljno sadržaja za ovakve tipove igara.

S obzirom na današnja tehnološka ograničenja, možda je ipak bolje fokusirati se na izradu videoigara koje će stvarati manje ali stabilnije i kvalitetnije igrače iskustvo. Iskustva kreirana na ovaj način, uz primjenu koncepata koji su opisani u ovom radu, će i dalje, tehnološki gledano, dati bolje mogućnosti od trenutne generacije računalnih igara. Stoga je u ovom radu fokus bio na izradu prototipa igre koja se temelji na mečevima. Ograničenje je postavljeno na 200 igrača. Postoji niz besplatnih servisa u ovom rangu [11], s jednim dodatnim slojem za AI.

Uvjeti pobjede trebaju biti jasni svim igračima te meč mora biti vremenski ograničen. Ovo se postiže sustavom bodovanja i setom kontrolnih točaka čije posjedovanje doprinosi bodovima tima.

Da bi poboljšali udruživanje, interakciju te organizaciju i tijek igre, uveden je i koncept lanca zapovijedanja. Na vrhu stoji zapovjednik koji svojim odredima zadaje naredbe. Izvedbom zadanih naredbi dobivaju se motivacijski bodovi. Motivacijski bodovi se mogu iskoristiti na razna unaprjeđenja u igri, kao što su bolja oružja i slično. Na taj način se osigurava eskalacija bitke. Isti sustav može se koristiti i za progresiju između bitki, kao što je izvedeno u računalnoj igri „Call Of Duty 4“. Kako bi se održao realizam i raširile taktičke mogućnosti, uvodi se logistički sustav. Igrači se mogu ponovo stvoriti kada umru jedino ako za to postoje resursi. Naposljetku, bilo koju ulogu može preuzeti umjetna inteligencija, a svakog lika kojim upravlja umjetna inteligencija može preuzeti bilo koji igrač te se tako odmah vratiti u bitku ako umre.

U praktičnom prototipu implementirane su temeljne značajke: pravila igre, kontrolne točke te uvjeti pobjede.

Reference

[1] C. Crawford, »Chris Crawford Dragon Speech "I Had a Dream",« 14 06 2017. [Mrežno]. Available: https://www.youtube.com/watch?v=CBrj4S24074. [Pokušaj pristupa 24 06 2021].

[2] Epic Games, »Networking in 4.20: The Replication Graph | Feature Highlight | Unreal Engine Livestream,« 19 07 2018. [Mrežno]. Available: https://youtu.be/CDnNAAzgltw?t=575. [Pokušaj pristupa 24 06 2021].

[3] G. Fiedler, »Deterministic Lockstep,« 29 09 2014. [Mrežno]. Available: https://gafferongames.com/post/deterministic_lockstep/. [Pokušaj pristupa 24 06 2021].

[4] G. Fiedler, »State Synchronization,« 05 01 2015. [Mrežno]. Available: https://gafferongames.com/post/state_synchronization/. [Pokušaj pristupa 24 06 2021].

[5] Improbable, »Multiplayer Networking,« 2021. [Mrežno]. Available: https://www.improbable.io/multiplayer-networking. [Pokušaj pristupa 24 06 2021].

[6] Improbable, »Layers,« 06 2020. [Mrežno]. Available: https://documentation.improbable.io/spatialos-overview/docs/layers. [Pokušaj pristupa 24 06 2021].

[7] Improbable, »Workers and load balancing,« 05 2020. [Mrežno]. Available: https://documentation.improbable.io/spatialos-overview/docs/workers-and-load-balancing . [Pokušaj pristupa 24 06 2021].

[8] Improbable, »Offloading,« 12 08 2020. [Mrežno]. Available: https://documentation.improbable.io/gdk-for-unreal/docs/5-offloading-in-game-tutorial. [Pokušaj pristupa 24 06 2021].

[9] S. Pattuzzi, »Unreal Multiplayer Master: Video Game Dev In C++ Course,« 2021. [Mrežno]. Available: https://www.gamedev.tv/p/unrealmultiplayer. [Pokušaj pristupa 24 06 2021].

[10] ByteSumPi LTD , »DENT - Networked Destruction - APEX,« 1 03 2017. [Mrežno]. Available: https://www.unrealengine.com/marketplace/en-US/product/dent-destructible-environment?sessionInvalidated=true. [Pokušaj pristupa 24 06 2021].

[11] Improbable, »Networking service pricing,« 12 2020. [Mrežno]. Available: https://documentation.improbable.io/spatialos-overview/docs/managed-networking-services#section-networking-service-pricing. [Pokušaj pristupa 24 06 2021].

Izrada strateške igre u stvarnom vremenu u programskom alatu Unity

Deni Baban i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
deni.baban@gmail.com, mlkoneck@foi.hr

Sažetak

U ovom radu je opisan razvoj i implementacija osnovnih mehanika strateških igara u stvarnom vremenu u programskom alatu Unity. Strateške igre u stvarnom vremenu stavljaju fokus na: kontrolu većega broja jedinica u borbi s protivničim igračima, prikupljanju resursa i korištenju raznih vještina za stvaranje prednosti. Također je opisana implementacija jednostavne umjetne inteligencije koja je u mogućnosti izvršiti osnovne zadatke strateške igre u stvarnom vremenu.

Ključne riječi: Unity, računalne igre, strateška igra u stvarnom vremenu, algoritmi, programiranje

Uvod

Izrada računalnih igara je često dugotrajan i kompleksan proces koji zahtjeva znanje iz većeg broja područja kao što su dizajn, programiranje, animacija, modeliranje, pisanje, kinematografija i dr., no zahvaljujući modernim alatima i njihovoj pristupačnosti, individualnim kreatorima je znatno olakšano stvaranje vlastitih igara. Programski alat Unity [3] pruža mnoge ugrađene mogućnosti za razvoj bilo koje vrste računalne igre poput upravljanja fizikom, izradu modela, shadera, vizualnih efekata i njihovu distribuciju na većinu modernih platformi [1]. Primjena aspekata objektno orijentiranog programiranja je uvelike iskoristiva za razvoj strateške igre u stvarnom vremenu jer mnogi elementi imaju određene zajedničke karakteristike. Također, potrebno je poznavati različite algoritme, a također potrebna je i kreativnost u pogledu dizajniranja igre i vizualnih elemenata igre: jedinica, okoliša i dr.

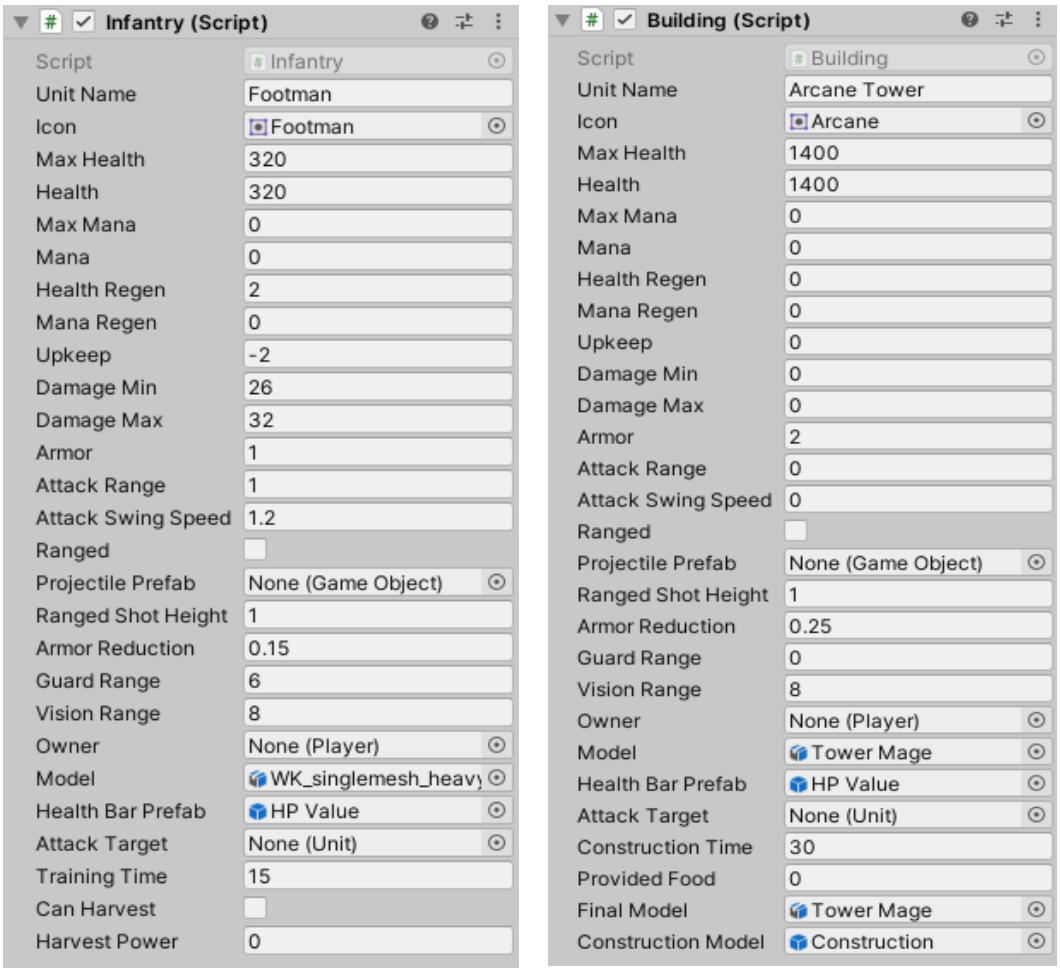
Elementi strateške igre u stvarnom vremenu

Žanr strateških igara u stvarnom vremenu (engl. Real Time Strategy) opisuje igre u kojima igrači igru započinje s malim brojem jedinica, razvija ekonomiju i gradi infrastrukturu za izgradnju vojske s kojom onda pokušava savladati protivnika. Temelj razvoja je neki resurs koji je ograničen: zadatak igrača je

donositi odluke kako na najbolji mogući način raspodijeliti resurse za unaprjeđenje ekonomije, potporne infrastrukture i raznih jedinica u igri. Ograničeni resursi diktiraju dinamiku napretka u igri. Izgradnja građevina otključava nove mogućnosti i često se koristi i za razvoj tehnologije i nadogradnju jedinica. Jedinice u strateškim igrama također mogu imati sposobnost korištenja posebnih vještina ili magija raznih efekata koje pomažu igraču u određenim aspektima igre. Jedan od populanih elemenata strateških igara je Fog of War, tj. efekt sakrivanja dijelova mape koji trenutno nisu istraženi. Igra završava onda kada je svim protivnicima igrača uništena glavna zgrada i jedinice radnika koje bi mogle napraviti novu. U nastavku će se opisati implementacija osnovnih mehanika jedne tipične RTS igre [2].

Jedinice i građevine

Glavni fokus strateških igara su jedinice vojske i građevine koje igrači mogu kreirati i kontrolirati u svojoj borbi protiv ostalih igrača. Svaka jedinica je definirana sa svojim atributima poput naziva, životnih poena, jačine napada, brzine kretanja, popisa vještina koje može koristiti, popisa jedinica koje može trenirati i mnogi drugi.



Slika br. 1: Komponente jedinica i građevina u igri

Jedinice sadrže nekoliko osnovnih mehanika igranja poput kretanja po mapi, borbe, korištenja vještina i treninga novih jedinica. Kretanje po mapi i traženje puta se u alatu Unity može vrlo lako ostvariti korištenjem ugrađenih NavMesh i NavMeshAgent komponenti gdje MavMesh predstavlja teren ili podlogu po kojoj je moguće kretati se, a NavMeshAgent predstavlja agenta koji pomiče objekt po MavMesh-u prema zadanoj destinaciji najkraćim putem.

Borba između jedinica se uz korištenje vještina može i odvijati pomoću osnovnih napada poput udaraca na blizinu ili ispaljivanja projektila prema protivnicima na daljinu. Svaka jedinica može dobiti naredbu od igrača da napadne određenu metu ili može automatski početi napadat protivničke jedinice koje se približe do određene točke udaljenosti. Napadi rade određenu količinu štete neprijateljskim jedinicama, imaju definiranu udaljenost sa koje mogu započeti napadati a svaki napad ima i svoju brzinu, tj. vremenski period u kojem se može izvesti jedan napad. Jedinice također imaju atribut Armor koji umanjuje nadolazeću štetu za određenu količinu.

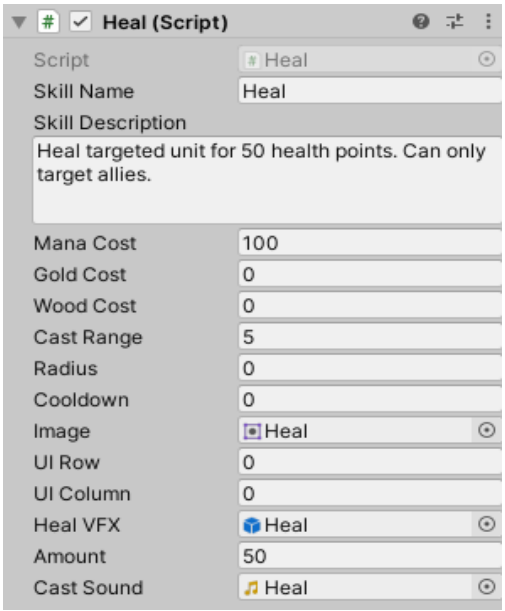
```
void Update() {
    if(attackTarget != null && agent.remainingDistance < 0.2) {
        if(Vector3.Distance(gameObject.transform.position,
           attackTarget.transform.position) > attackRange) {
            agent.SetDestination(attackTarget.transform.position);
        } else {
            transform.LookAt(attackTarget.transform);
            agent.ResetPath();
            if (attackCooldown <= 0) {
                attackCooldown = attackSwingSpeed;
                if (ranged) RangedAttack();
                else MeleeAttack();
            } else attackCooldown -= Time.deltaTime;
        }
    }
}
```

Trening novih vojnih jedinica se izvodi na način da svaka građevina koja ima sposobnost treniranja drugih jedinica ima red čekanja u koji se mogu dodati nove jedinice gdje čekaju da završi vrijeme treninga.

```
void Update() {
    if (infantryTrainingQueue.Count > 0) {
        trainingTime += Time.deltaTime;
        if(trainingTime >= infantryTrainingQueue[0].trainingTime) {
            Infantry topInfanty = infantryTrainingQueue[0];
            trainingTime -= topInfanty.trainingTime;
            SpawnInfantry(topInfanty);
            infantryTrainingQueue.RemoveAt(0);
        }
    }
}
```

Posebne vještine jedinica su efekti koje igrač može iskoristiti kako bi nanio štetu neprijateljskim jedinicama ili pomogao vlastitim jedinicama. Zbog načina na koji mehanike igre funkcioniraju, izgradnju zgrada i trening novih jedinica također možemo smatrati vještinama, dok primjerice jedinica klase „Mage“ može poslati vatrenu kuglu prema protivnicima i napraviti veliku štetu ili jedinica klase „Priest“ može izliječiti određeni broj životnih bodova prijateljskim jedinicama. Vještine imaju niz koraka koji se trebaju izvesti određenim redosljedom što se može elegantno implementirati korištenjem Template Method uzorka dizajna. Prvo je potrebno provjeriti je li vještina sprema na korištenje i ima li jedinica dovoljno resursa (Cooldown, Mana i sl.), nakon čega se traže validne mete od igrača, troše resursi, izvršava sam efekt vještine koje one nadjačavaju sa vlastitom implementacijom te konačno aktivira Cooldown period.

```
public override void Cast() {
    AudioSource.PlayClipAtPoint(castSound,
                                castingUnit.transform.position);
    Instantiate(healVFX, target.transform.position, Quaternion.identity);
    target.health += amount;
}
```



Slika br. 2: Komponenta vještine u alatu Unity

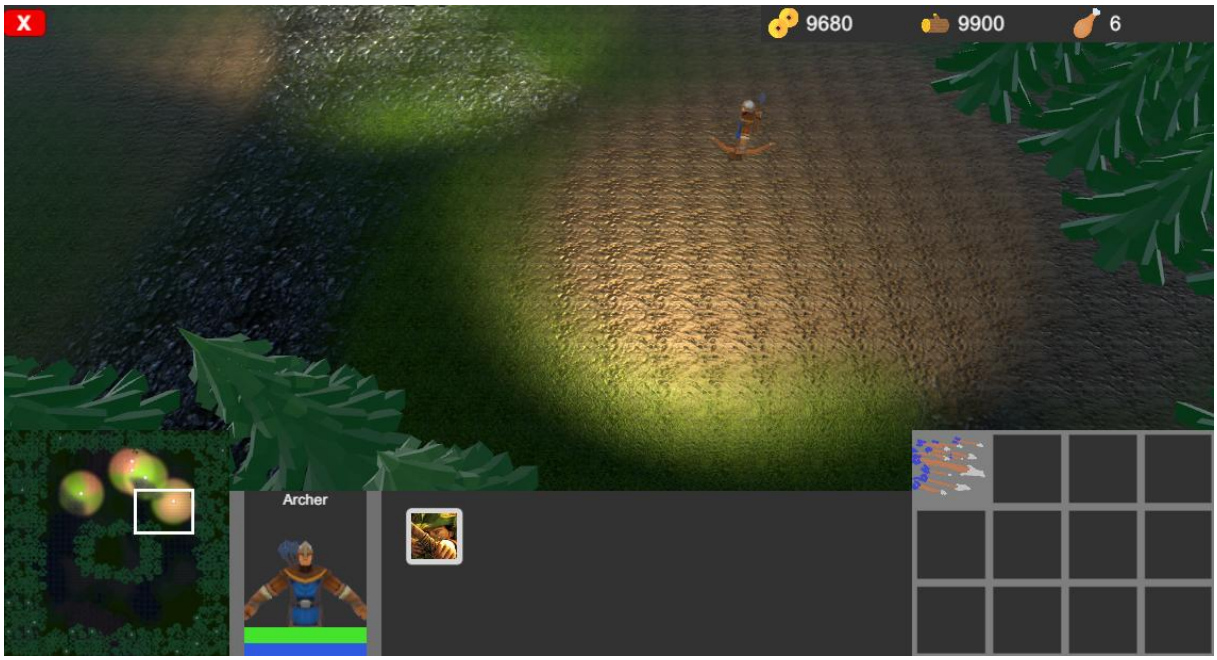
Grafičko sučelje i kursor

Jako važna komponenta svake videoigre je korisničku sučelje koje treba biti informativno i jednostavno, tj. HUD (eng. Heads Up Display) koji igraču pruža pristup svim bitnim informacijama za bolje donošenje odluka prilikom igranja. Sučelje prikazuje trenutno stanje igračevih resursa, kartu, prikaz trenutno odabranih jedinica i tipke s njihovim vještinama. Karta prikazuje pozicije

prijateljskih jedinica kao i vidljive neprijatelje, a moguće je i kontrolirati kameru klikom na točku karte. Također je moguće vidjeti model odabrane jedinice i njegove životne bodove.



Slika br. 3: Selekcijski prozor građevine sa treningom jedinica



Slika br 4: Korisničko sučelje I pogled na igru

Korisnik kontrolira sve aspekte igre pomoću miša i opcionalno pomoću tipkovnice, stoga je potrebno implementirati razna stanja kursora i sve moguće akcije u njima. Moguće je pomicati kameru pomicanjem kursora na rubove ekrana, odabirati jedinice, koristiti korisničko sučelje, odabirati mete vještina i pozicije na terenu. Funkcionalnosti kursora su implementirana pomoću State uzorka dizajna i sadrži normalno stanje za odabiranje jedinica, stanje za ciljanje jedinice, stanje za odabir područja i stanje za izgradnju građevina. Odabir područja pretvara kursor u 3D grafički prikaz lokacije koja će biti obuhvaćena efektom.

Umjetna inteligencija

Jedan od kompleksnijih aspekata implementacije strateške igre u stvarnom vremenu je izrada umjetne inteligencije koja bi mogla igrati protiv igrača što je vidljivo u iskazanom interesu za ovo područe i od strane poznatih globalnih kompanija kao OpenAI [4]. Ovaj projekt sadrži vrlo jednostavnu implementaciju umjetne inteligencije koja se temelji na odabiru prioriteta izgradnje. Sustav je podjeljen na dva odvojena nezavisna dijela: sustav za izgradnju jedinica i građevina te sustav za upravljanje jedinicama i borbu. Sustav za izgradnju jedinica i građevina prolazi kroz sve jedinice koje trenutno posjeduje, prebrojava radnike, obrambene tornjeve i računa snagu vojske te na temelju tih podataka prolazi kroz niz uvijeta poredanih po važnosti i odabire prvi koji zadovoljava. Prioriteti mogu biti: trening novih radnika, izgradnja glavne zgrade, izgradnja tornjeva, trening vojske, izgradnja farmi i sl.

Tablica br. 1: Prioriteti izgradnje

Broj	**Opis prioriteta**
0	Treniranje novih jedinica radnika
1	Izgradnja glavne zgrade
2	Izgradnja farmi
3	Izgradnja zgrade Barracks
4	Izgradnja obrambenih tornjeva
5	Izgradnja zgrade Stable
6	Izgradnja zgrade Blacksmith
7	Izgradnja zgrade Arcane Tower
8	Treniranje vojske

```
if (workersCount <= 0) priority = 0;
else if (!unitNames.Contains("Keep")) priority = 1;
else if (workersCount < 4) priority = 0;
else if (player.TotalUpkeepCost() <= 5) priority = 2;
else if (!unitNames.Contains("Barracks")) priority = 3;
else if (armyStrength < 14) priority = 8;
else if (towerCount < 1) priority = 4;
else if (!unitNames.Contains("Stable")) priority = 5;
else if (armyStrength < 34) priority = 8;
else if (!unitNames.Contains("Blacksmith")) priority = 6;
else if (!unitNames.Contains("Arcane Tower")) priority = 7;
else if (armyStrength < 50) priority = 9;
else if (towerCount < 3) priority = 4;
else priority = 8;
```

Drugi dio sustava služi za slanje borbenih jedinica u napad protiv ostalih igrača. Svaka jedinica zasebno zna koristiti svoje vještine i odabrati metu na kojoj će ju iskoristiti. Sustav također određuje točke interesa prema kojima će fokusirati vojne jedinice i započeti borbu sa svim protivnicima na koje naiđe.

Umjetna inteligencija također ima nekoliko pomoćnih funkcija poput algoritma za odabir lokacije izgradnje nove građevine.

Zaključak

Kroz ovaj rad su ukratko opisane glavne ideje i funkcionalnosti koje se nalaze u strateškim igrama u stvarnom vremenu. Prikazane su neke od mogućih implementacija pomoću programskog alata Unity i njegovih ugrađenih komponenti koje pruža korisnicima. Svi aspekti igre su lagano proširivi i omogućuju lagano, brzo i efikasno dodavanje novih jedinica, građevina, vještina ili mapa kao i izmjenu već postojećih. Unity se može koristiti za brzu izradu prototipa, kvalitetne izrade ispoliranih aplikacija i ima aktivne online sadržaje, forume i resurse za sve razine developera.

Reference

[1] Unity User Manual. Dostupno na: https://docs.unity3d.com/Manual/index.html. [Pristupljeno: 1. Prosinca 2021].
[2] Making a RTS game in Unity (C#), Mina Pêcheux. Dostupno na: https://medium.com/c-sharp-progarmming/making-an-rts-game-in-unity-91a8a0720edc. [Pristupljeno: 1. Prosinca 2021].
[3] Unity (game engine), Wikipedia. Dostupno na: https://en.wikipedia.org/wiki/Unity_(game_engine). [Pristupljeno: 1. Prosinca 2021].
[4] OpenAI. Dostupno na: https://openai.com/ [Pristupljeno: 1. Prosinca 2021.]

Izrada „shoot 'em up“ igre u programskom alatu Unity

Erik Baukovac i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
ebaukovac@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu opisan je proces izrade prototipa „shoot 'em up“ videoigre u programskom alatu Unity. Nakon kratkog uvoda, opisani su korišteni alati u izradi prototipa videoigre, opisuje se implementacija osnovnih mehanika igranja videoigre te je opisan i proces izrade prototipa.

Ključne riječi: Shoot 'em up, Unity, Visual Studio, C#, Stargazer, programiranje, algoritmi

Uvod

Danas možemo vidjeti strelovit napredak u produkcijskoj kvaliteti razvijenih videoigara. Svi aspekti su s vremenom tehnološki postajali sve bolji: grafika je sve veće i veće rezolucije i kvalitete, zvuk je dosegao visoku razinu kvalitete kako tehnološki tako i umjetnički, korisnička sučelja su pregledna i jednostavna za korištenje, mehanike videoigara su vrlo fluidne i iskustvo igranja videoigara je dostiglo zavidne razne kvalitete. Kako na tržištu rastu očekivanja u pogledu kvalitete videoigre, tako su videoigre sve bolje dizajnirane i ulaže se veliki napor kako bi se ostvarila očekivanja igrača.

S vremenom, razvili su se i brojni alati za razvoj videoigara koji omogućavaju daleko brži i jednostavniji razvoj videoigara. Ti alati već imaju gotova rješenja za mnoge osnovne stvari koje svaka videoigara treba imati i netko tko razvija videoigru se može više fokusirati na sami umjetnički dio a ne mora biti toliko opterećen tehnološkim dijelom. Jedan od takvih alata je i korišten programski alat u ovom radu a to je programski alat Unity [1].

Žanr videoigara „Shoot 'em up“ (također poznat i pod nazivom „shmups“) je podžanr videoigara pucanja, što je podžanr akcijskih videoigara [2]. Najčešći format videoigre ovog žanra dolazi u obliku letjelice kojom upravlja igrač te neprijatelja koji dolaze iz suprotnog smjera. Osnovne mehanike igranja se temelje na različitim vrstama kretnji, kako neprijatelja tako i različitih vrsta projektila. U kreiranom prototipu se implementiraju upravo ove temeljne mehanike.

Korišteni alati

Za izradu prototipa koji je nazvan „Stargazer“ korišten je programski alat Unity a za pisanje programske logike videoigre korišten je programski alat Visual Studio [3]. Unity je napisan u programskom jeziku C++, nekada je nudio podršku za pisanje skripti u programskim jezicima C#, Boo ili Javascript no danas je programsku logiku moguće pisati u programskom jeziku C# a također su uveli i mogućnost vizualnog skriptiranja u Boltu [4]. Izrađene skripte se vežu na objekte u obliku komponenti i na taj način je moguće „oživjeti“ elemente koji sačinjavaju videoigru.

Osnovne mehanike

Tipična videoigra žanra „Shoot 'em up“ se temelji na principu da je igrač okrenut prema gore te da u tom smjeru puca na neprijatelje (od tud i naziv žanra). Razina se obično pomiče prema dolje te igrača pokušavaju zaustaviti neprijatelji koji dolaze u različitim naletima. Neprijatelji se često kreću u raznim smjerovima te također uzvraćaju paljbu na različite načine. Zadaća igrača je prepoznati uzorke kretanja neprijatelja, uzorke njihovog pucanja te ih eliminirati a pritom izbjegavati njihove projektile. Igrač na početku raspolaže određenim brojem životnih bodova te ima neko osnovno oružje. Česta mehanika koja se koristi u ovom žanru je i mogućost nekog specijalnog napada. U kreiranom prototipu igrač igru započinje s tri života, jednim osnovnim oružje a također ima i jednu specijalnu vrstu napada.

Još jedna standardna mehanika je vraćanje izgubljenih životnih bodova tijekom prelaska razine. U kreiranom prototipu igrač životne bodove može vratiti skupljanjem posebnih baterija. Igrač gubi životne bodove ako se sudari s neprijateljem ili ako ga pogodi projektil koji ispucava neprijatelj. Također, tipično za ovaj žanr je da postoje različite vrste oružja koje imaju različite putanje napada. Ta mehanika je implementirana i u kreiranom prototipu.

U kreiranom prototipu kreirane su četiri vrste neprijatelja: brzi neprijatelji, neprijatelji normalne brzine kretanja, neprijatelj koji ima više životnih bodova (tzv. tenk) i glavni neprijatelj. Brzi neprijatelj ima veću brzinu kretanja ali zato ne može ispaljivati projektile i ima manju količinu životnih bodova. Neprijatelj normalne brzine je standardni neprijatelj koji se kreće prosječnom brzinom ali ima nešto malo više života te može pucati prema igraču. Tenk ima više životnih bodova ali se zato sporije kreće i sporije puca. Glavni neprijatelj dolazi na kraju razine te je po osnovnim karakteristikama jači od ostalih standardnih neprijatelja i predstavlja poseban izazov za savladati.

Izrada igre

Prva scena, kao i kod većine igara ovog tipa, je scena u kojoj igrač odabire pokretanje razine videoigre. Osim pokretanja razine, tu se nalazi i gumb za izlazak iz videoigre.



Slika br. 1: Početni izbornik

Kada igrač započne igru učitava se prva razina igre. U gornjem lijevom kutu ekrana su prikazani životni bodovi igrača te ukupni bodovi koje igrač skuplja ubijanjem neprijatelja. U desnom gornjem kutu igrač može vidjeti koliko ima specijalnih napada. Na početku razine instancira se predložak svemirskog broda kojim upravlja igra te se on stvara u sredini na dnu ekrana. Također, započinje i korutina koja je zaslužna za stvaranje neprijateljskih jedinica. U svakom predviđenom trenutku za stvaranje neprijatelja se stvara nasumični neprijatelj iz predodređene grupe neprijatelja. Pozadina se također lagano pomiče te na taj način dobivamo efekt prostora, tj. efekt paralakse.



Slika br. 2: Glavna scena

U nastavku se nalazi funkcija koja je zadužena za kretanje igrača i rad s oružjima.

```
void Update()
{
	ammo[0] = 100;
	body.AddForce(new Vector2(Input.GetAxis("Horizontal") * speed, 0));
	body.AddForce(new Vector2(0, Input.GetAxis("Vertical") * speed));

	if (Input.GetKey(KeyCode.Keypad1))
		selectedWeapon = 0;
	if (Input.GetKey(KeyCode.Keypad2) && ammo[1] > 0)
		selectedWeapon = 1;
	if (Input.GetKey(KeyCode.Keypad3) && ammo[2] > 0)
		selectedWeapon = 2;
	if (selectedWeapon != 0 && ammo[selectedWeapon] == 0)
		selectedWeapon = 0;

	if (Input.GetKey(KeyCode.Space) && delay > 120)
		Shoot();
	delay++;

	if (Input.GetKey(KeyCode.KeypadEnter) && bomb==1)
		NukeEm();
}
```

Na početku metode koja se izvršava svaku sličicu iscrtavanja videoigre broj metaka igrača postavljamo na 100. Predviđeno je da igrač na početku ima osnovno oružje koje ima beskonačno metaka stoga na početku igrač ne može ostati bez municije. Nakon toga se primjenjuje sila nad igračem u slučaju da igrač pokušava ići lijevo ili desno što su dozvoljeni smjerovi kretanja. Nakon toga u funkciji se nalazi nekoliko selekcijskih uvjeta koji služe za promjenu oružja: ako je igrač odabrao promjenu vrste oružja te postoji municija za tu vrstu oružja, tada će se dopustiti promjena oružja igrača. Tu se nalazi i selekcija koja provjerava pokušava li igrač ispucati metak. Ako je pritisnuta razmaknica te ako je prošlo dovoljno vremena od prethodnog ispucavanja metka (time se definira brzina pucanja) tada se poziva funkcija za ispucavanje metaka. Posljednje, na kraju funkcije se nalazi i uvjet kojim se provjerava pokušava li igrač ispucati posebni napad. Ako igrač ima raspoloživi specijalni napad, tada se poziva funkcija za ispucavanje specijalnog napada.

```
void Shoot()
{
	delay = 0;
	if (selectedWeapon==1)
	{
	Instantiate(bullet[selectedWeapon], bulletSpawn.transform.position,
Quaternion.identity);
```

```
      Instantiate(bullet[selectedWeapon], bulletSpawn.transform.position +
new Vector3(0.3f,0), Quaternion.identity);

      Instantiate(bullet[selectedWeapon], bulletSpawn.transform.position +
new Vector3(-0.3f,0), Quaternion.identity);
      }
      else
      Instantiate(bullet[selectedWeapon], bulletSpawn.transform.position,
Quaternion.identity);

      ammo[selectedWeapon]--;
}
```

Metoda Shoot() je zadužena za instanciranje metka na poziciji igrača. Ovisno o odabranoj vrsti oružja, instancirat će se odgovarajući projektil te će se umanjiti broj metaka za to oružje. Varijabla delay je zadužena za definiranje brzine ispucavanja metaka. Njenu vrijednost prilikom ispucavanja postavljamo na 0 a prilikom svakog iscrtavanja sličice videoigre vrijednost te varijable se povećava za 1. Kada varijabla dođe do vrijednosti 120, igrač ima mogućnost ispucavanja idućeg metka.

Zaključak

Izraditi videoigru nije jednostavan poduhvat no u svakom slučaju je zanimljivo i poučno raditi videoigre jer se na vrlo zanimljiv i interaktivan način mogu učiti različiti koncepti programiranja, različiti algoritmi i različita programska logika. Prilikom razvoja videoigara potrebno je unaprijed planirati i dizajnirati igru kako bi se postigao željeni ishod. Programski alat Unity je vrlo pristupačan, ima jednostavno sučelje za korištenje i odličan je alat za razvoj videoigara, kako za početnike tako i za profesionalne svrhe. S obzirom da postoji mnoštvo edukacijskih materijala, lako je započeti učiti programirati videoigre i kreirati svoje prve prototipe videoigara [5]. Za izradu cjelovite igre ipak je potrebno je uložiti daleko više vremena i truda. Kreirani prototip je početna inačica videoigre koja se lako može nadograditi novim mogućnostima kao što je dodavanje više različitih vrsta neprijatelja koji će se ponašati na nove i jedinstvene načine. Također, lako je nadograditi postojeća oružja ili kreirati nova. Posljednje, lako je moguće dodati i dodatne razine koje bi trebale biti dizajnirane na način da ne budu niti prelagane pa da igraču bude dosadno, niti preteške kako igrača ne bi frustrirale i dovela do odustajanja igranja videoigre.

Reference

[1] Unity. (17.11.2021.) Dostupno na: https://unity.com/
[2] Shoot 'em up. Wikipedia. (17.11.2021.) Dostupno na: https://en.wikipedia.org/wiki/Shoot_%27em_up

[3] Microsfot Visual Studio. (17.11.2021.) Dostupno na: https://visualstudio.microsoft.com/
[4] J. Brodkin: How Unity3D Became a Game-Development Beast. (17.11.2021.) Dostupno na: https://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/
[5] Learning C# and coding in Unity for beginners- Unity, Dostupno: https://unity3d.com/learning-c-sharp-in-unity-for-beginners#:~:text=The%20language%20that%27s%20used%20in,variables%2C%20functions%2C%20and%20classes

Izrada 2D platformera u programskom alatu Unity

Petar Mihalić
Fakultet organizacije i informatike, Sveučilište u Zagrebu
pmihalic@foi.hr

Sažetak
Tema ovog rada je izrada vlastite igre u programskom alatu Unity. Vrsta igre je 2D platformer te se prije same izrade igre upoznajemo s potrebnim alatima za izradu projekta i sa žanrom igre. Pregledom poznatih 2D igara iz povijesti i današnjice te istraživanjem o načinima izrade i različitim rezultatima koje je moguće ostvariti uz pomoć odabranih alata odlučio sam se za izradu 2D platformera korištenjem spriteova (eng. sprites) umjesto pločica (eng. tiles). Nakon odabira načina rada definirana je tema igre u kojoj imamo glavnog agenta kojim upravlja igrač čija je misija uništiti neprijatelje, izbjeći metke i ponore te pokupiti ukradene stvari tijekom noćne misije.

Ključne riječi: 2D, platformer, Unity, skripte, dizajn, animacija

Uvod

Prije izgradnje same igre prikazat ću povijest, utjecaj i trenutnu popularnost 2D platformera prisjećajući se nekih od najpopularnijih klasika pa sve do današnjih predivnih platformera čija slava leži na kvalitetnoj implementaciji i odličnoj priči. Naravno uz ideju potrebni su razni programski alati koji omogućuju realizaciju ideja u gotove proizvode, ovdje ću ukratko opisati korištene alate, od kojih je glavni Unity, snažan i popularan alat za izradu ne samo 2D već i najvećih 3D hitova u industriji igara. Dizajn i ilustracije bit će izrađene u Photoshop alatu dok će skripte za upravljanje i ponašanje dijelova igre biti napisane u C# jeziku koristeći Visual Studio. Kod izgradnje same igre prikazat ću dizajniranje, animiranje, implementaciju i povezivanje prvo okoline nivoa a zatim glavnog heroja, neprijatelja te ostalih elemenata i interaktivnih objekata. Objasnit ću programsku logiku kretanja, interakcije, borbe i ostalih funkcionalnosti implementiranih u igru. Za završavanje igre dodat ću zvukove, te naravno potrebno korisničko sučelje.

Unity i pomoćni programi

Tvrtka Unity Techologies je objavila prvu verziju Unity-a, razvojnog okruženja za izradu igara, 2005. godine, no razgovori i rad na alatu sežu sve do 2002. godine kada su se dvojca programera našla na forumu pokušavajući pronaći rješenje za svoje individualne projekte. Odlučili su spojiti snage, maknuti fokus

sa svojih projekata te stvoriti novo razvojno okruženje koje će kasnije postati jedno od najpopularnijih i najkorištenijih alata za izradu 2D i 3D igara. [1][2] Unity danas koristi više od 2.5 milijuna registriranih programera, što ga stavlja u sam vrh sa ostalim dobro poznatim i respektabilnim alatima poput Unreal Engine-a i Godot-a [3]. Jedan od razloga zašto je Unity toliko popularan danas je potpora za različite platforme, iako je prvo osmišljen i pušten u rad samo na Mac OS X platformi, danas je proširen na čak 25 različitih platformi [3]. 2D, 3D, VR i AR su također snažni atributi koji stavljaju Unity ispred ostalih alata, u svijetu 2D, VR i AR igara, Unity dominira tržište sa postocima korištenja [3]. Alat je poznat po kvalitetno implementiranom renderiranju i izgledu finalnog proizvoda te brojnim funkcionalnostima koje omogućuju programerima ostvarivanje njihovih ideja [3].

Visual Studio je integrirano razvojno okruženje (eng. Integrated developmet environment, IDE) koje ima brojne korisne funkcionalnosti za pisanje koda, vrlo je proširivo okruženje te podržava izradu aplikacija za najpopularnije platforme [4]. Iako nije nužno koristiti Visual Studio za izradu 2D platformera, odlučio sam uzeti ovaj alat za pisanje skripti u C# jeziku zbog toga jer sam već vrlo dobro upoznat s alatom i jer sadrži korisne funkcionalnosti poput IntelliSense-a, otklanjanja pogrešaka i jednostavnu navigaciju.

Budući da sam odlučio sam dizajnirati nivoe, likove i ostale elemente, za to ću koristiti Adobe Photoshop jer sam već sam od prije upoznat s programom i smatram da je jedan od najkvalitetnijih u svojem području. Photoshop je danas standard u digitalnom uređivanju slika, koristi se za rasterske slike, grafički dizajn i stvaranje digitalne umjetnosti [5]. Najpoznatiji je alat za uređivanje slika a jedna od glavnih karakteristika programa su slojevi.

2D platformeri

2D platformeri su žanr igara poznat po svojoj jednostavnosti, lakom učenju i jednostavnom pokretanju na gotovo svim uređajima. Iako su kvaliteta, detalji i kompleksnost igara napredovali zajedno s tehnologijom, 2D platformeri i dalje su u osnovi ostali isti.

Žanr 2D platformera je nastao u ranim osamdesetima, te je jedna od prvih igara bila Space Panic tvrtke Universal [6]. Kroz godine su izlazile vrlo popularne igre poput Donkey Kong-a, Super Mario-a te Metroid i Castlevania serijali igara [7]. Razvojem tehnologije 3D igre su počele dominirati tržištem no 2D platformeri nisu nestali već su se prilagodili novim trendovima. U zadnjih nekoliko godina pojavio se ponovni rast u popularnosti 2D platformera zahvaljujući brojnim indie naslovima koji su vratili žanr na scenu [6].
Kod izrade 2D platformera dolazi do podijele načina izrade koja se odnosi na stvaranje nivoa ili virtualnog svijeta igre s naglaskom na dizajn, naime riječ je o

podijeli na dizajn baziran na pločicama (eng. tile-based) i dizajn baziran na spriteovima (eng. sprite-based). Spriteovi po mojem mišljenju omogućuju veću slobodu u dizajniranju igre te pridodaju realizmu igre dok pločice daju retro osjećaj igri.

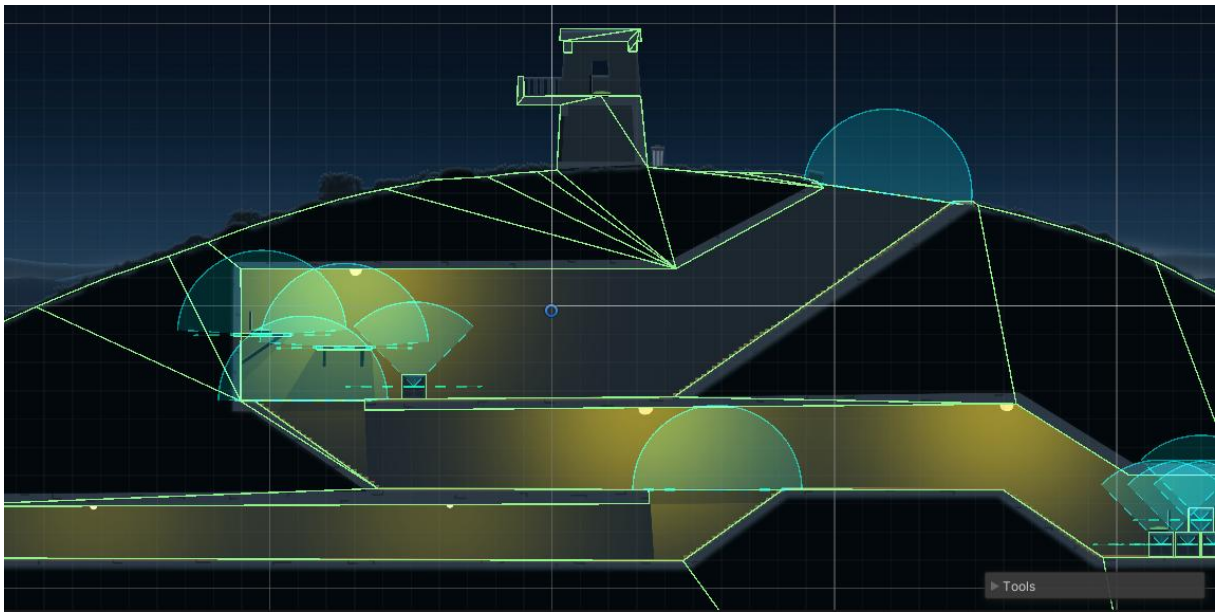
Izrada nivoa

Dizajnirao sam svaki grafički dio igre u Adobe Photoshop-u, pa tako i nivoe. Bitno je da je okolina podijeljena u slojeve kako bi mogao kasnije kontrolirati micanje daljnjih i bližih slojeva za efekt paralakse. Kada je proces dizajniranja gotov i slojevi željeno podijeljeni bitno je da se svaki sloj zasebno izveze kao PNG datoteka, za to sam koristio funkcionalnost Photoshop-a „Layers to files…". Kod uvoza datoteka u Unity potrebno je maknuti kompresiju kako bi dobili punu oštrinu slike u igri.

Iako okolina nakon toga izgleda gotova Unity i dalje ne zna gdje bi igrač trebao hodati pa bi kod dodavanja objekta sa fizikom isti objekt samo pao s ekrana zbog gravitacije, zbog toga se dodaje Collider2D, u ovom slučaju točnije PolygonCollider2D, to je komponenta koja se dodaje na objekt na kojem želimo da se sve odvija. Dodavanjem komponente nastat će zeleni peterokut kojeg lagano možemo urediti da s njime napravimo oblik po kojem će se igrač kretati. Uz glavni teren po kojem se igrač može kretati dodao sam zidne platforme poput polica, te drvene kutije i kantu za smeće. Na spomenute objekte sam uz Collider2D dodao i Platform Effector 2D komponentu koja omogućuje određivanje dijela i smjera koji će zapravo blokirati objekte koji dođu u susret. Na slici 1 kao primjer prikazana je drvena kutija i označeno je da će samo gornji dio blokirati igrača i to samo s gornje strane što znači da igrač može hodati ispred kutije i ako želi može skočiti na nju.

Kako okolina ne bi bila potpuno statična i da u 2D svijet dodam malo dubine što pomaže sa realizmom svijeta, na svaki prethodno spomenuti sloj dodao sam jednostavnu skriptu koja će micati slojeve na temelju određenog faktora i pomicanja kamere kako bi stvorio efekt paralakse.

Za upravljanje scenama putem skripte potrebno je uključiti biblioteku „Unity.SceneManagement". Prijelaz nivoa se aktivira tako da igrač dođe blizu desnog ruba trenutnog nivoa. Kako bi to postigao dodan je novi objekt u scenu koji sadrži *Collider2D* komponentu no ovaj puta je u komponenti označeno svojstvo „*Is Trigger*" što znači da neće blokirati objekte nego ga možemo koristiti kao okidač za željene radnje.



Slika br. 1: Prikaz Collider2D i Platfrom Effector 2D komponenti

Izrada likova

U igri se nalaze tri vrste likova, glavni agent kojim upravlja igrač, neprijatelji koji su lopovi i dio veće organizirane skupine koju vodi posljednji lik, njihov šef. Svi likovi su vizualno drugačije dizajnirani ali u istom stilu. Likove sam dizajnirao u Adobe Photoshop alatu, bitno mi je kod dizajniranja da odvojim dijelove koje želim individualno micati u posebne slojeve. Prije animiranja treba povezati dijelove likova. Unity za ovo ima po mojem mišljenju odlično i intuitivno rješenje tako da se na lika crtaju kosti i izradi kostur. Nakon dodavanja kosti generira se distribucija upravljanja, odnosno Unity pokuša odrediti koja kost upravlja sa kojim dijelom tijela, ako se dijelovi preklapaju tada je potrebno urediti upravljanje tako da svaka kost upravlja samo željenim dijelom tijela. Nakon kreiranja kostura, sve kosti i njihova hijerarhija je vidljiva u hijerarhiji scene, sljedeće je potrebno dodati *Animator* komponentu i otvoriti *Animator* i *Animation* prozore. U *Animation* prozoru se stvara novi animacijski isječak (*eng. Animation Clip*) te se pritiskom na tipku za snimanje mogu micati kosti lika kako bi se kreirala i snimila animacija. Kreirane animacije se automatski dodaju u *Animator* prozor gdje se mogu dodati parametri i logika kojom će se animacije izmjenjivati s jedne na drugu. Posljednji i najbitniji korak je dodavanje fizike na glavnog agenta te dvije skripte koje upravljaju kretanjem, pucanjem i interakcijom sa ostalim objektima. Slično glavnom agentu na neprijetelje su dodane slične funkcionalnosti uz pojednostavljeno okretanje i automatizirano pucanje. Svi likovi iznad sebe imaju *HealthBar,* vizualnu reprezentacija stanja lika koja prikazuje koliko štete je primio i koliko još može primiti prije nego umre.

Kamera

Za kontroliranje kamere sam odlučio koristiti Unity-ev paket Cinemachine koji se može lako i brzo dodati unutar Unity-a. Cinemachine paket dodaje novu karticu u sučelje alata preko koje se mogu dodati razne vrste kamera od kojih je za ovu igra potrebna 2D kamera. Dodana Cinemachine kamera postoji u sceni zajedno sa već postojećom kamerom te zapravo samo kontrolira ponašanje glavne kamere. Uz igranje sa postavkama dodane komponente moguće je lako postići željeno kretanje kamere u odnosu na agenta kojim upravlja igrač. Posljednje što je potrebno napraviti je kreirati novi objekt u sceni i na njega dodati Polygon Collider 2D komponentu koji će služiti kao granica kamere tako da je uvijek prikazan samo nivo i ništa izvan njega.

Korisničko sučelje

Iako je srž igre uspješno implementirana, kako bi igrač mogao upravljati igrom a ne samo agentom, potrebno je intuitivno, efikasno i jednostavno korisničko sučelje. Dijelovi poput glavnom i završnog menija su napravljeni u posebnim scenama koje se nalaze prije i poslije samih nivoa, dok su sučelja poput menija pauze, smrti i konstantnog vođenja rezultata implementirana unutar nivoa te se prikazuju uvijek ili se pozivaju putem koda. Za izradu korisničkog sučelja dodao sam u projekt besplatan Unity-ev paket, TextMeshPro, koji omogućuje detaljnije uređivanje teksta od ugrađenih opcija. Kod izrade skripte je važno u ovom slučaju uključiti dvije biblioteke, UnityEngine.UI i UnityEngine.SceneManagement. Praćenje rezultata je implementirano tako da se svaki broj nalazi u posebnom objektu koji sadrži skriptu sa funkcijom za ažuriranje prikaza koja se poziva kod interakcije s pripadnim objektom za skupljanje.

Pozadinska glazba i zvučni efekti

Posljednji no vrlo bitan element koji sam dodao u ovu igru je zvuk. Pozadinska glazba može odrediti atmosferu cijele igre dok neki zvučni efekti pomažu kod osjećaja realizma i povezanosti sa svijetom igre. Smatram da je bitno staviti zvučne efekte primarno kod radnji koje obavlja igrač kao u ovom slučaju skupljanje novca i satova kako ne bi bilo potrebno svaki puta pogledati brojač u gornjem desnog kutu već zvuk indicira da je radnja uspješno obavljena. Osim pozadinske glazbe u ovaj projekt sam dodao zvukove pucanja puške i pištolja, otvaranja kovčega i uzimanja novca, zvuk padanja tijela na tlo kod umiranja, zveckanja kod uzimanja sata s mrtvog neprijatelja, zvuk koraka kod trčanja agenta te obavijesti dobivanja poruke i završavanja cijele misije. Za zvuk u igrici sam kreirao dvije različite skripte, jedna je specifična samo za pozadinsku dok druga sadrži kod za upravljanje ostalim zvučnim efektima.

Zaključak

Ovim radom sam obradio osnove 2D platformera kao žanra igre, prisjetio se starih klasičnih igara koje su dominirale tržištem te upoznao sa novijim naslovima koji održavaju popularnost žanra. U praktičkom dijelu sam izradio vlastitu igru u Unity programskom alatu koristeći dodatne alate poput Adobe Photoshop-a za kreiranje svih slika korištenih u igri te Visual Studio za pisanje mnogih skripti potrebnih za uspješnu izradu igre.

Izradom ovog rada smatram da sam ostvario svoje ciljeve te iskoristio svoje umjetničke sposobnosti i znanje programiranja u C# jeziku kako bi uspješno izradio svoju prvu igru sa osnovnim elementima 2D platformera. U procesu izrade sam naučio koristiti programski alat Unity te smatram da sam stekao mnogo znanja o kreiranju igara što je nešto s čim se prije nisam susreo.

Reference

[1] J. Haas, „A History of the Unity Game Engine," Electronic Projects Collection, ožujak 2014. [Na internetu]. Dostupno: https://web.wpi.edu/Pubs/E-project/Available/E-project-030614-143124/unrestricted/Haas_IQP_Final.pdf [pristupano 30.06.2021.].

[2] MVC Staff, „United they stand", MCV Development News, prosinac 2009. [Na internetu]. Dostupno: https://www.mcvuk.com/development-news/united-they-stand/ [pristupano 30.06.2021.].

[3] Arnia software, „What Makes Unity So Popular in Game Development?", 10.08.2020. [Na internetu]. Dostupno: https://www.arnia.com/what-makes-unity-so-popular-in-game-development/ [pristupano 30.6.2021.]

[4] Microsoft, „Visual Studio Community", (bez dat.) [Na internetu]. Dostupno: https://visualstudio.microsoft.com/vs/community/ [pristupano 30.6.2021.]

[5] Techopedia, „Adobe Photoshop", (bez dat.) [Na internetu]. Dostupno: https://www.techopedia.com/definition/32364/adobe-photoshop [pristupano 30.6.2021.]

[6] Nodwin Gaming, „The evolution of platform games in 9 steps", 23.03.2017. [Na internetu]. Dostupno: https://www.redbull.com/in-en/evolution-of-platformers [pristupano: 01.07.2021.]

[7] H. Kantilaftis, „Genre Revival: 2D Platformers", studeni 2014. [Na internetu]. Dostupno: https://www.nyfa.edu/student-resources/genre-revival-2d-platformers/ [pristupano: 01.07.2021.]

Automatizirani robotski proces praćenja IR objekta unutar određenih granica

Ivan Kolarić i Mario Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
ikolaric@foi.hr, mkonecki@foi.hr

Sažetak

Cilj ovog djela je pomno predočiti izradu kompletnog robota koji ima funkciju praćenja infracrvenog objekta unutar određenih granica, gdje infracrveni objekt predstavlja lopta sa infracrvenim odašiljačima, a određene granice su zapravo nogometni teren sa zelenom podlogom obilježene bijelim linijama. Izrada samog robota prolazi kroz nekoliko faza redom: nabava elektroničkih komponenti, Dizajn i 3D printanje dijelova šasije, faza konstrukcije i na kraju faza programiranja. Prolaskom kroz sve faze, produkt je funkcionalan robot.

Ključne riječi: Robotika, 3D print, konstrukcija, programiranje, Arduino,

Uvod

Robot „nogometaš“ kojeg ću u ovom radu napraviti, u cijelosti spada u granu znanosti pod nazivom robotika. Robotika je jedna grana znanosti koja obuhvaća znanja iz mnogo različitih područja znanosti, a kao što su to znanja iz mehanike u svrhu gradnje robota u konstrukcijsku cjelinu. Također, primjenjuju se i znanja iz elektronike, kako bi se povezali i koristili elektronički sklopovi u cjelinu te kroz znanja iz automatike primijenili zajedno sa znanjima iz programiranja u jednu funkcionalnu cjelinu koja ima mogućnosti zamijeniti neki ljudski rad. Robotika se može podijeliti na dvije grane: mobilnu robotiku i industrijsku robotiku. Industrijska robotika koristi stacionarne robote koji su u principu robotske ruke koje teže ka preciznosti i brzini. Najčešća uporaba je u autoindustriji gdje se susrećemo sa robotima koji prenose dijelove, pričvršćuju i montiraju te iste dijelove itd., dok mobilna robotika, kao što i sama riječ govori, koristi robote koji imaju neku funkciju mobilnosti, koji se mogu micati i kretati od točke A do točke B. Robot kojeg ću ja izraditi spada u kategoriju mobilne robotike iz razloga što je pokretan, odnosno mobilan i kretat će se unutar određenih granica. Mobilni roboti se također mogu dalje razvrstati u dvije kategorije i to prema načinu kretanja i mjestu kretanja. No, kao što nam sama definicija robota govori, robot je u stvari programabilna tehnička tvorevina koja je u mogućnosti izvršavati seriju naredbi autonomno ili poluautonomno. S obzirom kako ćemo u ovom radu isprogramirati robota, prenijeti na njega program i pustiti ga da sam odrađuje neki niz naredbi, naš će robot biti autonoman. [1], [2]

Nabava elektroničkih komponenti

Što se tiče nabavke elektroničkih dijelova, dugogodišnjim volontiranjem u udruzi Hrvatsko društvo za robotiku, dobio sam uvid u nabavku elektroničkih komponenti. Velika većina elektroničkih komponenti se nabavlja putem jednog dobavljača, odnosno tvrtke Microline. Sama tvrtka Microline nam je dugi niz godina izlazila u susret, i čak modificirala neke dijelove u elektroničkom smislu, a kako bi nam omogućili što bolje elektroničke komponente koje odgovaraju Hrvatskom društvu za robotiku za daljnju primjenu u Robocup natjecanjima. Sama tvrtka je dugogodišnjim istraživanjem i radom u području robotike lansirala svoje verzije elektroničkih komponenata zvanom MRMS (Mobile robot modular system).

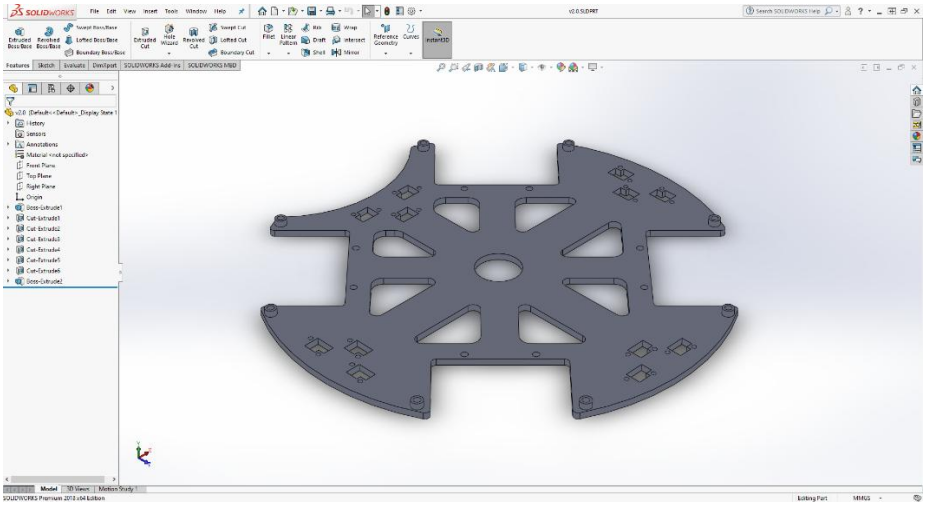
Jedna od najbitnijih komponenti mog robota je mikrokontroler. Teensy 3.2 je u stvari glava samog robota te upravlja svim komponentama koje se povezuju na njega. On dakle kontrolira ostalim elektroničkim komponentama odnosno ulazno/izlaznim jedinicama. Na nabavu Teensy 3.2 mikrokontrolera sam se odlučio iz razloga što tvrtka Microline ima unaprijed napravljenu razvodnu pločicu za taj mikrokontroler, a što će mi u daljnjem radu olakšati posao spajanja ulaza i izlaza s mikrokontrolera. Jedna od komponenti je i razvodna pločica, no kako ne bi došlo do zabune u radu, imam dvije razvodne pločice, jedna služi za napajanje, a jedna za razvod pinova mikrokontrolera. Obje pločice sam nabavio kod istog dobavljača. [3], [4], [5]
Sljedeći skup elektroničkih komponenata su senzori od kojih je najbitniji senzor za loptu „MRMS IR ball finder 2“ koji je unaprijed napravljen kako bi hvatao točne zrake lopte u točnim intervalima i u mikrokontroler slao podatke o kutu i udaljenosti lopte od senzora. Idući senzor je senzor boje „MRMS Reflectance Sensor A“ koji ima namjenu očitavanja refleksije podloge i ovisno o boji i refleksiji podloge šalje signal mikrokontroleru. Senzora boje sam naručio osam komada i postavio ih u parovima na prednju, stražnju, desnu i lijevu stranu robota. Pomoću njih se robot snalazi kako bi uvijek ostao u terenu, budući da može učitati bijelu liniju. Sljedeći senzor je ustvari kompas pomoću kojeg robot stalno održava isti smjer dobivajući podatke o trenutnoj usmjerenosti odnosno pomaku od sjevera. „MRMS IMU“ kompas je baziran na Bosch-ovom čipu BNO055 koji je u današnjem svijetu jedan od najboljih izbora što se tiče kompasa. Na robotu se koristi i jedno tipkalo kako bih se pokrenuo program pomoću njega. [6], [7], [8], [9]

Od izlaznih komponenata koristio sam 4 četkana istosmjerna motora sa dva polariteta. Rade na naponu od 3V do 12V i imaju maksimalnu brzinu od 730 okretaja po minuti. Motore sam nabavio sa Servocity webshopa. Kako bi upravljali motorima potrebno je pribaviti i kontroler motora iz tog sam razloga nabavio „MRMS Motor Driver 4x3.6A B“ kontrolere. [10], [11]

Napajanje je jedan od najbitnijih dijelova iz razloga što se sva energija dobivena iz baterije u njemu pretvara step-down ili step-up metodom u nama potreban napon, a u našem konkretnom slučaju, potreban nam je napon od 5 V i 3.3 V što nam izabrano „MRMS Power Supply 3x (mrm-ps3x)“ napajanje omogućava. Što se tiče ostalih materijala i komponenata koji su mi bili potrebni pri izradi, spajanju i puštanju robota u pogon su baterije, punjači za baterije, vodiči, vijci, matice, odstojnici, Omni-Wheel kotači i isprintani dijelovi šasije. [12]

Dizajn i 3D printanje šasije robota

Kao jedan od alata sa kojim sam se već ranije susreo je SolidWorks, a koristio sam ga u svrhe dizajniranja i 3D printanja manjih dijelova, kao npr. manje robotske ruke koje su se mogle pričvrstiti na manje robote. No, nikad nisam 3D modelirao kompletnu šasiju, što mi je bio izazov. SolidWorks je jedan od najrasprostranjenijih i najkorištenijih CAD (Computer aided design) programa u industrijskom svijetu. Čak bi se moglo reći i kako je SolidWorks ustvari standard u inženjerstvu. Izrađen je od strane francuske tvrtke Dassault Systemes i fokusira se primarno na inženjerskom dizajnu pomoću geometrijskog parametarskog modeliranja. Prije no što sam krenuo dizajnirati u programu SolidWorks, potrebno je bilo razmisliti na koji način ću dizajnirati samu šasiju. Tada sam shvatio kako ću šasiju morati podijeliti na dva dijela i kako ću graditi i dizajnirati od donjeg dijela prema gornjem što sam i učinio višestrukim korištenjem naredba Extrude i Extruded cut uz pomno mjerenje i odmjeravanje, kako bi svaka komponenta mogla sjesti na svoje mjesto.



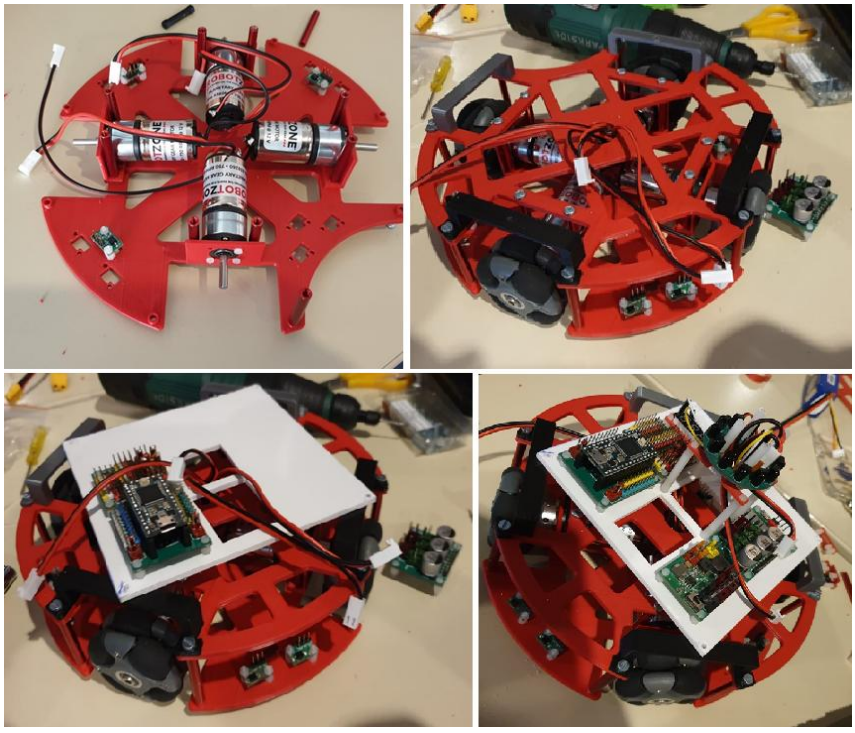
Slika br. 1: Dizajn donjeg dijela šasije u programu SolidWorks

Kada sam izradio gornje i donje dijelove šasije krenuo sam u izradu držača za motore koji će povezivati gornji i donji dio šasije. Pažljivo sam uračunao visinu odstojnika, šasije i motora, kako bi se sve moglo savršeno poklopiti gornjim dijelom šasije, a opet da kotači budu na podu, uz idealnu visinu donjeg dijela

šasije od podloge do nje od otprilike 5mm – 10mm. Nakon dizajna šasija slijedi i 3D printanje, a za moje svrhe, otišao sam u Varaždinsku udrugu „Robofreak“, gdje mi je predsjednik udruge Ivica Kolarić dozvolio da iskoristim printer „Zortrax M300“ u svrhu printanja već nacrtanih 3D modela. Na korištenje sam tako dobio program Z-suite pomoću kojeg sam namjestio postavke printanja svakog pojedinog elementa. Svaki od zadanih elemenata sam postavio na iste postavke, kako bi print bio jednak i čvrst na sva 3 dijela šasije. Donjem dijelu šasije je bilo potrebno 8 sati i 17 minuta kako bi se isprintao i bilo je potrebno 92 grama PLA materijala. Dok je gornjem dijelu šasije bilo potrebno 6 sati i 28 minuta za printanje i 73 grama PLA materijala. Jednom držaču za motore je bilo potrebno 32 minute kako bi se isprintao i 5 grama PLA materijala.

Konstrukcija robota

Samu konstrukciju robota sam krenuo raditi od donjeg dijela prema gornjem dijelu, što znači da sam prvo pričvrstio motore zajedno sa držačima i senzore linije na donji dio šasije. Jednom kada sam pričvrstio sve potrebne senzore i motore na donji dio šasije, tada sam prvo provukao vodiče motora kroz gornji dio šasije, a tek onda sam krenuo na pričvršćivanje gornjeg dijela šasije zajedno sa donjim dijelom šasije.



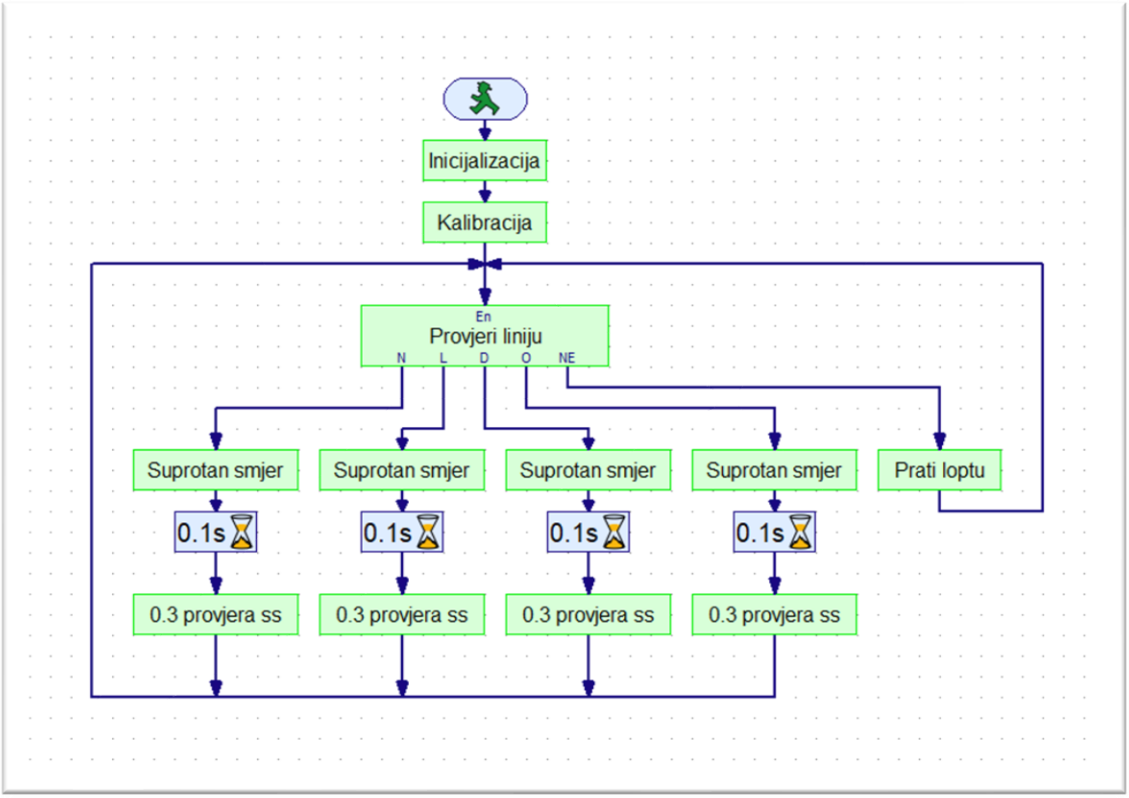
Slika br. 2: Izrada robota

Pažljivo sam pričvrstio odvijačem sve vanjske dijelove šasije kako bi kompletna šasija bila što kompaktnija i nakon toga sam krenuo na pričvršćivanje motora sa gornjim dijelom šasije. Potom sam pažljivo na plastičnu mekanu pločicu u obliku A4 papira pričvrstio sve preostale elektroničke elemente kako bih

izbjegao kratki spoj. Senzor lopte sam montirao pomoću odstojnika na višu poziciju, a kako bih izbjegao blokiranje signala smetnjom ostalih komponenata i vodiča.

Programiranje robota

Programski jezik koji ću ja koristiti za programiranje izrađenog robota u svome radu je Arduino, koji je u principu C++ sa nekim dodatcima i dodatnim funkcijama. Sam programski jezik se koristi u Arduino IDE razvojnoj okolini.



Slika br. 3: Dijagram toka programa

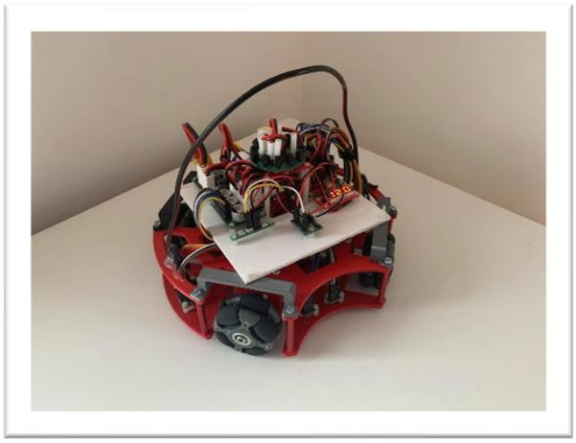
Samo programiranje robota u svrhu ispunjavanja zadatka je potrebno postepeno odrađivati. Isto kao što sam i do sada postepeno odradio testiranje robota, tako sam odradio i programiranje samog robota. Krenuo sam top-down metodom gdje mi je za početak najvažnije bilo kako napraviti da robot prati infracrveni objekt odnosno loptu. Shvatio sam da ću to postići kombinacijom dvije funkcije, a to su funkcija motors.goOmni() i irFinders.irSource().angle. Funkciju irFinders.irSource().angle sam zapisivao u varijablu smjer konstantno, no ispitivanjima sam shvatio kako mi vrijednost same funkcije izbacuje sirove vrijednosti između 0 i 950. Slijedom uvrštavanja same varijable, smjer samo uspio iskorigirati na vrijednost smjera lopte tako da uvijek bude između -180 i 180. Takvom korekcijom sam uspio vrijednost senzora lopte prilagoditi funkciji motors.goOmni() kako bi kasnije u 2. argument mogao uvrstiti varijablu smjer,

budući da 2. argument čita vrijednosti između -180 i 180 i kreće se prema tom smjeru. Kada sam uvrstio smjer u funkciju motors.goOmni robot je pratio loptu, no shvatio sam kako se robot nakon otprilike minute i sudaranja sa drugim objektima, gotovo uvijek rotira na kompletno drugu stranu. Iz tog razloga, morao sam u program uvrstiti kompas. Budući kako MRMS sistem već ima ugrađene funkcije za poravnanje po kompasu, samo sam uvrstio funkciju rotationToMaintainHeading(headingToMaintain) u treći argument, jedino što je bilo potrebno je bilo zapamtiti u varijablu headingToMaintain vrijednost 0 prilikom pokretanja robota. Na taj način je robot znao koji mu je početni smjer kojeg se treba pridržavati. Kasnije sam se igrao sa tipkalima i uvrstio mogućnost napadanja na suprotni gol gdje bih na početku programa zapravo postavio vrijednost headingToMaintain varijable na 180. Kada sam uspio korigirati kompas i smjer dolaska lopte sa usmjerenjem kretanja motora, tada je robot već pratio loptu i sve što je dalje trebalo, je u već postojeći program pomoću nekoliko grananja uvrstiti senzore za očitavanje boje. Kako bi senzori boje funkcionirali normalno, na početku je trebalo zapamtiti vrijednosti svakog pojedinog senzora, što sam i učinio upisavši njihove vrijednosti u varijable kojima sam zbrojio konstantu SENZORI, koja se kasnije prema potrebi može mijenjati, no u vrijeme pisanja ovog rada iznosila je 200. Konstantu SENZORI sam zbrojio iz razloga kako se bijela boja očitava na vrlo visokim vrijednostima od preko 1000, dok je zelena podloga površine terena na otprilike 400. Kako robot uvijek započinje na zelenoj površini, ideja mi je bila kako bi na početku robot zapamtio vrijednosti površine + toleranciju koju sam morao uvrstiti zbog robotove brze kretnje. Te početne vrijednosti onda kasnije provjeravam sa trenutnim vrijednostima senzora u svakom trenutku. Na taj način provjeravam postoji li bijela linija. Senzori su raspoređeni taktički na 4 strane robota, kako bi znao sa koje strane je učitana linija i u koju stranu je potrebno odmaknuti robota od linije. Zbog urednosti programa, provjeru postoji li linija, izvršavam u funkciji senzoriLinije(), gdje postavljam varijable svakog senzora posebno na true odnosno false. Kasnije u kodu grananjem vršim provjeru tih istih varijabli i ukoliko je linija očitana, robot staje na 100 milisekundi i nakon toga se kreće unatrag za konstantu LINIJAMS iz razloga jer u većini slučajeva robot ne stigne zakočiti i prijeđe liniju. Nakon toga, robot kreće u stranu još za jednu konstantu VRATI, no kada se vraća, svaku milisekundu provjerava je li učitao bilo koji od preostalih 6 senzora, a ukoliko ih je očitao, robot staje nakon 300 milisekundi i izlazi iz petlje.

Zaključak

Povezavši sva prethodna poglavlja u jednu cjelinu, shvatio sam kako je ono najbitnije što sam naučio kontrola nad samim robotom. Kontrola nad robotom se provodi pomoću programiranja, gdje čovjek nad robotom ne mora imati direktnu kontrolu već unaprijed određenim instrukcijama upravlja njime.

Smatram da u automatiziranom načinu programiranja leži budućnost i da će se na tržištu u budućnosti pojavljivati sve više i više automatiziranih robota.



Slika br. 4: Robot „Nogometaš“

Reference

[1] Z. Kovačić, S. Bogdan, V. Krajči, Osnove robotike, Graphis Zagreb, 2002.
[2] https://vub.hr/images/uploads/3314/podjela_robotike__industrijska_i_mobilna_robotika.pdf
[3] https://www.pjrc.com/store/teensy32.html
[4] https://www.microline.hr/HTML.ashx?_command=showItem&itemKey=mrm-tee
[5] https://www.microline.hr/HTML.ashx?_command=showItem&itemKey=mrm-distrib-a
[6] https://www.microline.hr/HTML.ashx?_command=showItem&itemKey=mrm-ir-finder2
[7] https://www.microline.hr/HTML.ashx?_command=showItem&itemKey=mrm-ref-a
[8] https://www.microline.hr/HTML.ashx?_command=showItem&itemKey=mrm-imu-an3
[9] https://www.microline.hr/HTML.ashx?_command=showItem&itemKey=mrm-switch
[10] https://www.servocity.com/730-rpm-premium-planetary-gear-motor/
[11] https://www.microline.hr/HTML.ashx?_command=showItem&itemKey=mrm-mot4x3.6b
[12] https://www.microline.hr/HTML.ashx?_command=showItem&itemKey=mrm-ps3x

Razvoj proceduralno-generirane računalne igre simulacijskog tipa u razvojnom okruženju Unity

Alan Vranić i Mario Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
avranic@foi.hr, mkonecki@foi.unizg.hr

Sažetak

Ovaj se rad bavi procesima istraživanja, definiranja koncepata, razvoja te dokumentiranja izrade proceduralno generirane igre u programskom alatu Unity. Cilj je napraviti igru u kojoj ni jedna sesija igranja nije ista, te potaknuti igrača da eksperimentira s različitim tehnikama i pristupima igranja. Ovo se ostvaruje na više načina, no dva ključna koncepta su proceduralno generiranje svijeta koji se tijekom igranja može mijenjati ovisno o odlukama igrača, te nepostojeća (ili izuzetno ograničena) kontrola nad likovima u igri. Igrač nema izravnu kontrolu nad likovima, već im zadaje naredbe i smjernice kojih se oni mogu, ali i ne moraju držati. Nadalje, cilj je pokazati da proceduralna generacija i korištenje slučajno generiranih brojeva može biti dobra alternativa pristupu dizajniranja razina ručnim radom, koji se još uvijek pretežito koristi u većini igara.

Ključne riječi: unity; igra; preživljavanje, građenje, proceduralna-generacija;

Uvod

Jedna od najvećih kvaliteta igre u koje je uložen trud možemo iščitati iz njene sposobnosti da zaokupi igrača, omogući mu da se upusti u svijet igre te u njemu provede značajnu količinu vremena. Ovu kvalitetu ponovljivosti (eng. *replayability*) možemo vidjeti u puno igara koje pripadaju simulacijskom žanru (npr. „The Sims“ ili „Rimworld“) ili žanru MOBA (eng. *Massive Online Battle Arena*), poput igre „League of Legends“. Cilj ovog završnog rada jest napraviti snažan temelj za daljnji razvoj igre simulacijskog tipa te opisati procese razmišljanja i donošenja odluka tijekom njenog razvoja.

Značajnost ovakve igre je velika, jer sa konceptima poput proceduralne generacije svijeta moguće je dizajnirati iskustvo koje je svaki put jedinstveno i igraču pruža drugačije iskustvo. U principu, dizajnirati igru simulacijskog tipa (eng. *colony simulator*) značajno je teže zbog same nepredvidljivosti i raznolikosti okoline u svakoj sesiji igranja. Ipak, ukoliko je ovaj ključni proces cjelovito i skladno napravljen, igrač nikad neće znati što očekivati; iz igre može izvući sate, dane, pa čak i mjesece igranja.

Moja motivacija za izradu ovog tipa igre te odabirom žanra simulacije jest ogromna strast prema igrama istog tipa, kao i želja za stvaranjem nezaboravnih i jedinstvenih iskustava. Ako možemo napraviti svijet koji je toliko zadivljujući da potakne igrača da u njemu provede puno vremena, pa čak i zaboravi na stvarni svijet, onda možemo reći da smo napravili izvrsnu igru.

Prilikom izrade bilo koje proceduralno generirane igre, osobito simulacijskog tipa, bitno je znati koje su prednosti i nedostaci ovog pristupa razvoja.

Prednosti:

- **Ponovljivost** – svaka je nova igra drugačija za svakog igrača zbog svijeta koji uvijek može biti drugačiji, dok slučajni događaji i ostali elementi određeni slučajno generiranim brojevima ostvaruju nepredvidljivost, što čini igru dinamičnijom i zabavnijom.
- **Efikasnost** – nije potrebno dizajnirati razine (eng. *levels*) u igri; razvojni tim definira pravila po kojima se svijet i događaji generiraju, nakon čega igra sve radi sama.
- **Sloboda** – igrač je slobodan da eksperimentira sa svijetom, gradi, uništava te se povezuje se s likovima u igri.

Nedostaci:

- **Iznimna složenost razvoja** – razvojni tim mora uzeti u obzir da se u proceduralno generiranom svijetu može dogoditi bilo što, te mora definirati skup strogih pravila kako bi se osiguralo da se događa samo ono što je zamišljeno.
- **Prilagodljivost** – u svijetu u kojem igrač može graditi ili rušiti teren, potrebno je osigurati da likovi u igri reagiraju na sve promjene da ne dođe do nepoželjnih ponašanja.
- **Potrošnja vremena** – kada se u igru dodaje nova značajka, potrebno je uzeti u obzir sve moguće interakcije koja bi ta značajka mogla imati s ostalim dijelovima svijeta (koji je također proceduralno generiran), što zahtijeva puno isprobavanja, eksperimentiranja te ispravljanja grešaka.

Proceduralna generacija

Razvoj igre krenuo je od najbitnijeg dijela – proceduralne generacije svijeta u kojem će se sve ostalo zbivati. Koncept proceduralne generacije bio je korišten već od 1978. godine u igrama poput „Beneath Apple Manor“ [1] i „Rogue“ [2]. Iako postoji puno načina za implementaciju proceduralne generacije, među najpopularnijima spada tehnika generiranja slučajnih brojeva.

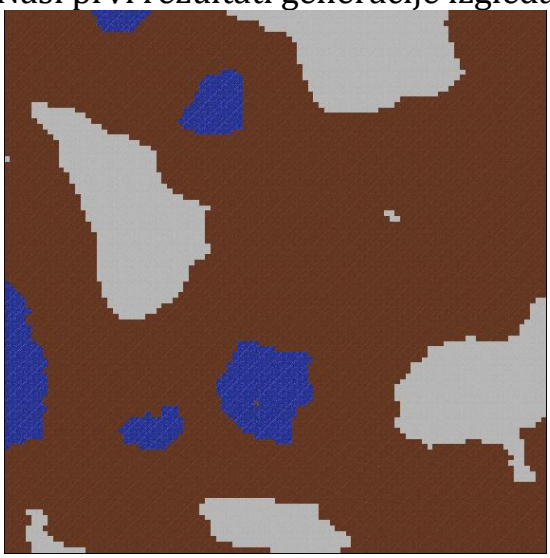
„Slučajni“ brojevi zapravo uopće nisu slučajni – oni su pseudoslučajni (eng. *pseudorandom*), što znači da se u teoriji može predvidjeti koji će se slučajni broj generirati [3]. Prije generacije, potrebno je odrediti sjeme (eng. *seed*), što je

zapravo broj koji se koristi kako bi se slučajni broj generirao i kako bi se njemu pružio pojam slučajnosti. Nakon što imamo *seed*, možemo izračunati slučajan broj. Ovaj slučajan broj može se iskoristiti za velik broj različitih primjena. Primjerice, za pojedinog lika u igri možemo odrediti koliko će štete činiti njegov mač neprijatelju prilikom napada, u rasponu od pet do deset. Ipak, nas zanima naprednji način korištenja slučajnih brojeva putem proceduralnog šuma. Najpopularniji proceduralni šum koji je korišten u svrhu izradu video igara zove se Perlinov šum (eng. *Perlin noise*). Osmislio ga je 1983. godine Ken Perlin [4]. Kod koji je korišten za generaciju šuma originalno potječe iz serije videa koje je napravio Sebastian Lague na YouTube-u, te je izmijenjen za svrhe ove igre [5].

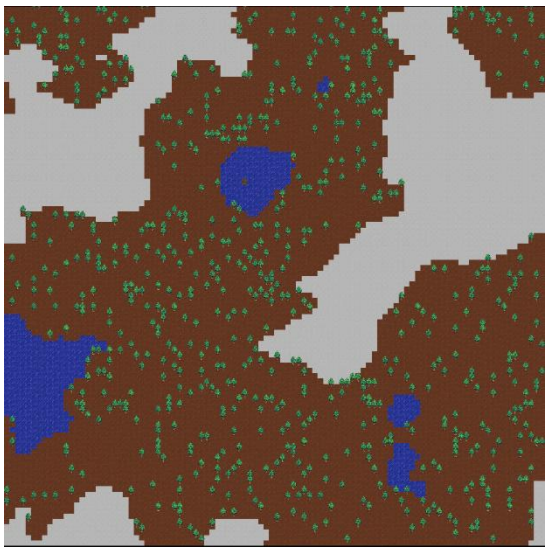
Generacija svijeta

Naš je cilj za svaku poziciju u našem svijetu odrediti koji bi izgled ta pozicija trebala poprimiti. S obzirom da proceduralno generirani svjetovi mogu biti promjenjive veličine, za potrebe ovog rada bavit ćemo se sa svijetom veličine 100x100 (sto pozicija na x osi, sto pozicija na y osi). Kako bi grafički vizualizirali svijet, potrebno je na svakoj poziciji postaviti sliku. Ove slike (eng. *sprites*) se određuju na temelju vrijednosti broja Perlinovoga šuma te pozicije. Kao rezultat izračuna dobije se dvodimenzionalno polje vrijednosti „*float*“, gdje je svaki broj u rasponu od nula do jedan. Ovisno o dobivenom broju možemo definirati da za vrijednosti od 0.0 do 0.2 postavljamo sliku vode, za vrijednosti od 0.2 do 0.7 postavljamo sliku zemlje, dok za vrijednosti od 0.7 do 1 postavljamo sliku kamenja. Osim ovoga, na sličnom principu možemo odrediti hoće li se na pojedinoj poziciji generirati drvo.

Naši prvi rezultati generacije izgledati će ovako:



Slika 1: Rezultat prve generacije svijeta



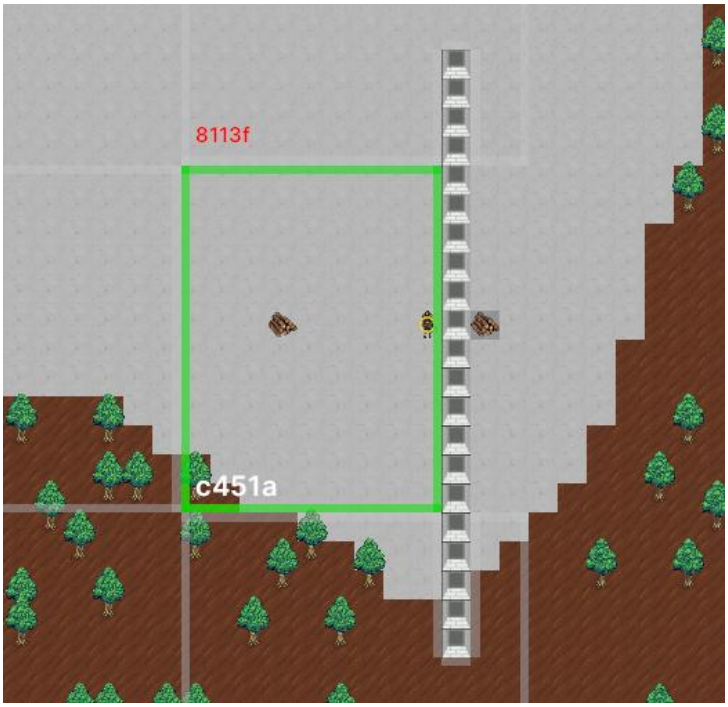
Slika 2: Generirana drveća

Regije

Kako bi traženje stvari diljem svijeta bilo što brže i efikasnije, igra koristi intrigantan sustav regija. Korištenjem takozvanih poveznica i korištenjem procesa hashiranja istih, cijeli svijet podijeljen je u regije koje se dinamično mogu mijenjati, te uvijek znaju koje su njihove susjedne regije.

Ovo rješava dva znatna problema:

1. Prolaženje kroz sve regije u svijetu kada tražimo neku stvar je izuzetno brzo jer ne moramo raditi dodatne izračune. Cijela je struktura podataka trenutačno izračunata kada se regija promijeni samo za tu regiju i njene susjede, dok sve ostale regije ostaju iste, što rezultira iznimno brzim operacijama za računalo.
2. Problem udaljenosti. Zamislimo da lik u igri stoji pored veoma dugog zida te traži najbliže drvo za izgradnju kuće. S njegove strane zida nalazi se drvo, no udaljeno je od njega za pet pozicija. S druge strane zida se također nalazi drvo te je udaljeno za samo dvije pozicije. Kada bi spiralno tražili najbliže drvo, zaključili bi da je drvo s druge strane zida, prema apsolutnoj udaljenosti, bliže poziciji lika, te bi onda taj lik morao hodati okolo cijelog zida po to drvo, što nije željeno ponašanje. S regijama, prvo tražimo stvar koju trebamo u našoj regiji, pa onda u susjedima te regije, pa onda u njihovim susjedima i tako sve dok ne nađemo tu stvar. Ovim pristupom, ipak bi zaključili da nam je bliže drvo koje je udaljeno pet pozicija, jer se ne nalazi na suprotnoj strani zida koja ne spada u našu regiju, što možemo vidjeti na sljedećoj slici.



Slika 3: Problem udaljenosti

Pathfinding

Kako bi se likovi u igri mogli kretati, potrebno im je unaprijed odrediti put kretanja, što se zove pronalaženje puta (eng. *pathfinding*). Iako postoji puno vrsta pronalaženja puta, nas općenito zanima koji je najkraći put od točke A do točke B. Postoji puno algoritama za pronalaženje puta, no jedan od njih zove se Dijkstrin algoritam (eng. *Dijkstra's algorithm*). Ovaj algoritam uvijek pronađe točan i najkraći put između dviju točaka, no njegov rezultat izgleda poprilično neprirodno za kretanje humanoidnih likova, te je poprilično spor za izračunati na većim veličinama svijeta, a pogotovo za veći broj likova u svijetu.

Kao ekstenzija Dijkstrinog algoritma, u igru ulazi A* (čitaj: „*A-star*“) algoritam. A* također se koristi za pronalaženje puta od točke A do točke B, no mnogo je brži jer koristi heuristike. U kontekstu pronalaženja puta, heuristike su matematičke optimizacije koje se koriste za rješavanje poznatih problema na više efikasan način [6]. Ovisno o brzini heuristike, put koji dobijemo možda nije u potpunosti točan, no za naše potrebe ovo je bolja opcija zbog znatno veće brzine izračuna. Osnovna implementacija algoritma poprilično je jednostavna, no Unity Asset Store, trgovina gdje se mogu preuzeti i koristiti gotova softverska rješenja, sadrži „A* Pathfinding Project“ (kraće A*PP) kojeg ćemo koristiti u ovom projektu.

Poslovi

Vrijeme je da dizajniramo par konkretnih poslova. Služit ćemo se tehnikom umjetne inteligencije zvanom stabla ponašanja. S Unity Asset Store-a također je preuzet paket „Behavior Designer“ (kraće BD) koji olakšava dizajniranje ovih poslova. BD dolazi spreman s nekoliko osnovnih zadataka (eng. *tasks*) koje ćemo koristiti. Kategorije ovih zadataka su sljedeće:

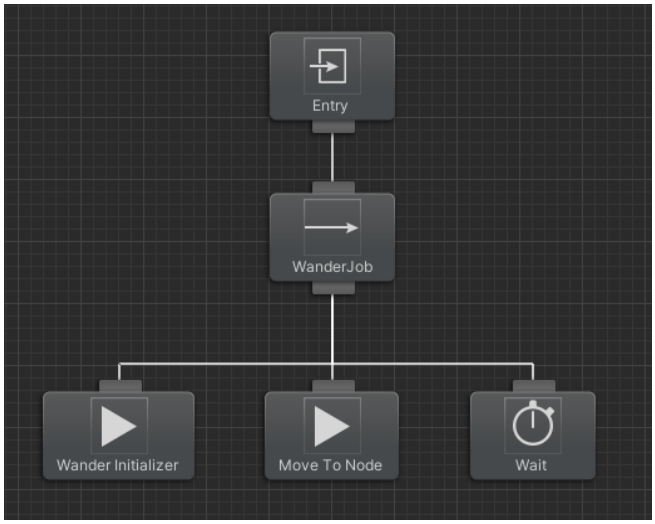
- Kompozitni (eng. *composites*)
- Dekoratori (eng. *decorators*)
- Radni (eng. *actions*)
- Kondicionalni (eng. *conditionals*)

Svaki zadatak u BD u bilo kojem dijelu svojeg izvršavanja može vratiti rezultat „Uspjeh“, „U tijeku“ ili „Neuspjeh“, što određuje slijed daljnjeg izvršavanja stabla. Za naše svrhe, koristit ćemo kompozitne, radne te kondicionalne zadatke. Prije je bilo spomenuto da se zadaci mogu ugnježđivati, stoga možemo prvo napraviti zadatak „Move To Node“ koji će narediti liku da hoda do određene poziciju u svijetu. Prednost rada sa stablima ponašanja jest da samo jednom moramo napraviti ovakav zadatak, te ga možemo ponovno iskoristiti u svim ostalim zadacima. Svaki zadatak unutar sebe krije C# kod gdje možemo točno definirati što se događa. Zanimaju nas dvije metode, „*OnStart*“ i „*OnUpdate*“. „*OnStart*“ će samo jednom na početku izvršenja zadatka izvršiti kod, što je korisno za postavljanje inicijalnih vrijednosti varijabla i slično.

„*OnUpdate*“ će svaki određeni vremenski interval (eng. *tick*) evaluirati zadatak, te ovisno o kondicionalnom uvjetu vratiti „Uspjeh“, „U tijeku“ ili „Neuspjeh“.

Lutanje

Posao lutanja (eng. *wander*) započet ćemo kompozitnim zadatkom *slijeda* te će on imati troje djece. U svakom od sljedećih ovakvih stabala na početku vidjet ćemo zadatak koji u svom nazivu sadrži „Initializer“. To je pomoćni zadatak koji prenosi informacije iz instance klase tog posla (u ovom slučaju wander) u stablo odlučivanja, no konkretno ne odrađuje nikakve radnje. Izuzetno u ovom slučaju, „Initializer“ bira slučajnu poziciju u svijetu prema kojoj će se lik kretati. Sljedeće, možemo iskoristiti zadatak koji smo prije napravili, „Move To Node“. Naposljetku, dodajemo već ugrađeni zadatak u Behavior Designer-u „Wait“, koji će narediti liku da stoji na mjestu i ništa ne radi u slučajno odabranom intervalu od jedne do pet sekundi. Naše trenutno stablo izgleda ovako:



Slika 4: Stablo zadatka "Wander"

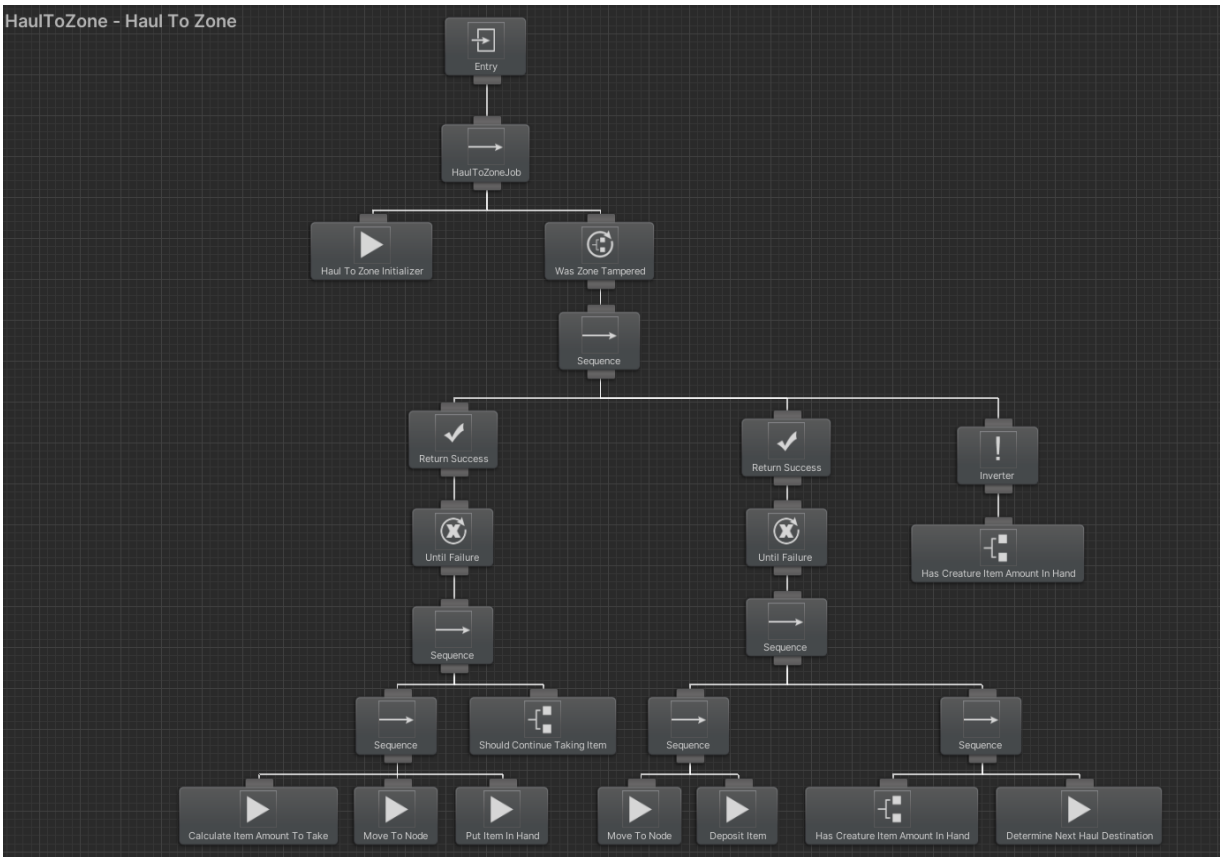
Dakle, kada lik počne izvršavati ovaj posao, bit će odabrana slučajna pozicija u svijetu, lik će se kretati do te pozicije i čekat će slučajni vremenski interval. Rezultat ovog posla prirodna je radnja lutanja koja će se ujedno koristiti kada lik ne može pronaći ni jedan drugi posao, pa će besposleno lutati.

Nosi stvar u zonu

Ovaj zadatak ujedno je i najsloženiji. Kada likovi pocijepaju drveće, ona će svoj predmet drva stvoriti na podu pored toga mjesta. S obzirom da likovi mogu bilo gdje cijepati drveće, uskoro bi se suočili sa situacijom gdje su predmeti drva od pocijepanih drveća posvuda. Ovo postaje problematično kada želimo napraviti naselje te imati sve dostupne materijale pri ruci. Kako bi riješili ovaj problem, moguće je odrediti zone u koje bi likovi trebali nositi stvari s poda. Zona je struktura podataka koja označuje na koje pozicije u svijetu treba dostavljati stvari. Cilj posla „nosi stvari u zonu“ (eng. *Haul To Zone*) je odrediti koje je stvari

potrebno odnijeti u za to predviđene zone te izvršiti tu radnju. Ovaj posao mogao je biti poprilično jednostavan, no da u potpunosti iskoristimo stabla odlučivanja i istražimo njegove mogućnosti, posao izgleda ovako:

1. Odredi postoji li slobodna zona te odredi njen slobodni kapacitet
2. Odredi inicijalan materijal koji je potrebno nositi u zonu
3. Ukoliko u blizini postoji sličan materijal, usputno uzmi i njega, te ponavljaj do kad postoji slobodan kapacitet nošenja u rukama i slobodan kapacitet u zoni
4. Odnesi materijal u zonu gdje se nalazi već postojeća hrpa tih materijala, te ponavljaj dok ne ponestane takvih hrpa
5. Ukoliko više ne postoje slobodne hrpe postojećih materijala, pronađi slobodno mjesto u toj istoj zoni, te ostatak materijala ostavi ovdje. Ponavljaj toliko dugo dok svi materijali nisu dostavljeni u postojeće zone sa slobodnim kapacitetom.



Slika 5: Stablo zadatka "HaulToZone"

Rezultat poslova

Nakon što su svi poslovi napravljeni, stvorimo par likovi u igri. Prvo što će se likovi napraviti jest evaluirati postoji li koji dostupan posao. Na početku igre ne postoji, stoga će stablo odlučivanja svakom liku zasebno narediti da luta. Bilo koda možemo početi postavljati zone, označivati drveće za cijepanje, te postavljati nacrte koji trebaju biti izgrađeni. Likovi će sami zaključiti postoji li

dostupan posao, rezervirati potrebne materijale te krenuti raditi, bez upletanja u posao ostalih likova.



Slika 6: Nošenje materijala, cijepanje drva i konstrukcija zidova

Zaključak

Cilj ovog projekta bio je napraviti proceduralno generiranu igru u kojoj postoje likovi koji samostalno odrađuju poslove, te stvaraju dojam realističnog i živućeg svijeta. Navedeno je uspješno postignuto korištenjem Perlinovog šuma za generaciju svijeta, dok se tehnika umjetne inteligencije stabla odlučivanja koristila kako bi se željeno ponašanje definiralo u obliku zadataka i poslova te bilo izvršeno od strane likova u igri.

Osim velike količine istraživanja i konstantnog učenja, upotrijebljeni su alati poput JetBrains Rider, Visual Studio Code za programiranje, Adobe Photoshop za dizajniranje jednostavnih slika i ostalih grafika koje su bile potrebne za vizualizaciju svijeta, kao i razni paketi preuzeti s Unity Asset Store-a kako bi se olakšao proces razvoja.

Sam projekt dokazao je kako je moguće koristiti slučajno generirane brojeve za izradu proceduralno generiranog svijeta koji svakom igraču, svaki puta, omogućuje drugačije i zanimljivo iskustvo, a sve to bez ručnog dizajniranja

razina. Isto tako, vidljivo je da je moguće kreirati svjetove u kojima žive ljudi koji su izvan izravne kontrole igrača, te kako se putem inteligentnog oblikovanja stabla odlučivanja i davanja neizravnih naredbi može dočarati osjećaj da je svaki od tih likova jedinstven pojedinac koji izvršava poslove u zajednici.

Reference

[1] procedural-generation, „Procedural Generation“, *Procedural Generation*. https://procedural-generation.tumblr.com/post/113776409275/beneath-apple-manor-1983-if-ive-learned-one (pristupljeno kol. 26, 2021).

[2] „Going Rogue-like: When to use Procedurally Generated Environments in Games“. https://www.gamasutra.com/blogs/CalebCompton/20180410/316326/Going_Roguelike_When_to_use_Procedurally_Generated_Environments_in_Games.php (pristupljeno kol. 26, 2021).

[3] „Pseudo-Random Number - an overview | ScienceDirect Topics“. https://www.sciencedirect.com/topics/mathematics/pseudo-random-number (pristupljeno kol. 24, 2021).

[4] „GPU Gems - Chapter 5. Implementing Improved Perlin Noise“. https://developer.download.nvidia.com/books/HTML/gpugems/gpugems_ch05.html (pristupljeno kol. 26, 2021).

[5] Sebastian Lague, *Procedural Landmass Generation (E01: Introduction)*. Pristupljeno: kol. 24, 2021. [Na internetu Video]. Dostupno na: https://www.youtube.com/watch?v=wbpMiKiSKm8&list=PLFt_AvWsXl0eBW2EiBtl_sxmDtSgZBxB3

[6] „Heuristic Algorithm - an overview | ScienceDirect Topics“. https://www.sciencedirect.com/topics/mathematics/heuristic-algorithm (pristupljeno kol. 26, 2021).

Uloga računalnih igara u obrazovanju

Ivan Blažek
Fakultet organizacije i informatike, Sveučilište u Zagrebu
Ivan.blazek@foi.hr

Sažetak

U ovome radu objasnit će se pojam video igara, te njihova korist u današnjem svijetu, točnije njihova uloga u obrazovanju. Video igre postaju jedan od glavnih medija kroz koje novije generacije imaju interakciju sa svijetom, sve su proširenije i postaju dio slobodnog vremena među sve većom populacijom. Objasnit će se sama razlika među različitim video igrama, njihovim žanrovima, te kasnije kakav imaju utjecaj na osobu, nakraju prikazati kako bi bilo moguće implementirati svijet video igara unutar edukacije. Prikazat će se na koji način video igre mogu unaprijediti nastavu, motivirati studente te na koji način koristiti efektivno kao pomagalo u nastavi, alat za učenje i unaprjeđivanje vještina i ljudskih sposobnosti, poučavanja i samostalnog ponavljanja edukacijskog gradiva.

Ključne riječi: video igre, obrazovanje, edukacija, školstvo, igrice, utjecaj, visoko školsko

Uvod

Od samog nastanka računala pa tako i video igara pojavljuje se pitanje koliko je štetno za okoliš, a najviše koliko su video igre štetne za ljude pa tako je naglasak na djecu i njihovo obrazovanje jer je u ljudskoj prirodi da se zabavljaju.

Utjecaj današnjeg doba i lako dostupnost računalu te video igara, uzrokuje da se sve više djece a i odraslih zabavlja na računalima, netko iz same zabave, drugi iz sporta, natjecateljskog duha ili samo kako bi iskoristili slobodno vrijeme.

Sukob generacija i opiranju tehnologije i video igrama postavlja se pitanje može li išta dobro nastati iz video igara, kratki odgovor je da. Kako u zadnjih nekoliko godina E-sport popularnost raste pokazuje se da se može i od toga živjeti, tokom pandemije kako su pravila bila da se ne izlazi iz domova ako nije potrebno pa su tako ljudi sve više počeli svoje vrijeme provoditi za računalima te se zabavljali kroz video igre što je uzrokovalo nagli rast i popularnost takvom mediju.

Kroz ovaj rad objašnjeno je kakvu korist video igre imaju, kakav utjecaj na ljude stvaraju, te odgovoriti na pitanje imaju li video igre utjecaj na obrazovanje i jeli se može uz pomoć video igara unaprijediti takvo školstvo, omogućiti kroz zabavu i tehnologiju profesorima i studentima da bolje surađuju, kako bi se unaprijedilo obrazovanje.

Video Igre

Video igre, video igrice ili računalne igre su pojam, grupacija koja označava te obuhvaća sve vrste aplikacija koje kroz svoj rad, interakciju korisnika i njihov utjecaj prikazuju rezultat na ekranu. Primarno služe kako bi zabavile korisnika, dali osjećaje igračima, primarno osjećaj sreće i uspjeha.

Žanrovi video igara

Akcijske, eng. Action - Akcijske video igre se primarno baziraju na nekoj vrsti izazovu igraču, kako fizički tako i psihički, cilj povećati igraču adrenalin, i natjerati ga da donesi odluke u trenu kako bi pobijedio, došao do kraja razine te i u konačnici igre.

Primjeri video igara ovakvog tipa: Red Dead Redemtion, Grand Theft Auto, Call of Duty… Svim ovim igrama koje smo naveli kao primjer zajedničko je podizanje razine motoričkih vještina i perceptivnih sposobnosti igrača, što znači umjereno i ciljano korištenje takvih video igara pomaže igračima bolje sakupljati nove informacije jer su u igri izloženi konstanto sa novim odlukama i novim podacima, jedan od razloga zašto je to tako je aspekt adrenalina koji igrači dobivaju dok se zabavljaju i konstantno bore za pobjedu, pa takav način rada kada bi se preslikao na edukacijske svrhe ima pozitivan utjecaj. Igre koje su kao Call of Duty ratne, se izrađuju u obliku simulacija kako bi se primjenjivale u vojno-edukacijske svrhe koje čak Američka vojska i koristi.

Avanturističke, eng. Adventure - Avanturističke igre se primarno baziraju na istraživanju okoline, pa tako igrač preuzima ulogu u kojoj proživljava priču kako bi pobijedio igru, došao do cilja, kraja igre. Svrha ovakvih igara je sveukupna priča i doživljaj igre, pa tako se očekuje od igrača visoka razina koncentracije te kritično donošenje odluka. Ishod priče, igre najčešće ovisi o sveukupnim odlukama igrača, pa tako nije isti kraj svakom igraču.

Neke od igara koji spadaju ovom žanru su: Portal, Monkey Island, Life is Strange… Kako se ovaj žanr znatno razlikuje od akcijskog pa tako i njena svrha u edukaciji je u potpunosti različita, kao što smo već naveli ovdje je najbitnija koncentracija, igraču je nemoguće napredovati kroz igru ako ne zna o čem se radi i šta se događa unutar video igre. Kroz korištenje ovakvih igara igrač mora razumjeti priču, pa tako najčešće ako već ne zna jezik uči kroz interakciju sa

objektima, nakon toga dolazimo do rješavanja problema kojem je igraču zadano, bez kritičnog razmišljanja ne može se napredovati, iako nije svrha davati igraču nova iskustva već omogućiti povezivanje informacija koji se rastežu kroz cijeli narativ priče, pa tako ukomponirana video igra sa ciljanom tematikom unutar edukacije može pomoći kroz zabavu i izazov unaprijediti igrača.

Mozgalice, eng. Puzzle - Mozgalicama je svrha da igračima da nekakav problem ili zadatak koji moraju riješiti kako bi nastavili igrati video igru, kako bi pobijedili. Vrste mozgalica su mnoge, logičke, prepoznavanje motiva, slijedno rješavanje, igre riječima, povezivanje okoline.

Primjeri igara ovakvoga žanra su: Professor Layton, Portal, Human fall flat. Za edukacijske svrhe igre ovakvog tipa bi bile idealne a možda čak i najbolje ako gledamo isključivo dobivanja novog znanja koje bi inače igrač dobio kroz čitanje knjiga. Dovoljno je samo da video igrica ima priču napravljenu na neku temu, i da sve mozgalice budu pametno sastavljene i ukomponirane u video igru, kako igrač ne bi mogao samo preletjeti kroz igru, već bi bio natjeran pamtiti i međusobno povezivati informacije ranije stečene u video igri kako bi u konačnici sve bilo povezano da pobjedi i završi sa video igrom.

Sportske, eng. Sport - Sportske igre kao i samo ime govori imaju tematiku sport, cilj je prikazati sportske i vanjske aktivnosti iz stvarnog životu u obliku video igre sa naravno naglasak na zabavi. Sportske video igre uključuju igre kao što su timski sportovi: nogomet, košarka, te ekstremne sportove poput motorsporta, planinarenje, pa tako imamo i arkadni oblik sportskih video igara koji i dalje zadržavaju temu sporta ali je naglasak više na zabavu nego realnost. Primjer video igara ovakvog žanra su: FIFA, Need for speed, Forza horizon, NBA 2K.

Sportske igra pružaju jako malo prilika na edukacijskoj strani jer jednostavno ne pružaju novog znanja nit zahtjeva od igrača da povezuje informacije, unatoč tome sportske igre se mogu i počinu gledati sve više kao e-sport jer iako igrač nema prevelike fizičke aktivnosti i dalje tijelo radi i osoba kroz dugotrajno igranje razvija tijelo i um. Neki sportovi poput šaha pružaju odlično okruženje i online kako bi igrač mogao stalno biti aktivan.

Simulacije, eng. Simulation - Simulacije stavljaju naglasak na stvarni svijet, kako bi što točnije i preciznije prikazala stvarni svijet u video igri, i cilj je stvoriti prividni svijet na temelju stvarnog u kojem igrači mogu donijeti odluke i isprobati određene stvari bez stvarnih posljedica i ikakvog rizika na stvarni svijet. Takve igre mogu služiti za puno svrha, simulirati ekonomiju, medicinu, ratove, povijesne događaje, također se simuliraju i stvari samo radi zabave i testiranja raznih imaginarnih scenarija.

Neke od igara koje spadaju ovom žanru su: Sims, Cities Skylines, F1 2020, Football Manager.

Simulacijske video igre baš zbog prednosti da mogu simulirati pravi svijet pružaju jako puno mogućnosti za edukacijske sustave, pogotovo u medicinske i vojne svrhe pri čemu studenti mogu bez terenske nastave vidjeti 3D modele, i pomoću virtualne realnosti pristupiti takvom znanju bez da fizički budu prisutni, što omogućuje u situacijama gdje zbog pandemije ili drugih razloga nema kontaktne nastave da studenti ipak okuse i napreduju svoje znanje.

Strateške, eng. Strategy - Strateške igre se baziraju na sposobnost igrača da planiraju i taktički razmišljaju kako bi došli do pobjede, igrač treba biti u smirenom stanju, kalkulirano, logički i strateški razmišljati kako bi postigao što mogući rezultat unutar video igre. Najčešće ovakve igre su borilačkog tipa, pri čemu istraživanje, ekonomija i raspodjela resursa ima najveću ulogu igraču, razlikujemo strateške igre na poteze i u stvarnom vremenu.

Primjeri igara koje bi ovdje spadale su: Civilization, Age of Empires, XCOM, Total War. Ovakve igre također nemaju preveliku svrhu u edukacijskim sustavima, izuzetak bi bio u vojnim edukacijama, jer je sustav previše nasumičan i očekuje da se igrač prilagodi situaciji kako bi u različitim trenucima izvršio drugačiju strategiju, taktiku do pobijede.

Igra uloga, eng. Role playing (RPG) - Video igre ovakvog žanra stavljaju igrača u ulogu jednog lika ili rijetkim slučajevima malog tima likova kojima igrač upravlja kako bi proživio priču na koji način igrač želi u svijetu u kojem je video igra napravljena, na raspolaganju igraču stoji jako puno resursa i mogućnosti jer je naglasak kod ovog žanra sloboda igrača i mogućnost istraživanja svijeta, objedinjuje sve sposobnosti koje igrač posjeduje, znanje jezika, kritičnog razmišljanja, donošenje odluka, refleksi, fizička i psihička izdržljivost, jer u takvim zamišljenim svjetovima, video igrama je granica samo sposobnost igrača, i na različite načine se može doći do cilja ovisno kakav pristup je igrač uzeo. Primjeri video igara za ovaj žanr bi bili: Skyrim, Diablo, Dark Souls, Monster Hunter, Final Fantasy.

Ovakav tip video igre ispada kao idealan za edukaciju jer naime pruža sve što smo do sada spomenuli, od kritičnog razmišljanja, donošenja odluka, povezivanja informacija, psihičkog napora, i ostalo, ipak nije savršeno jer je sustav previše otvoren i pruža previše prilika igračima kako će doći do rješenja, te proizlazi da nakon igranja video igre svaki od igrača izađe sa različitim znanjem i različitim iskustvima.

Svrha video igara

Video igrice diljem svijeta u obliku računalnog medija imaju svoju vlastitu kulturu, pa tako imaju i veliki utjecaj na današnji svijet i današnje društvo. Kultura video igrica skupa sa Internet kulturom rastu u popularnosti te jedna na drugu utječu. Sve većim rastom tehnologije i tehnološkim napretkom, bilo to razvoj računala, mobilnih uređaja, video igrice se konstantno unaprjeđuju skupa sa razvojem tehnologije, jer kako novije generacije ljudi uzrasta uz tu tehnologiju, tako se i njihov način zabave i ponašanja svede na temelju što dožive kroz tehnološki medij.

Video igrice imaju mogućnost biti medij za opuštanje ali i medij za natjecateljske svrhe pa je tako nastao i pojam e-sport, kroz razvoj takve kulture nastala su i mnoga internetska društva, i skupine ljudi koji baziraju cijeli svoj život na temelju video igrica. Također se utjecaj video igrica može primijetiti najviše u trenucima poput pandemije gdje su svi u svojim kućama i moraju pronaći nekakvu zanimaciju kako bi se zabavili, opustili, natjecali, i družili sa svojim prijateljima i bližnjim.

Učinak video igara

Pozitivan - Kao što smo ranije spomenuli svaki žanr video igrica imaju svoje prednosti, što znači da video igrice imaju mogućnosti utjecati na ljudske sposobnosti kao što su:

1. Vizualno-prostorne sposobnosti- mogućnost osobe da se snalazi u prostoru pozicionirati objekte, znati gdje se koji objekt nalazi u prostoriji, brzo se snaći u prostoru, kroz igranje određenih video igrica ta sposobnost se da unaprijediti.
2. Emocionalno stanje- mogućnost dati korisniku pozitivne osjećaje te igrajući miroljubive video igre, potiče se sretno razmišljanje, smanjuju se negativne misli, smanjuje se agresivnost kod korisnika i povećava pozitivne misli.
3. Znanje i edukacija- edukacijske video igre pokrivaju jako puno domena, te su u stanju naučiti korisnika od osnovnih stvari poput osnovne matematike, biologije, čitanje, pisanje, sve ovisno na temelju čega je video igrica bazirana, naučiti strani jezik kroz video igrice nije problem, različite i naprednije sposobnosti poput ekonomije, organizacije video igrice koje se temelje na tim temama su u stanju naučiti kroz zabavu korisnika takve tematike.
4. Razviti tijelo- Postoji niz video igrica kojima je glavna tema vježbanje tijela i duha, jedan primjer takve igre je Wii Fit, takve igre nudi korisniku niz aktivnosti koje drže tijelo aktivnim, mogućnost vježbanja, treniranja, razgibavanja, i u fokusu se nalaze tri stvari: potrošnja energije, trajanje

aktivnosti, preferencija aktivnosti. Ljudi koji nemaju preveliku motivaciju prema vježbanju i treniranju tijela, kroz zabavu i video igre ipak dobiju tu želju da isprobaju i na takav način motivacija raste, te uz razvoj tijela i rast sreće, također utječu na emocionalnu stabilnost korisnika.

5. Društvene sposobnosti- igrajući video igre koje imaju primarni fokus društvo, timski rad internetskim putem kako bi se u igri moglo napredovati, znatno povećava komunikaciju i druženje među korisnicima što se preslikava i na stvarni život i utječe pozitivno na korisnika.

Negativan - Što znači da video igre imaju mogućnost utjecati na individualca negativno na moguće sljedeće načine:

1. Ovisnost- kao što alkohol, nikotin, tako i video igrice imaju mogućnost stvaranja ovisnosti, naravno veći rizik je kod mlađih ali svi su mogu postati ovisni o video igricama što naravno utječe negativno na život individualca, pri čemu osoba provede previše vremena pred ekranom da izgubi osjećaj za vrijeme.
2. Gubitak motivacije- korisnik može dobiti toliki interese da igra video igru da izgubi svu motivaciju za stvari u stvarnom životu pa tako krene sam sebi štetiti, štetiti svojoj edukaciji, poslu, životu.
3. Štetiti zdravlju- suzbijanje emocija, suzbijanje stvarnih problema, stvaranje dodatnog stresa, ovisno o kakvoj video igrici je riječ ako je video igrica puna strahova, puna likova sa psihičkim problemima, ili je jednostavno video igra van sposobnosti korisnika, u stanju su se pojaviti strahovi, osjećaji sramote, slabosti. Naravno postoje i fizičke ozlijede ako korisnik više sati uzastopno provodi za video igricom koja zahtjeva stalnu interakciju, korisnik si je u stanju zbog stresa ozlijediti ruke.
4. Društveni problemi- kroz udubljenje u svijet video igara, korisnik zaboravlja na prijatelje, obitelj u stvarnom životu, i takvi odnosi krenu stvarati probleme poput, izdvajanja iz društva, samoizolacija korisnika, nedostatak interakcije sa stvarnim osobama.
5. Negativni osjećaji- lako dođe do osjećaja mržnje prema drugima, mržnja prema sebi ako su video igre frustrirajuće teške, napete, van korisnikovih sposobnosti.
6. Fizički oslabiti- bez igranja video igrica koje uključuju vježbanje u stvarnom životu osoba je u stanju kroz velik period vremena oslabiti fizički i štetiti svom zdravlju.

Pozitivno vs Negativno

Naravno to su samo pozitivne strane na koji način video igre utječu na individualca, ali isto tako ima pozitivnih stvari na globalnoj razini poput,

mogućnost simuliranja stvari u kontroliranim uvjetima bez straha ugrožavanja stvarnih života, utjecaj na ekonomiju dobar dio današnjeg tržišta su video igre. Naravno sve ove negativne stvari koje smo naveli mogu se pojaviti, isto tako mogu se i izbjeći ako je korisnik u stanju sam sebe kontrolirati, znati svoje vlastite sposobnosti i znati koje su video igre namijenjene za korisnika su jedan od prvih koraka kako bi se izbjegli negativni problemi.

Uočljivo je da sve ovisi o pojedincu, korisniku koji se koristi video igricama, naravno moglo bi se argumentirati da pozitivni aspekti prevladavaju nad negativnim.

Može se zaključiti ako uzmemo pozitivne stvari u obzir, i implementiramo video igre u obrazovanje, postavimo osobu koja nadgleda poput profesora, tada bi trebali samo i pozitivne rezultate video igara i dobivati. U kontroliranim uvjetima iako tehnologija dopušta da korisnik pristupa video igricama bilo gdje bilo kada, i dalje kroz odgoj, rast, i edukaciju osoba bi trebala više pozitivnih učinaka osjetiti nego negativnih.

Implementacija video igara u obrazovanju

Cilj korištenju video igara u obrazovanju bi bio jednostavan, povećati studentima motivaciju, interaktivnost i znanje kroz zabavu. Konačni rezultat korištenja video igara i konačni cilj bi bio dati studentima transverzalne sposobnosti koje bi bile: kritično i inovativno razmišljanje, medijska i informacijska pismenost, društvene sposobnosti te unutarnje sposobnosti poput emocionalne inteligencije, samo pouzdanje i slično.

Mozgalice - efektivna implementacija, dobro napravljena mozgalica mora sadržavati karakteristike igrača kao što su godina, spol, edukaciju za koju je mozgalica namijenjena, dok bi razine i težina mozgalica ovisila o igračevim sposobnostima i rezultatima samog igranja. Izgradnja same mozgalice morala bi sadržavati priču kako bi igrači imali šta pratiti dok igraju da imaju nekakav sporedni motiv uz učenje, drugi dio bi bio težina mozgalica ne smiju biti prelagane a ni preteške stvari kako igraču ne bih bilo previše naporno ili previše dosadno, a treća stvar bi bila sama mozgalica, zadatak koji se daje, jer zabavan dio svake mozgalice dolazi od dekodiranja, pamćenja i sastavljanja dijelova kako bi se riješila, jer ipak zabava mozgalice leži u rješavanju problema. Dalje koji bi aspekti bili bitni za konačan proizvod kako bi video igra bila uspješna jest da se igračima da jasno definirana pravila, rješenja koja mogu dobiti prema kojima trebaju ciljati, te mogućnost rješavanja na logički ili konceptualni način u određenom vremenu ili uvjetovano ostalim dijelovima igre kako bi dobili taj osjećaj učenja kao dok slušaju profesore u kratkom vremenu, ili čitaju literaturu u određenom vremenskom intervalu.

Dobra stvar dolazi i od same izgradnje mozgalica jer uzimaju se neke situacije iz stvarnog života i problema, iz različitih područja jer mi možemo i zagonetku prikazati kao mozgalicu, „Escape Room“ poznati način zabave također je mozgalica, sve to pridonosi sudionicima takvim igrama da nešto nauče i donose odluke, i kako bi uspjeli stvarni život ukomponirati u našu video igru kao mozgalicu moramo gledati koje su nam sposobnosti potrebne i za rješavanje takvih situacija u stvarnom životu, to bi bile sposobnosti prilagođavanju varijablama, nepoznanicama jer ne znamo kako će se situaciju u stvarnom životu razviti, također trebamo primijeniti znanje domene na kojoj radimo i koristiti se metodama i informacijama koji se baziraju na toj domeni područja, i opet spominjemo najbitniju stvar korištenje kritičnog razmišljanja i donošenja odluka.

Simulacijske - Simulacijske video igre bi trebale biti tu da nadopunjavaju edukacijske elemente koji su nedostupni u pravom vremenu, cilj bi bio nadopuniti kognitivne sposobnosti, omogućiti studentima da samo sebe ocjene kako bi evaluirali gdje rade greške i gdje se mogu unaprijediti.

Simulacije pružaju okruženje iz stvarnog svijeta, sa stvarnim podacima i informacijama koji omogućuju da studenti mogu eksperimentirati sa raznim strategijama, preuzeti različite uloge, i biti odgovorni za svoje postupke i djela. Omogućuju da studenti rekonstruiraju, stvaraju nove probleme te omogućuje rješavanje istih, pružaju priliku studentima da povežu teoriju, koncepte i ideje sa stvarnim situacijama. Mogućnost simulacije je naravno i pružanje studentima razvoj društvenih sposobnosti, kroz zajednički rad unutar većih simulacija razvoj timskog rada, donošenje grupnih odluka, unaprjeđivanje projektnog menadžmenta, sposobnosti i razvoj komunikacijskih vještina. Na takav način studenti kroz zajednički rad imaju aktivnost sa predmetom i kroz zabavu napreduju i uče, dobivaju izravni povrat informacija pri čemu je to jedno od najbitnijih stvari tokom edukacije, postavljaju konkretne ciljeve unutar simulacije i dobivaju jasnu motivaciju.

Vidimo da je mogućnost korištenja simulacijskih igara vrlo velika, u različitim kontekstima poput fizičkim, društvenim, kulturološkim, tehnološkim, logičkim, nije ni bitno gdje se izvode, video igrice i simulacije mogu se izvoditi od doma, fakulteta, škole, van školskim aktivnostima. Naravno korištenje simulacija u svrhu edukacije uvijek bi bilo bolje izvoditi na nastavi na fakultetu radi motivacije, pri čemu na nastavi profesori imaju utjecaj na interakciju studenta sa simulacijom. Potencijal leži u mogućnosti prilagođavanja individualnim sposobnostima, uskladiti brzinu učenja, interese studenta, te određenim potrebama individualca, što bi utjecalo na količinu znanja koje se može prenijeti kroz simulaciju i kroz generacije prikupljeno iskustvo.

Virtualna Realnost, eng. Virtual Reality (VR) - Mogućnost korištenja virtualne realnosti unutar edukacije je velika, jer omogućuje studentima da budu skupa na istoj nastavi, iako nisu fizički prisutni jedni kraj drugih, što uključuje mogućnost da se studenti pronađu diljem svijeta u istom digitalnom prostoru, također studenti mogu biti fizički u istoj prostoriji a kroz virtualnu realnost da posjete drugu državu, firmu, radionicu, bilo šta a da ne moraju fizički biti prisutni u njoj. Omogućuje studentima jedinstveno iskustvo u koje se mogu opustiti, a profesorima je omogućeno lako upravljati i pratiti što studenti rade. Pogotovo u situacijama poput pandemije virtualna realnost pruža mogućnost nalik terenskoj nastavi bez prisutnosti studenata uživo, kako bi studenti bili aktivni i sudjelovali u radu.

Postoje VR sustavi koji su prilično jednostavni pa vam treba samo kartonski okvir u koji umetnete mobilni telefon na kojem pokrećete VR aplikaciju, dok drugi imaju nešto sofisticiranije dodatke koji vam omogućuju da dublje „uronite“ u novi svijet, pri čemu se koriste rukavice, slušalice ili neki drugi elementi, a moguće je čak i kretanje unutar 3D prostora.

VR uređaji za glavu su praktični zbog njihove dostupnosti, prenosivosti, cijeni i veličini, ne zauzimaju puno prostora što omogućuje studentima da bilo gdje i bilo kada koriste, naravno uređaj je zaseban i moguće ga je koristiti bežično bez spajanja na računalo, nije teško za koristiti a glavna prednost VR uređaja za glavu što pruža bolje virtualno iskustvo, jer je iskustvo puno realističnije koje studenti teško zaborave.

VR u edukaciji naravno ima veliku prednost da pomogne djeci sa posebnim potrebama, pogotovo djeci sa autizmom. Na način da kroz zabavu djeci prikazuju najbitnije i ciljane podatke koje oni mogu primiti, shvatiti i primijeniti kako bi mogli u određenim situacijama samostalno funkcionirati.

Proširena stvarnost, eng. Augmented Reality (AR) - Proširena stvarnost AR u edukaciji ima mogućnost transformirati interakciju korisnika, studenta sa aplikacijama u nešto realističnije i grafički prikazano uživo. Omogućuje prebaciti grafiku generiranu na računalu u 3D obliku u našoj okolini, bilo to kod kuće, dvoranama, javnim površinama ili bilo gdje skroz dok imamo uređaj kroz koji možemo percipirati tu stvarnost, u današnje vrijeme najdostupniji uređaj i najlakše za koristit bio bi pametni mobitel. Što znači kroz kameru od mobilnog uređaja vidimo računalno generirani objekt u stvarnom vremenu na ekranu od uređaja što omogućuje veću razinu interakcije studenata u njihovoj okolini, naravno proširena stvarnost samo integrira našu postojeću okolinu sa virtualnom, virtualne objekte sa stvarnim svijetom.

Proširena realnost pruža veliku interaktivnost pir čemu profesori imaju priliku kroz video igre elemenata prikazati koncepte i primjere u virtualnom obliku

kako bi proširena realnost bila potpora normalnoj nastavi i literaturi služila kao suplement, povezivanjem vizualnog 3D oblika sa tekstualnim materijalima studenti lakše pamte. Primjeri takvih igara su Dinosaur 4D+, Element 4D, Minecraft Earth.

Zaključak

Video igrice primarno su namijenjene da zabave korisnika, dok obrazovanje služi primarno da pruži novo znanje svojim korisnicima. Na prvi pogled video igre i školstvo nemaju ništa zajedničko, no ipak ovaj rad pokazuje kako ta dva pojma mogu skupa nešto novo i pozitivno napraviti jer oboje zahtijevaju interakciju i motiviranost svojih korisnika. U radu je prikazano da video igre imaju svoj vlastiti i široki svijet prikazan kroz niz žanrova u koje se mnoge video igre uklapaju, te da svaki žanr pruža svoje jedinstvene prednosti.

Među svim žanrovima unutar svijeta video igara, mozgalice su se pokazale kao najučinkovitije za visokoškolsko obrazovanje jer pružaju korisnicima povezivanje informacija, povezivanje znanja i iskustva kako bi pobijedili video igru, što razvija korisnika i njegove sposobnosti pamćenja, međusobno povezivanja informacija i podataka kako bi došao do nekog zaključka, rješenja, unaprjeđuje kritično razmišljanje i ako je uz to mozgallica napravljena na temu literature, gradiva edukacije koje korisnik pohađa priprema se i uči uz zabavu, naravno video igra sama po sebi treba biti samo suplement gradivu i glavnoj literaturi.

Ovaj rad je prikazao dodatno i tehnologiju koja bi bila najbolji suplement uz redovito nastavu, naravno riječ je o virtualnoj realnosti, iako VR se najčešće koristi za igranje video igrica, isto tako se može primijeniti sa zabavom u nastavi, pri čemu se stvara virtualno okruženje u obliku digitalizacije svijeta, ili prikazivanja virtualnih objekata u stvarnom svijetu što bi bilo proširena realnost AR, skupa naravno pružaju najbolje oba svijeta što čini mješovitu realnost, na taj način studenti kroz zabavu i 3D objektima kroz njihove uređaje, pametne mobitele mogu unaprijediti svoje znanje i nadopuniti svoju edukaciju sa jedinstvenim iskustvom i velikom razinom motivacije.

Može se zaključiti da mogućnosti korištenja video igara u obrazovanju postoje, moguće je, pozitivno je, isplativo je, korisno je, i puno potencijala za napredak i razvoj, sve u svemu rad prikazuje neke od bitnih stavki na šta se treba pripaziti, ali uz napredak tehnologije, video igre i virtualna realnost zasigurno su budućnost visokoškolskog edukacijskog sustava i već niz država radi na testiranju i implementaciji.

Reference

[1] Bandić Kristina, Ružić Baf Maja, „Video igre i slobodno vrijeme Video games and free time“ 2020, https://urn.nsk.hr/urn:nbn:hr:137:304922

[2] J. Birt, Z Stromberga, M. Cowling, C. Moro, „ Mobile Mixed Reality for Experiential Learning and Simulation in Medical and Health Sciences Education“ Dostupno na: https://www.mdpi.com/2078-2489/9/2/31/htm

[3] Sara Prot, Craig A. Anderson, Douglas A. Gentile, Stephanie C. Brown, and Edward L. Swing (2014). The Positive and Negative Effects of Video Game Play , https://drdouglas.org/wp-content/uploads/2019/03/14PAGBS.pdf

[4] M. Matijević, T. Topolovčan, „Multimedijska didaktika“ Zagreb: Školska Knjiga, 2017.

Popularnost žanrova računalnih igara

Julijan Brlić
Fakultet organizacije i informatike, Sveučilište u Zagrebu
julbrlic@foi.hr

Sažetak

Računalne igre klasificirane su kroz mnoštvo žanrova pomoću kojih ljudi odabiru omiljene igre. Svrha ovog rada je klasificirati žanrove računalnih igara te dati osvrt na njihovu popularnost. Svaki navedeni žanr ima za primjer po jednu igru, te je kod svakog dan osvrt na njegovu popularnost.

Ključne riječi: popularnost, žanrovi, računalne igre

Uvod

Predmet ovog rada je popularnost žanrova računalnih igara. Kroz rad biti će prikazana povijest računalnih igara, odabrani žanrovi računalnih igara te preferencije žanrova računalnih igara. Računalne igre su nešto što napreduje eksponencijalno. Kroz povijest možemo vidjeti kako su se igre razvijale, od vrlo jednostavnih do modernih i kompliciranih. Za razvoj igara najzaslužnije su tehnologije koje se neprestano razvijaju i omogućuju bolji doživljaj same igre. U ovom radu obraditi će se deset najpopularnijih žanrova, njihove posebnosti te popularnost, te preferirani žanrovi računalnih igara prema parametrima poput dobi, spola, zarade i dužine vremena igranja pojedinih žanrova.

Povijest računalnih igara

Povijest računalnih igara započinje 1958. godine kada je William Alfred Higinbotham uz pomoć analognog računala kreirao igru *Tennis for Two,* koja se smatra prvom igrom u povijesti. Igra *Tennis for Two* kreirana je u demonstrativne svrhe. 1962. godine Steve Russell kreirao je igru *Spacewar!* na Sveučilištu Massachusetts Institute of Technology (MIT). *Spacewar!* je također napravljena u demonstrativne svrhe. (Lowood, bez.dat.)

1972. godine osnovan je Atari. Atari je proizvodio igraće konzole i kreirao je prvu globalno popularnu igru *Pong*. Igra je bila vrlo jednostavna, cilj igre bio je ne promašiti lopticu koja je putovala između lijevog i desnog ruba ekrana. Atari je izradio kutiju u koju je integrirao računalo, ekran i aparat za kovanice. Prvi put u povijesti igra postaje dostupna široj javnosti. Tokom 1980-ih godina pojavljuju se prva osobna računala, što rezultira krahom konzola i Atarijevim bankrotom. 1985. godine Nintendo predstavlja svoju konzolu NES, tehnologije

bivaju redefinirane, igre postaju kompleksnije i imaju bolju grafiku od prethodnika.

Sredinom 1990-ih godina Sony je lansirao konzolu PlayStation, što je bio ogroman iskorak u igranju igara na konzolama. Kreatori igara okreću se kreativnim idejama i inspiraciju traže u povijesti, tako nastaju igre *Age of Empires, Command & Conquer, Tomb Raider i Wolfenstein 3D.* Zanimljivo je da je *Wolfenstein 3D* prva igra oko koje se počinju voditi debate da li se nasilje u video igrama reflektira i u stvarnom životu.

Početkom 2000-ih stvorene su online igre, no s obzirom na to da je internet još uvijek spor, nerasprostranjen i skup, igrači koriste lokalne mreže pomoću kojih zajedno igraju igre jedni protiv drugih. 2004. godine na tržište dolazi igra *World of Worcraft - WOW.* WOW je prva igra koja se prvenstveno igrala online putem interneta.

Moderno doba igara počinje nakon 2010. godine. Pojavom pametnih telefona (eng. Smartphone) igre se mogu igrati bilo gdje i bilo kad. Eksponencijalni razvoj tehnologije omogućava igranje istih igara na različitim platformama, igre donose zaradu igračima, stoga igranje postaje sport. Igrači snimaju svoju igru te okupljaju milijune obožavatelja koji gledaju snimke igre na platformama kao što su YouTube i Twitch. (Rechsteiner, bez dat.)

Žanrovi računalnih igara

Žanr je stil ili kategorija nečega, na primjer žanrovi filmova, žanrovi tv serija ili žanrovi računalnih igara. Ovdje ćemo govoriti o žanrovima računalnih igara. Računalnim igrama pridružujemo žanrove kako bi ih mogli grupirati. Primjerice, igre s istom mehanikom i stilom igranja svrstavamo u isti žanr. Sve igre uklapaju se u jedan ili više žanrova, a definiraju se strukturnim, tematskim i funkcionalnim kriterijima. Postoje određene poteškoće u grupiranju igara u žanrove. Razlog tome leži u postojanju mnoštva žanrova te se pojedine igre mogu pridružiti većem broju žanrova. (Kelly, bez dat.)
S obzirom na to da je žanrova mnogo, u ovom radu biti će obrađeno deset najpopularnijih žanrova u 2020. godini prema istraživanju provedenom od strane Straits Research-a. (Straits research, 2020)

Akcijske igre

Ovaj žanr naglašava reflekse igrača, koordinaciju vida, pokreta ruku i vremena reakcija. Sastoji se od tri glavna mehanizma, a to su: trčanje, skakanje i napad. Radnja akcijskih igara najčešće vodi igrača kroz priču napredujući po razinama do glavnog cilja. Kako igrač napreduje kroz igru izazovi postaju sve veći i teži.

Takve igre pružaju jedinstveno iskustvo igračima zbog živopisnih slika te stvaranja napetosti u iščekivanju nepredvidivih događaja. (Oxford, 2018)

Popularnost akcijskih igara prikazati ćemo kroz igru Grand Theft Auto – GTA. GTA pruža igraču slobodu kakvu ne može iskusiti u stvarnom životu, doslovno sve je dopušteno. U jednom dijelu igre može voziti koje god vozilo poželi, skakati sa zgrada padobranom, plivati morem, pucati iz raznih oružja, izazvati uličnu borbu i pobjeći, ukrasti bicikl, avion, brod… Drugi dio igre čine misije koje igrač prolazi, glavne i sporedne. Moramo spomenuti i kodove za dohvat nekih funkcija ili vozila (eng. Cheat code). Primjerice, pomoću kodova možemo dobiti helikopter, motor, auto, avion. Također možemo manipulirati vremenom, isključiti gravitaciju, letjeti autima, uključiti besmrtnost i dr. Zbog svega navedenog ne čudi da igrači obožavaju ovakve igre u kojima mogu biti slobodni te pobjeći od ograničenja iz stvarnog života.

Sportske igre

Sportske igre su jedan od najpoželjnijih žanrova igara u svijetu. One simuliraju razne vrste sportova u kojima igrač, pomoću stvarnih sportskih taktika i pravila, pokušava pobijediti protivničke momčadi kojima upravljaju umjetna inteligencija (eng. Artificial intelligence) ili botovi. Sportske igre krase visoki grafički detalji igrača ili natjecatelja. Takve igre imaju nove nastavke uglavnom svake godine jer prate stvarni svijet sporta.

Ovaj žanr popularan je kod onih igrača koji žele preuzeti ulogu igrača iz stvarnog svijeta sporta. Igrači najviše igraju sportove kojima se bave i u stvarnom životu. Ovaj žanr može poslužiti i kao orijentacija za sport koji će igrači odabrati u stvarnom životu. Sportske igre vrlo su popularne jer potiču natjecateljski duh, ali to ne znači da će uvijek svi natjecatelji biti sretni s ishodom igre. (Straits research, 2020)

Avanturističke igre

Inspiracija za nastanak avanturističkih igara pronalazi se u raznim književnim djelima, znanstvenoj fantastici, filmovima i drugim izvorima iz svijeta zabave. Avanturističke igre temelje se na refleksima, prikupljanju predmeta, rješavanju zagonetki i istraživanju mapa. Priča igre je prepuna radnji i uzbudljivih slika koje vode igrače kroz avanture. Kod ovakvog žanra pripovjedač uvodi igrača u pojedinosti određene misije. Avanturističke igre mogu imati razne tematike i to ih čini jednim od najomiljenijih žanrova u svim dobnim skupinama.

Popularnost ovom žanru daju tri glavne karakteristike avanturističkih igara: priča, rješavanje zagonetki i istraživanje okoline. Priča kojom se igrač uvodi u igru može krenuti u različitim smjerovima, ovisno o igračevom odabiru, stoga

se ista igra može odigrati svaki put s drugačijim završetkom. (Straits research, 2020)

Battle Royale igre

U ovom žanru igrači se međusobno bore za resurse i oružja s kojima kreću u obračun s preostalim igračima u areni. Kad igrač bude eliminiran, više se ne može vratiti u akciju nego mora započeti novu igru. Onaj tko ostane zadnji u međusobnim borbama, eliminirajući sve protivnike, smatra se pobjednikom. Ovaj žanr je popularan jer su battle royale igre jednostavne za igru tj. sama radnja igre je vrlo jednostavna za shvatiti. Ovakav tip igre dostupan je gotovo svima te je igraju svi uzrasti. Igra tjera igrače da razmišljaju i reagiraju brzo vodeći se svojom intuicijom. Zanimljivo je da kod velike većine battle royale igara nema mogućnosti ponovnog oživljavanja (eng. Respawn) tokom same igre, kad igrač umre igra je za njega gotova (Horowitz, 2019)

Igre uloga

Igre uloga (eng. Role-Playing games - RPG) su žanr u kojem igrač preuzima kontrolu nad izmišljenim likom koji ispunjava razne zadatke u imaginarnom svijetu. Kroz igru lik napreduje povećavajući svoju statistiku i razine, poboljšava svoje vještine, sposobnosti, snagu, oružja i oklope. Odabirom pojedine rase, karakteristika i sposobnosti određenog lika (čarobnjak, lopov, trol, patuljak) igrač ima direktan utjecaj na tok igre, te će tokom igre pojedini likovi biti u prednosti nad drugima. RPG igre pružaju ono što je u stvarnom životu nemoguće. RPG igre nam privuku pozornost i uvuku nas u svijet fantazije i magije. Igrači se „izgube“ u prekrasnim mapama kroz koje prolaze. U RPG igrama uvijek možemo nešto dodatno raditi, npr. odraditi pojedine misije koje nemaju utjecaj na glavnu misiju, istražiti mapu izvan granica misije ili unaprijediti sposobnosti lika s kojim igramo. Igranje uloga je moguće kroz mnoštvo likova od kojih niti jedan nema iste osobine, pa tako svaki lik pruža potpuno drugačiju perspektivu na istu igru. (Technopedia, bez dat.)

Trkaće igre

Trkaće igre vjerne su kopije stvarnih utrka u bilo kojem okruženju i bilo kojem terenu bili to voda, zrak ili zemlja. Najčešće trkaće igre vezane su za svijet automoto sporta. U njima igrač upravlja određenim vozilom natječući se protiv drugih igrača, protiv vozila kojim upravljaju botovi ili protiv samog sebe pokušavajući postići što bolje prolazno vrijeme. U prošlosti su bile najznačajnije arkadne utrke, a danas su to trkaće simulacije koje vjerno prikazuju vozila i staze iz stvarnog života. Trkače igre popularne su zbog toga što nam pružaju da isprobamo nešto što u stvarnom životu vjerojatno nikad ne bi radili. Tko ne bi želio isprobati koliku maksimalnu brzinu mogu određeni auto ili motor postići?

Tko ne bi vozio bolid svog omiljenog vozača Formule 1? Prije i tokom vožnje, igrači se susreću s mnoštvom parametara koji se uslijed utrke mogu promijeniti. Primjerice, tokom utrke može početi padati kiša što drastično mijenja dinamiku igre jer vožnja po kiši je skroz drugačiji tip utrke. Igrači koji se najbolje snađu ostvare i bolje rezultate.

Borbene igre

Borbene su igre žanr u kojem, kako i samo ime govori, igrač stupa u borbu s protivnikom kojim upravljaju druga osoba ili umjetna inteligencija. Posebnosti ovog žanra su specifični potezi generirani ili brzim pritiscima tipki ili unosom određene sekvence tipki u kombinaciji s mišem/ joystickom. Igrač svog lika promatra s bočne strane, a nasuprot njega je protivnik. U novije doba borbene igre prelaze s 2D prikaza na 3D prikaz. Borbene igre postoje preko 20 godina, a popularnost su stekle kad su se igrale kao arkadne igre. U borbenim se igrama igrači bore međusobno, s tim da igra najčešće, prije samog početka, sortira igrače prema njihovim razinama sposobnosti. Upravo zbog igre „jedan na jedan“, ovaj žanr razvija veliku kompetitivnost između igrača. (Technopedia, bez dat.)

Strateške igre u stvarnom vremenu

Posebnost je navedenog žanra to što se igra odvija u stvarnom vremenu, pa igrači nemaju kontrolu nad vremenom ni tempom, čime se vrši pritisak na igrače da brzinski donose odluke na temelju pruženih ograničenih informacija. Igra tjera igrače da donesu odluke izvan zone komfora te da procijene učinkovitost i kvalitetu svojih odluka u vrlo kratkom vremenu. U igri igrač upravlja jednim likom ili skupinom likova, gledajući ih uglavnom iz ptičje perspektive. Igrači vole strateške igre jer ih tjeraju na razmišljanje, planiranje i donošenje pametnih odluka. Igrači usavršavaju svoje vještine strateškog razmišljanja u okruženju gdje ništa nije „na kocki“, osim ponosa koji donosi pobjeda. Strateške su igre puno jednostavnije od stvarnog života jer je u njima manje varijabli koje moramo uzeti u obzir da bi ostvarili cilj. (Wayward, 2015)

Simulacijske igre

Simulacijska igra dizajnirana je da prikaže aktivnosti iz stvarnog života. Igrač ovog žanra slobodno kontrolira lik ili okruženje, najčešće bez strogo definiranog cilja. Postoje razne vrste simulacija iz stvarnog svijeta, od simulacija letenja, vožnje podmornice, farme pa čak i simulacije života u virtualnom gradu. Ovisno o vrsti simulatora, igrač mora imati predznanje kako bi mogao upravljati određenim vozilom. Popularnost ovog žanra leži u činjenici da simulacijske igre repliciraju različite stvarne aktivnosti. Tako ljudi mogu isprobati poslove koje u stvarnom životu nikad ne bi mogli, primjerice izgraditi vlastiti grad, voziti vlak

ili zrakoplov, operirati ljude u ulozi kirurga, boriti se u ratu, isprobati neki ekstremni sport… Igrači s godinama postaju sve zreliji, te prelaze sa žanrova poput igara pucanja na žanrove s manje akcije i adrenalina, kao što su simulacijske igre. (Technopedia, bez dat.)

Igre pucanja

Radnja ovih igara odvija se iz perspektive prvog lica igrača. Kao što i samo ime govori, radi se o igrama u kojima igrači samostalno ili uz pomoć suigrača prolaze zadane scenarije te upotrebom oružja pokušavaju pobijediti protivnike. Popularnost ovog žanra leži u tome što igraču pruža snažan osjećaj kontrole nad nekim događajima i situacijama tokom igre te ga tjera da donosi odluke u vrlo kratkom vremenu. Igrač koji voli ovaj žanr nije nužno nasilan u stvarnom životu. Mnogim igračima ovaj žanr izvrstan je ispušni ventil za stres, ljutnju, uzrujanost, negativne emocije… Igre pucanja na samom su vrhu žanrova igara koji sudjeluju u raznim esport natjecanjima u svijetu, a najviše ih igraju muškarci.

Preferirani žanrovi

U ovom dijelu rada prikazati ćemo preferirane žanrove računalnih igara kroz nekoliko zanimljivih istraživanja. Analizirati ćemo najpopularnije žanrove kod određenih skupina igrača (žene, muškarci, starija populacija, mlađa populacija, igrači koji plaćaju pretplatu za igre). S obzirom na to da je industrija računalnih igara vrlo profitabilna, saznati ćemo koliko zarađuju pojedini žanrovi kroz natjecanja u esport-u. Prikazati ćemo na koje je žanrove pandemija COVID-19 utjecala pozitivno, a na koje negativno.

Preferirani žanrovi kod igrača starijih od 50 godina

Iz istraživanja provedenog u SAD-u 2016. godine dobili smo informacije o popularnosti žanrova video igara kod igrača starijih od 50 godina. Igrači stariji od 50 godina najviše vole igre s kartama (46%) i puzzle igre (44%). Njihov treći izbor je trivia (19%), slijede kockarske igre (14%), logičke igre (11%), arkadne igre (6%), igre pucanja, sportske igre i akcijsko-avanturističke igre (5%), igre utrkivanja, igre strategija i edukacijske igre (3%), simulacijske igre i igre uloga (2%), borbene igre (1%) i ostali žanrovi (10%). Iz ovih podataka zaključujemo da većina starijih igrača ne preferira žanrove s puno akcije i adrenalina. Žanrovi poput igara s kartama i puzzle igara smanjuju stres i potiču pozitivno raspoloženje, stoga ih stariji najviše vole. (Anderson, 2016)

Preferirani žanrovi igara kod djece u dobi od 4 do 13 godina

Joan Ganz Cooney Center 2016. godine proveo je istraživanje u kojem je sudjelovalo 696 djece u dobi od 4 do 13 godina s ciljem prikupljanja saznanja

koje žanrove igara preferiraju djeca spomenute dobi. Primijetili su da djeca imaju četiri glavna žanra: puzzle igre (78%), zatim avanturističke igre (64%), simulacijske igre (60%) te igre utrkivanja (44%). Ostale žanrove djeca igraju znatno manje od navedenih. Tako samo 22% djece igra igre uloga; 20% djece igra glazbene, logičke i borbene igre, dok sportske igre igra njih 19%. Očekivano je da djeca te dobi najmanje vole pucačke igre (17%), a najviše puzzle igre. Takve igre im razvijaju koncentraciju i strpljenje, logiku, razumijevanje određenih procesa, prepoznavanje boja, uzoraka, oblika, itd., što bude nagrađeno napretkom na sljedeću razinu. (Joan Ganz Cooney Center, 2016)

Preferirani žanrovi za vrijeme pandemije COVID-19

S obzirom na to da se ovaj rad izrađuje za vrijeme pandemije COVID-19, prikazati ćemo u kojoj mjeri se povećalo vrijeme provedeno igrajući pojedine žanrove. Vrijeme provedeno igrajući Battle Royal žanr povećalo se za čak 143%, slijede MOBA sa 120%, Borbene igre sa 100% te Simulacijske igre s povećanjem od 50%. Značajan je rast onih žanrova u kojima sudjeluje puno igrača u istoj igri. Jedan od razloga je, zasigurno, nemogućnost druženja u stvarnom svijetu zbog raznih nametnutih ograničenja kretanja. Niski rast žanrova zabilježile su igre pucanja (8%), sportske igre (6%) i avanturističke igre (3%). S druge strane, žanrovi koji su ostvarili pad su: igre igranja uloga (-15%), akcijske igre (-27%) te puzzle igre s najvećim padom (-57%). (Gough, 2020)

Popularnost žanrova igara prema spolu

Istraživanje provedeno 2017. godine na uzorku od preko 270.000 igrača govori nam kakve su preferencije ženskog i muškog spola u odabiru žanrova igara. Ženski spol prednjači sa 69% u samo dva žanra, a to su puzzle igre (npr. Candy Crush Saga) i simulacijske igre (npr. The Sims), dok 31% muškaraca odabire te žanrove. Najveća je razlika u sportskim igrama gdje su muškarci zastupljeni u 98% slučajeva. Udio muškaraca u trkaćim igrama je 94%; u igrama pucanja 93%; online borbenim igrama s mnogo igrača (eng. Multiplayer online battle arena) 90%; igrama strategija 89%; otvoreni svijet poput igre Grand Theft Auto V, 86%; online igrama s mnogo igrača (eng. Massively multiplayer online game) s naglaskom na fantaziju i čarobnjaštvo, poput igre World of Warcraft, 84%; avanturističkim igrama 82%; akcijskim igrama 80%; igrama izgradnje gradova 78%; igrama uloga 74%; online igrama s mnogo igrača (eng. MMO game) s temama iz znanstvene fantastike, kao što je igra Star Wars, 64%. Rezultati istraživanja tjeraju nas da se zapitamo preferiraju li muškarci navedene žanrove jer u većini njih nema protagonista ženskog spola? Dalje zaključujemo da žene najviše vole igre koje simuliraju stvarni život te logičke igre koje testiraju vještine rješavanja zagonetki i problema, dok muškarci preferiraju adrenalinske i natjecateljske igre.

Popularnost žanrova unutar eSport-a

Popularnost žanrova unutar eSport-a prikazat ćemo pomoću fonda novčanih nagrada osvojenih u razdoblju od 2010. do 2020. godine. Žanr koji prednjači s vrtoglavih $358 milijuna je MOBA, unutar kojeg samo Dota 2 nosi udio od $229 milijuna. Drugo mjesto zauzimaju igre pucanja s $236 milijuna, a glavna u ovom žanru je Counter- Strike: Global Offensive na koju otpada $107 milijuna. Posljednji od tri glavna žanra je Battle royale s $142 milijuna te igrom Fortnite koja, s $99 milijuna, nosi većinu nagrada. Sljedeći žanrovi imali su znatno manji fond nagrada. Prvi takav žanr su strategije s $68 milijuna, zatim igre s kartama (eng. Collectible Card Game) $35 milijuna, slijede sportske igre s $27 milijuna, borbene igre s $20 milijuna i trkaće igre s $8 milijuna. (Willingham, 2018)

Preferirani žanrovi igrača koji plaćaju pretplatu za igre

Iz istraživanja provedenog 2020. godine na uzorku od 13 000 igrača iz cijelog svijeta u dobi iznad 13 godina, saznajemo omiljene žanrove igrača koji igre igraju na način da plaćaju pretplatu za iste, tj. ne kupuju određenu igru, nego uplate određeni iznos pretplate kroz neki vremenski period. Iz grafa (Grafički prikaz 7) vidimo da su akcijske igre najpopularniji žanr s ukupnom preferencijom od 45%, slijede battle royale sa 40%, igre pucanja imaju 37%, sportske igre (34%). U donjem dijelu grafa nalaze se MOBA (29%), strategije (25%), borbene igre (24%), igre igranja uloga (20%) te simulacije na koje otpada 19% preferencija igrača (Clement , 2021).
Iz ovog istraživanja možemo zaključiti da igrači koji plaćaju pretplate najviše igraju žanrove u kojemu sudjeluje veći broj igrača.

Zaključak

Ovaj rad upoznaje nas s računalnim igrama kroz njihovu povijest, žanrove igara te popularnost pojedinih žanrova.

Igre su, kroz povijest, napredovale paralelno s razvojem tehnologije. Razvoj igara bio je brz, igre postaju sve kompleksnije i bolje. Konzola NES prekretnica je u razvoju i definicija žanrova igara kakve danas poznajemo. U radu je obrađeno deset najpopularnijih žanrova. Kroz klasifikaciju žanrova saznali smo glavne razlike između pojedinih žanrova te njihove posebnosti koje doprinose popularnosti pojedinih žanrova.

Koristeći nekoliko zanimljivih istraživanja, prikazali smo popularnost pojedinih žanrova po određenim kategorijama. Tako smo pobliže upoznali koje žanrove preferiraju muškarci, žene, djeca, starija populacija i dr. Zaključujemo da starija populacija bira žanrove koji smanjuju stres i potiču pozitivno raspoloženje, dok djeca biraju žanrove koji razvijaju koncentraciju, strpljenje, logičko

razmišljanje, itd. Žene vole najviše žanrove koji simuliraju stvarni život i logičke igre, dok muškarci preferiraju adrenalinske i natjecateljske žanrove.

Zahvaljujući mnoštvu žanrova igara, pruža nam se mogućnost da odaberemo onaj žanr koji najviše odgovara našim interesima, preferencijama, osobnosti i drugim čimbenicima. Budućnost nam sa sigurnošću nosi mnoštvo novih naslova (igara) koji će utjecati na porast broja korisnika i samim time povećati popularnost pojedinih žanrova.

Reference

[1] Anderson, G. O. (lipanj 2016). *American Association of Retired Persons*. Preuzeto 30. prosinca 2020 iz Video Games: Attitudes and Habits of Adults Age 50-Plus: https://www.aarp.org/research/topics/technology/info-2016/electronic-gaming-research-adults-50plus.html

[2] Clement , J. (29. siječanj 2021). *www,statista.com*. Preuzeto 24. travnja 2021 iz https://www.statista.com/statistics/1188479/gaming-subscription-genre/

[3] Gough, C. (20. Studeni 2020). *www.statista.com*. Preuzeto 8. Siječnja 2021 iz https://www.statista.com/statistics/1188559/gaming-genres-covid/

[4] Horowitz, D. (6. Rujan 2019). *HP Store*. Preuzeto 28. Prosinca 2020 iz https://store.hp.com/us/en/tech-takes/reasons-games-like-fortnite-battle-royale-are-popular

[5] Joan Ganz Cooney Center. (20. Listopad 2016). Preuzeto 3. Siječnja 2021 iz Digital Games and Family Life: The Games Families Play: https://joanganzcooneycenter.org/2016/10/20/digital-games-and-family-life-the-games-families-play/

[6] Kelly, M. (bez dat.). *Game Design & Development 2021*. Preuzeto 24. lipnja 2021 iz Game genres: https://ecampusontario.pressbooks.pub/gamedesigndevelopmenttextbook/chapter/game-genres/

[7] Lowood, H. E. (bez.dat.). *Britannica*. Preuzeto 22. lipnja 2021 iz Electronic game: https://www.britannica.com/topic/electronic-game

[8] Oxford, N. (11. Svibanj 2018). *Lifewire*. Preuzeto 27. Prosinaca 2020 iz Dotdash: https://www.lifewire.com/nintendo-action-game-1126179

[9] Rechsteiner, A. (bez dat.). *Swiss National Museum*. Preuzeto 22. lipnja 2021 iz The history of video games: https://blog.nationalmuseum.ch/en/2020/01/the-history-of-video-games/

[10] Straits research. (11. Travanj 2020). *Straits research*. Preuzeto 2020. Prosinca 28 iz Back to blogs: https://straitsresearch.com/blog/top-10-most-popular-gaming-genres-in-2020/

[11] Technopedia. (bez dat.). Preuzeto 28. Prosinca 2020 iz Role-Playing Game: https://www.techopedia.com/definition/27052/role-playing-game-rpg

[12] Wayward. (25. Rujan 2015). *Wayward*. Preuzeto 29. Prosinca 2020 iz What Is A Real-Time Strategy Game? An Exploration and Definition: https://waywardstrategy.com/2015/09/25/what-is-an-rts-game/
[13] Willingham, A. (27. Kolovoz 2018). *CNN*. Preuzeto 24. Siječnja 2021 iz What is eSports? A look at an explosive billion-dollar industry: https://edition.cnn.com/2018/08/27/us/esports-what-is-video-game-professional-league-madden-trnd/index.html

Izrada akcijske 3D računalne igre

Lino Zadravec, Mario Konecki i Alen Lovrenčić
Fakultet organizacije i informatike, Sveučilište u Zagrebu
lzadravec@foi.hr, mkonecki@foi.hr, alen.lovrencic@foi.unizg.hr

Sažetak

Tema ovog rada je izrada 3D računalne igre. Žanr igre koja će u ovom radu biti opisana je akcijska igra obrane od zombija. U ovoj igri igrač će imati sposobnost, u prvom licu, kontrolirati karakter i rukovati oružjima koja će pomoći u zaustavljanju zombija. Kroz ovaj rad bit će opisan alat Unity, proces konceptualiziranja i razvoja igre te proces implementacije ovakve igre u alatu Unity, koji je jedan od najpoznatijih alata za kreiranje *indie* igara.

Ključne riječi: Akcijska igra, Unity, Razvoj igre, Zombi, Programiranje, C#

Uvod

U modernom vremenu, eksplozijom internetskih te kompjuterskih sposobnosti došlo je do velikog razvitka industrije video igara. Smatra se da u svijetu barem 1,5 milijarda ljudi s pristupom internetu igra video igre. [1] Zbog ovoliko velikog broja igrača može se zaključiti da je i tržišna vrijednost veoma velika. Prema [2] svjetska vrijednost tržišta video igara 2020. bila je 155 mlrd. američkih dolara što je približno 1 bln. hrvatskih kuna.

Pošto video igre donose velike količine novaca, divovi video igara kao što su Sony, Tencent, Nintendo, Activision te ostali, podižu standarde igara do točke gdje novi individualni programeri imaju probleme razvijanja igara do minimalnih očekivanja prosječnog igrača.

Upravo iz ovog razloga individualni programeri odlučuju koristiti alate zvane „*game engine*“ kao što je Unity. Ova vrsta alata omogućuje ubrzani razvoj video igara na način da pruža platformu s već puno implementiranih svojstava (eng. *feature*) koje su korištene u velikoj količini video igara te nam time štedi vrijeme koje bismo morali trošiti na njihov razvoj. [3] Neke od ovakvih svojstava su primjerice: 3D rasvjeta, sjene i fizika koja je krovni pojam za veoma veliku količinu svojstava.

Dizajn igre ovog rada

Sve započinje odabirom žanra igre u našem slučaju to je „akcijska 3D računalna igra“. Od tog trenutka došlo je do odluke da će u ovom radu jezgra igre biti pucanje neprijatelja. Iz ove jezgre postoji mnogo raznih podžanrova koji se mogu ostvariti što bi uključivalo: *Shoot 'em up*, *Run-and-gun shooter*, *Shooting gallery*, *Light gun shooter*, *First-person shooter* i *Third-person shooter* koji nisu prevedeni zbog gubitka u prijevodu. [4]

U našem slučaju igra će biti *First-person shooter*, tj. pucanje iz prvog lica ili ubuduće FPS. Točnije radit će se o FPS igri beskrajne obrane od zombija koja je pronašla inspiraciju u igri „Call of Duty: Black Ops – Zombies“.

Motivacija igre bazira će se na demonstraciji sposobnosti koja će igrača motivirati da pokušava napredovati i doći dalje nego što je uspio u svojem prošlom pokušaju.

Igrač će imati smislene izbore u obliku izbora lokacije na kojoj će se braniti od zombija te koja će oružja birati za obranu. Ova oružja razlikovat će se po svojim svojstvima gdje će svako oružje u jednom pogledu biti smislen izbor barem u jednom dijelu igre.

Kako bismo pratili ravnotežu uživljenosti igrača u igru, beskrajni zombiji dolazit će u valovima između kojih će biti mala vremenska pauza. Prilikom ove pauze igrač će moći odmoriti od intenziteta obrane te promijeniti svoju lokaciju i oružje ukoliko je to potrebno.

Upotrebljivost, koja obuhvaća cilj igre i kako doći do njega, u ovoj igri jest preživjeti što dulje prilikom napada zombija koji svakim valom postaju sve jači i višebrojniji. Ovo se radi na način da se zombiji pucaju oružjem koje je igraču pri ruci. Prilikom pucanja zombija igrač dobiva novac koji može zamijeniti za novo i bolje oružje. Kako bi igrač došao do boljeg oružja, mora kupiti i novi prostor. Ukoliko igrač odluči ne kupiti novi prostor, zaglavio je s oružjem koje neće biti dovoljno dobro za ubijanje jačih zombija.

Estetika ove igre velikim dijelom bit će *outsourceana*, tj. fontovi, zvukovi, 3D modeli igre i mapa kupljeni su ili pronađeni kao besplatni za korištenje na internetu. Ovo je veoma česta praksa u indie igrama zato što programer obično nije dovoljno sposoban ili talentiran za kreirati nešto estetski lijepo u kratkom vremenu. Estetska tema koja će se provlačiti kroz igru jest tamna i strašna atmosfera, a proizlazi iz cilja igre - pucanje zombija.

Mehanike igre

Igrač ima mogućnost kretanja po mapi tipkama „W“, „A“, „S“, „D“ koje su standard FPS igara. Tipkom „Shift“ kretanje se ubrzava za određeni koeficijent pri čemu igrač prelazi u stanje trčanja. Razmaknicom igrač može skočiti kako bi preskočio određene zapreke ukoliko ih ima.

Lijevim klikom na mišu igrač može pucati oružjem koje je trenutno aktivno. Pri jednoj od pušaka igrač može desnim klikom zumirati kako bi lakše naciljao udaljene zombije. Zumiranje je implementirano samo kod puške *Enfield* jer je ona namijenjena za dalekometno streljaštvo. Pri pucanju igrač pomoću nišana na zaslonu cilja protivnike. Razlikuje se hitac u glavu i hitac u ostatak tijela, gdje hitac u glavu radi veću štetu te donosi više novaca.

Igrač može imati maksimalno dva oružja: primarno i sekundarno. Sekundarno oružje je pištolj koji je dostupan igraču kroz cijelu igru i koji ima beskonačnu količinu metaka. Loša strana pištolja je da on donosi malu količinu novaca i nanosi veoma malu štetu zombijima, zbog čega bi se trebao koristiti samo ako igraču ponestane metaka u primarnom oružju. Primarno oružje ima konačnu količinu metaka, no radi puno više štete od pištolja. Igrač ne započinje s nijednim primarnim oružjem, a kako bi ga dobio mora ga kupiti pomoću novca. Izgled početka igre, gdje igrač počinje sa sekundarnim oružjem (pištoljom) možemo vidjeti na slici 1.

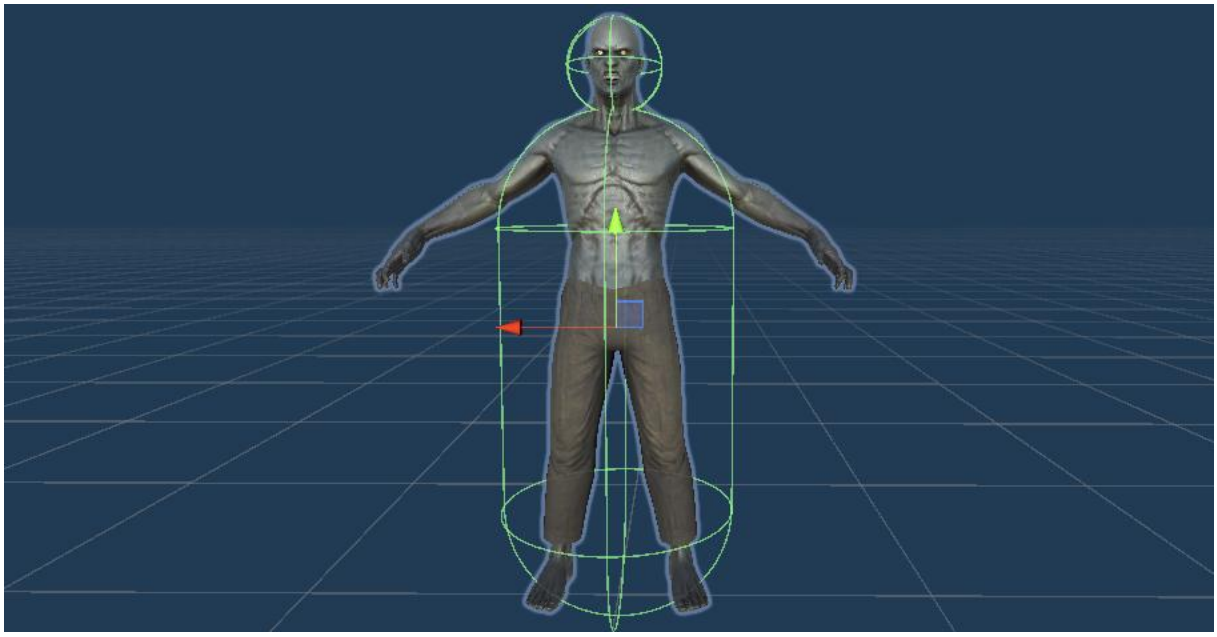


Slika br. 1: Prikaz igrača na početku igre

Zombiji na beskonačnu udaljenost lociraju i love igrača. Kada dođu na dovoljnu blizinu, udaraju ga te nanose štetu na životne bodove igrača. Igrač započinje sa 100 životnih bodova. Ukoliko igrač ima manje od maksimalne količine životnih bodova, a više od nula, nakon što prođe pet sekundi životni

bodovi se počinju vraćati tako dugo dok ne dođu ponovno do maksimalne količine.

Zombiji dolaze u valovima što znači da se svake 42 sekunde instancira novih N zombija, gdje je količina određena ovisno o trenutnom broju vala. Svakim valom povećavaju se životni bodovi zombija, a samo pri određenim valovima povećava se količina i šteta koju zombiji nanose. Na sljedećoj slici 2. možemo vidjeti kako izgleda model zombija i njegove *Collidere*.



Slika br. 2: Model zombija

Kako bi igrač imao šanse protiv sve jačih i jačih zombija, pri pogodcima zombija dobiva novac koji može zamijeniti za bolja oružja. Prodavaonice oružja nalaze se na određenim statičnim mjestima na mapi i svaka prodavaonica prodaje samo jedno oružje. Kako bi ga igrač kupio, mora doći blizu prodavaonice i kliknuti na tipku „F“.

Osim oružja, igrač može svoj novac uložiti i u otvaranje novih vrata. Kada se uloži novac u vrata, pritiskom na tipku „F“ ona se otvaraju i otkrivaju novi dio mape koji prije nije bio dostupan. Na novim dijelovima mape nalazi se i novo, bolje oružje koje će igraču pomoći dulje preživjeti.

Igra završava samo ako igraču životni bodovi padnu na nulu ili manje, a u suprotnom igra se može nastaviti beskonačno s beskonačnom količinom valova zombija. Kada igrač umre, prikazuje mu se izbornik u kojem odlučuje želi li pokušati ponovno ili ugasiti igru.

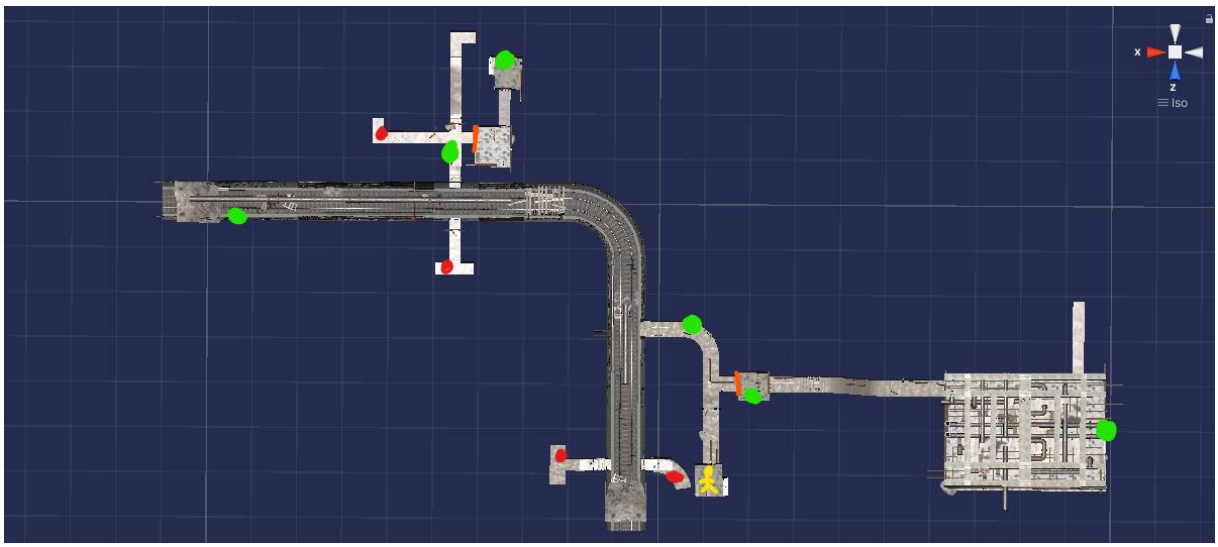
Na slici 3. možemo vidjeti izgled prodavaonicu oružja M4A1.



Slika br. 3: Prodavaonica oružja u igri

Izgled mape

Kao što je prije navedeno, mapa ove igre je *outsourceana* kako igra ne bi gubila na svojoj estetici. Ova mapa je stigla s demo scenom koja je već imala raspoređene objekte u sceni, no mapa je bila prevelika za ciljani žanr ove igre. Upravo zbog toga, skupine objekata igre bile su premještene na druge lokacije na mapi. Točnije cijele sobe sa svim njihovim objektima igre bile su zajedno pomaknute, čime je bio postignut novi tlocrt koji se može vidjeti na sljedećoj slici 4.



Slika br. 4: Tlocrt mape

Prilikom snimanja zaslona ove slike korišten je izometrički pogled na mapu koji je prigodan za slike tlocrta. Na ovom tlocrtu možemo vidjeti točke i crte različitih boja čija će značenja biti objašnjena. Žuti čovječuljak je mjesto na

kojem igrač započinje svoju igru. Svijetlozelene točke na tlocrtu označavaju mjesta na kojima igrač može kupiti novo oružje. Crvene točke označavaju mjesta na kojima se pojavljuju zombiji, a narančaste crte su vrata koja se kupnjom otvaraju i proširuju prostor igre.

Zaključak

Unity je veoma moćan alat za izradu širokog spektra žanrova igara. Ističe se svojom sposobnošću crossplatforminga pošto mu je struktura identična za više podržanih platformi što omogućava da projekt bude izgrađen na drugu platformu i to ponekad bez ikakvih promjena. Ovu sposobnost možemo vidjeti na igri Hearthstone koja je igriva na računalima, macintoshima i mobitelima.

Pri razvoju igre bitno je započeti dizajnom u koji se mora utrošiti dovoljna količina vremena kako bi se postavili temelji igre. Nakon što su temelji igre kao jezgra igre osmišljeni, potrebno je i protumačiti kako će izgledati iskustvo igrača u igri. Bez dizajna igre, pri razvoju bi moglo doći do kontradikcija u funkcionalnostima što bi kreatoru stvorilo poteškoće zbog kojih bi se razvoj igre morao vratiti u fazu dizajna.

Bez programa kao što je Unity, indie programeri ne bi imali šanse u borbi na tržištu protiv divova kao što su Sony, Tencent, Nintendo i Activision. Skup alata koji omogućava izvanrednu brzinu razvoja igara za individualne programere koji u prošlosti nisu bili mogući, dozvoljavaju da bilo tko ima šansu svoju ideju pretvoriti u pravu igru unutar razumne količine vremena.

Reference

[1] R. Chikhani (2015). The History Of Gaming: An Evolving Community. Dostupno na: https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/ . [Pristupljeno 02.08.2021.].
[2] J. Clement (2021). Global video game market value from 2020. to 2025. Dostupno na: https://www.statista.com/statistics/292056/video-game-market-value-worldwide/ . [Pristupljeno 02.08.2021.].
[3] A. Shiddiq (2017.) Why does a game need a game engine? Dostupno na: https://www.quora.com/Why-does-a-game-need-a-game-engine/answer/Andy-Shiddiq . [Pristupljeno 02.08.2021.].
[4] Nepoznat Autor (2021.) Shooter game. Dostupno na: https://en.wikipedia.org/wiki/Shooter_game#Subgenres . [Pristupljeno 16.08.2021.]

Dizajniranje računalnih igara

Tomislav Jukica i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
tomislav.jukica07@gmail.com, mlkoneck@foi.hr

Sažetak

U ovom radu će se govoriti o procesima razvijanja i dizajniranja računalnih igara. Odgovoriti će se na neka temeljna pitanja kao što je to dizajn i koji je cilj dizajnera. Objasniti će se što je to računalna igra i od čega se sve sastoji. Zatim će se govoriti o samom igraču, njegovoj ulozi, te o svijetu koji dizajner videoigre stvara.

Ključne riječi: dizajn, računalna igra, mehanike igranja

Uvod

U ovom radu govorit će se o problematici dizajniranja igara (engl. game design). Naglasak će biti dan na dizajniranje računalnih igara. Kao polazišna točka korištena je jedna od najpopularnijih knjiga u domeni dizajna računalnih igara a to je knjiga autora Jesse Schella naslova The Art of Game Design[3]. Ova knjige je knjiga koja je dobila brojne nagrade za svoju izvrsnost a namijenjena je početnicima koji ulaze u svijet dizajna igara. Uz ovu knjigu, korišteni su i intervjui s raznim drugim poznatim dizajnerima, podcasti o računalnim igrama i članci o različitim igrama. Također, dio rada se temelji i na vlastitom iskustvu igranja računalnih igara od preko 20 godina.

Što je to dizajn?

Dizajn je plan ili specifikacija za izgradnju objekta ili sustava; za provedbu aktivnosti ili procesa; ili je rezultat tog plana ili specifikacije u obliku prototipa, proizvoda ili procesa [1]. Prema N. Stefyn dizajn igre je korištenje kreativnosti i dizajna za razvijanje igara za obrazovne i zabavne svrhe [2]. Za J. Schella su stvari puno jednostavnije te po njemu je dizajn igara jednostavno **donošenje odluka** [3].

Tko može biti dizajner?

Dizajner Troy Dunniway kaže da je posao dizajnera multidisciplinaran posao koji zahtjeva razumijevanje svega pomalo [2]. Kao što arhitekt mora razumjeti i biti u kontaktu sa zidarima, vodoinstalaterima i električarima, te koordinirati

radove sa svima, tako je i game dizajner na neki način veza između pisaca, vizualnih umjetnika, programera i drugih. Postoje mnoge vještine koje vam mogu pomoći u dizajnu igara (animacija, kinematografija, ekonomija, povijest, psihologija itd.), no nemoguće ih je znati sve. Ipak, postoji jedna vještina koja je najvažnija, a to je **slušanje**. Kao dizajneri moramo slušati pet grupa ljudi, a to su:

- tim koji radi na igri,
- publiku,
- igra koje se izrađuje,
- klijenta, i
- sebe.

Cilj dizajnera

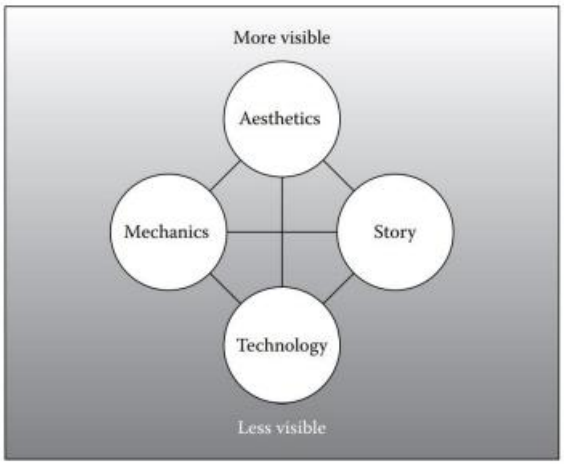
Postoji jednostavna rečenica kojom se može opisati srž posla dizajnera, a to je da **dizajner stvara iskustva**. Igra koju on izrađuje je samo alat kojim postiže željeno iskustvo. No, problem je što **dizajn igara nije znanost** te ne postoji skup pravila koji treba pratiti kako bi dizajnirali željeno iskustvo. Razlog tome je to što su video igre **umjetnost**. Stoga, kako onda napraviti dobru video igru?

Razumijevanje iskustva

Iskustvo je proces kroz koji svjesni organizmi percipiraju svijet oko sebe. Iskustva mogu biti popraćena aktivnom sviješću osobe koja ima to iskustvo, iako to ne mora biti [4]. Kao dizajneri igara potrebno je **odvojiti iskustvo od igre**. Također, samo iskustvo ne ovisi samo o igri koju igramo, već može ovisiti o raznim faktorima od kojih je bitniji **mjesto igranja**.

Igra

Svaka igra izrađena je od četiri osnovna elementa.



Slika br. 1: Osnovni elementi svake igre

1. **Mehanike** – Mehanike igre se odnose na pravila i ciljeve igre, te načine kako se taj cilj postiže. Mehanike u jednostavnoj igri poput Super Marija bi bile skakanje, kretanje, skupljanje pojačanja i novčića.
2. **Priča** – Neke igre stavljaju jači, a neke manji naglasak na priču, no svaka igra je ima. Može biti linearna, ili kao što većina današnjih RPG-ova reklamira, priča može ovisiti o vašim akcijama unutar igre.
3. **Estetika** – Kako vaša igra izgleda, zvuči i kako je igrači percipiraju. Ovo je najvidljiviji element igre.
4. **Tehnologija** – Tehnologija pomoću koje je vaša igra realizirana.

Tema igre

Tema je **centralna** stvar u dizajnu igre. Sve što je u skladu i podupire temu ostaje u igri, a stvari koje se kose s njom je potrebno korigirati. No, prije same teme, postoji **ideja**. Dizajner Chris Klug kaže da bi svaki dizajner trebao pronaći jedno **ključno emocionalno iskustvo** oko kojeg treba pokušati izgradi igru [3]. Evo što je grupa studenata naučila kada je u jednom semestru izradila preko 50 igara [5]:

- Za svaku ideju se može izraditi prototip u manje od sedam dana
- Ograničenja naglašavaju kreativnost
- Skupljanje konceptualnih skica i glazbenih isječaka da bi stvorili emocionalno iskustvo

Iteracije

Da bi igru poboljšali nužno je korištenje **iterativnog razvoja**. Jedan od najpopularnijih razvojnih modela jest **Scrum**. Schell predlaže sljedeću vrstu iteracije [3]:

1. Izražavanje problema.
2. Brainstorming sesija potencijalnih rješenja.
3. Odabir rješenja.
4. Zapis rizika s kojima bi se mogli sresti za odabrano rješenje.
5. Izrada prototipa radi umanjivanja rizika.
6. Testiranje prototipa. Ako su dovoljno dobri, iteracija je završena.
7. Ponovno izraziti nove probleme koji su se pojavili: zatim je potrebno vratiti se na korak 2.

Mehanike igre

Mehanike su jedna od četiri glavna elementa svake igre. Jedna od glavnih mehanika jest **prostor** u kojem se igra odvija. On je najčešće 2D ili 3D. Također, igre mogu sadržavati nekoliko slojeva prostora, pa je često u RPG-

ovima moguće vidjeti ovu vrstu apstrakcije gdje se putovanje između gradova odvija u 2D prostoru (često prikazano u obliku mape) dok su sami gradovi prikazani u 3D prostoru.

Sljedeća mehanika jest **vrijeme**. Ono može biti **diskretno** (igre na poteze) i **stalno**. Također, igre mogu sadržavati i kombinaciju, gdje je dio igre u stalnom vremenu (istraživanje), a dio se igra u diskretnom (bitke).

Objekti, atributi i **stanja** su također dio mehanika. Objekti predstavljaju stvari koje se nalaze u prostoru, dok se atributima opisuju objekti. Osim atributa, objekti mogu imati i stanja koja se tijekom igre mijenjaju.

Akcije se odnose na stvari koje igrač može uraditi. Skakanje, pucanje ili bacanje karte su sve primjeri akcija i odnose se na **osnovne akcije**. Druga razina akcija su **strateške akcije**. One se odnose na više ciljeve kao što je „ubiti prvo najopasnijeg neprijatelja“. Često kada se govori o igrama, spominje se riječ **meta**. Meta označava trenutno najjaču strategiju za tu igru. Strateške akcije nastaju međusobnom interakcijom osnovnih akcija [3]. Na primjer, recimo da su osnovne akcije igre postavljanje mine i pucanje. Mina funkcionira na način da kada neprijatelj stane na nju ona eksplodira. No, recimo da su neprijatelji lukavi i često znaju izbjegavati mine. Ukoliko je moguće da pucanjem u minu ona eksplodira, igraču je dano „novo oružje“, jer sada on može aktivirati minu pucanjem u nju kada je neprijatelj u blizini. Igrač bi trebao imati dobar balans kratkoročnih (izraditi obranu baze) i dugoročnih (uništiti neprijateljsku bazu) ciljeva.

Svaka igra koju igrač igra zahtjeva od njega da pokaže određenu vrstu vještine. Pa tako imamo **fizičke** (motoričke sposobnosti, koordinacija), **mentalne** (donošenje odluka) i **socijalne** (čitanje neprijatelja, suradnja s drugim igračima).

Danas rijetko vidimo igru u kojoj ne postoji element **šanse/vjerojatnosti**. Za Schell-a je to **tajni sastojak zabave**. No, treba paziti kako se ona ne bi zloupotrijebila. Ako je ima previše, igrač će misliti da njegov uspjeh ne ovisi o njegovoj vještini, već o šansi, a ako šansu izbacimo, tada imamo stanje u kojem postoji **optimalno rješenje** danog problema, te smo od igre napravili **zagonetku**.

Balans

Za dobru igru nije dovoljno samo dizajnirati dobre mehanike, već je također i potrebno osigurati da su te mehanike u **balansu**. Prva vrsta balansa je **poštenje**. Najosnovniji oblik balansa su **simetrične** igre, gdje svi igrači kreću sa istim resursima i pozicijama. Ovo je najčešći oblik balansa kod **igara za više**

igrača. No, **igre za jednog igrača** su često **asimetrične**, često da bi prikazale realnost situacije (u stvarnom životu gotovo ništa nije simetrično).

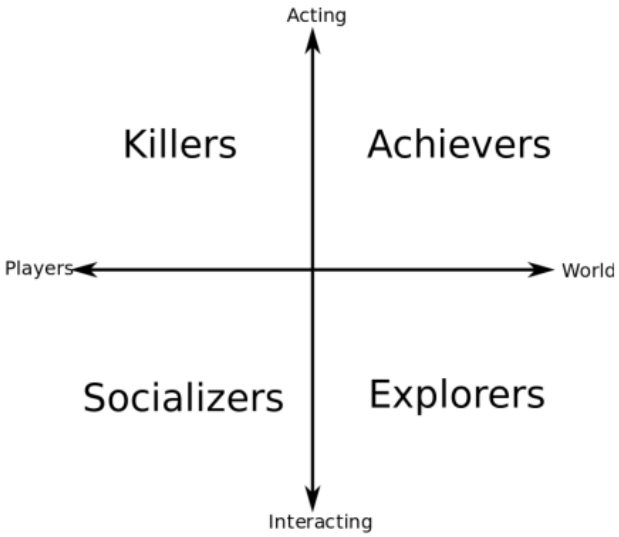
Jedna od najbitnijih stvari koju je potrebno balansirati jest **tok** naše igre. Ukoliko je igra preteška, igraču će to stvarati stres, a ako je prelagana igraču će brzo dosaditi. Zbog toga su se neke igre odlučile koristiti **dinamičku težinu** gdje bi se sama težina igre prilagođavala vještini igrača.

Igrač

Igrač je jedna od najbitnijih komponenata dizajna igre. Igrače se može podijeliti u puno kategorija, no najvažnije su **dob** i **spol**. Svaka od tih kategorija na svoj način konzumira videoigre. Pa tako igrači muškog spola češće igraju igre gdje se može vidjeti: **savladavanje prepreka, natjecanje, uništavanje** te **prostorne zagonetke**. Kod igrača ženskog spola su bitnije sljedeće stavke: **emocije, stvarni svijet, empatija** te **dijalog i verbalne zagonetke**.

Osim prema dobi i spolu, Richard Bartle je 1996. godine igrače klasificirao u četiri skupine: [6]

- **Achievers** – žele postići sve ciljeve u igri. Njihovo primarno zadovoljstvo su izazovi.
- **Explorers** – žele maksimalno upoznati svijet igre. Njihovo primarno zadovoljstvo je otkrivanje.
- **Socializers** – oni su primarno zainteresirani za stvaranje odnosa s drugim ljudima/igračima.
- **Killers** – oni su najviše zainteresirani za natjecanje sa drugim igračima.



Slika br. 2: Bartleova taksonomija tipova igrača

Motivacija

Psiholog Abraham Maslow je 1943. godine napisao rad u kojem je predložio hijerarhiju ljudskih potreba, poznatu kao **Maslowljeva hijerarhija potreba**. Često se prikazuje kao piramida. Ideja je da ljudi neće tražiti veću razinu potreba dok im nije zadovoljena niža razina. Osim fizičkih potreba, Edward Deci i Richard Ryan govore da ljudi imaju i mentalne potrebe [3]. Prvenstveno, to su:

1. Kompetentnost
2. Autonomija
3. Povezanost

Premda djeluje jednostavno, postoji puno dokaza u točnost ovih potreba. Videoigre tako ispunjavaju sve tri potrebe. Igre od nas često zahtijevaju da demonstriramo kompetentnost da bi prešli razinu. Često nam dopuštaju da zadatke obavimo na različite načine. Oko mnogih igara postoji socijalna komponenta, čak i oko igara koje su prvenstveno iskustvo za jednog igrača se znaju formirati zajednice igrača koji raspravljaju o najboljim strategijama.

Drugi način na koji je moguće gledati na motivaciju jest u pogledu **unutarnje** i **vanjske** motivacije. Akor adi sebe nešto želimo napraviti tada smo intrinzično motivirani. Ako postoji neki vanjski razlog (npr. netko će nam platiti) tada to predstavlja vanjsku motivaciju [3]. U videoigrama možemo gledati na sljedeći način: radim li nešto u igri zato što želim ili zato što će me igra nagraditi? Tu dolazimo do pojma **grinding** koji označava obavljanje repetitivnog posla unutar videoigre [7]. Taj posao se obavlja da bi igrač dobio nekakvu nagradu od igre što će mu olakšati daljnji napredak u igri. Iako se smatra da je grinding rezultat lošeg game dizajna [8], danas je to gotovo neizostavan dio većine igara. Jedan razlog je taj što developeri igru od 20 sati mogu pretvoriti u igru od 50 sati ako dodaju grinding. Trend je takav da developeri uvijek moraju napraviti veću igru od prethodne, pa tako broj sati potrebnih za završiti igru konstantno raste kroz godine.



Slika br. 3: Prosječno vrijeme završavanja igre kroz godine

Svijet

Pod svijetom ne govori se samo o fizičkom prostor već se misli i na priču svijeta, likove koji borave u svijetu i sl. Pojava videoigara je promijenila način na koji pisci pišu **priče**. Razlog tomu je što, za razliku od tradicionalnih medija poput knjiga i televizije, videoigre **nisu linearne**. Najjednostavnija metoda za pisanje priče u igrama se naziva **niz bisera**. Forma priče u takvim igrama obično izgleda tako da postoji linearni dio (obično u obliku video isječka) nakon kojeg slijedi dio gdje igrač može donositi odluke (mora na neki način prijeći razinu).

Nadalje, postoje tzv. **generatori priče** (*Rimworld*) gdje se priča stvara nizom *„nasumičnih"* događaja. Svaka dobra priča mora imati glavnog lika koji ima određeni cilj i prepreke koje ga sprječavaju da postigne taj cilj. Filozofija iza ovoga je jednostavna: k**onflikt stvara zanimljive situacije**.

Likovi

Nakon što je stvoren svijet igre, potrebno je definirati i likove koji žive u tom svijetu. Idealno, likovi bi trebali biti zanimljivi i lako pamtljivi. Kako bi to postigli potrebno je pratiti nekoliko korisnih uputa:

- Napravite listu funkcija likova (heroj, negativac, mentor)
- Definirajte osobine svakog lika
- Napravite mrežu odnosa likova
- Definirajte status svakog lika (npr. kralj se ponaša različito od običnog seljaka)

No, ono što je najbitnije zapamtiti jest da **jake priče mijenjaju likove**. Ukoliko ste napisali dobru priču vaši likovi na početku i na kraju ne bi trebali biti isti.

Estetika

Zadnji temeljni element svake videoigre je i estetika. Estetika je grana filozofije koja se bavi prirodom ljepote i ukusa [9]. Pod estetikom u igri smatra se njen vizualni izgled ali i zvukovi koji se nalaze u igri. Na ovu stavku je bitno misliti rano u razvoju i dok se još uvijek razvijaju mehanike igre uključiti umjetnike da rade konceptualne skice. Da bi bolje pojmili što se smatra pod estetikom, pogledajmo za primjer igru **Darkest Dungeon**. Na slici 4. se nalazi tipična scena brobe iz igre.



Slika br. 4: Scena borbe iz igre Darkest Dungeon

Kao što je vidljivo, igra ima poprilično karakterističan vizualni identitet koji je u skladu s njezinom **temom**. Naime, poprilično mračna atmosfera odgovara igri u kojoj su jedne od glavnih mehanika neuroze prouzročene visokim stresom i konstantno umiranje likova. Što se tiče zvuka i on je u jednakom skladu sa temom. Igra koristi naratora kao **jedini glas** koji nam opisuje što se događa dok se istražuje svijet igre.

Glazbena podloga (engl. soundtrack) je također jako bitna stvar estetike igre. Primjerice, u serijalu videoigara Heroes of Might and Magic, skladatelj Paul Anthony Romero je napravio posebnu glazbu za svaku od frakcija unutar igre. Tako frakcija vilenjaka, patuljaka i jednoroga ima glazbenu podlogu gdje su korišteni instrumenti poput harfi, violina i drvenih puhačih instrumenata dok frakcija živih mrtvaca sadrži glazbene motive koji su daleko napetiji, s otkucajima sata u pozadini te glazbenim artikulacijama koje je moguće čuti u filmovima strave.

Jednako tako su važni i zvučni efekti u videoigri. Na primjer, videoigra DOOM ETERNAL kombinira elemente industrijskog i teškog metala skladatelja Micka Gordona s raznim zvučnim efektima oružja, eksplozija i motornih pila što je fantastična kombinacija za visoko adrenalinsku pucačinu iz prvog lica.

Zaključak

U ovom radu dan je pregled osnovnih elemenata koje dizajneri videoigara moraju imati na umu. Dizajn videoigara je vrlo opširno područje koje je gotovo nemoguće u potpunosti savladati. Zbog svoje široke multidisciplinarnosti, dizajneri videoigara mogu biti ljudi različitih profesija, te tehnička pozadina nije nikakav uvjet. Također, ovo je jedno od onih područja gdje se primarno

kroz iskustvo i kontinuiranim radom može napredovati u spoznaji i kvaliteti kako dizajnirate videoigre koje će biti atraktivne i aktualne.

Reference

[1] »Design,« Wikipedia, the Free Encyclopedia, [Mrežno]. Available: https://en.wikipedia.org/wiki/Design. [Pokušaj pristupa 26. 08. 2021.].

[2] N. Stefyn, »What is game design and how to become a game designer,« 2019. [Mrežno]. Available: https://www.cgspectrum.com/blog/what-is-game-design. [Pokušaj pristupa 26. 08. 2021.].

[3] J. Schell, The Art of Game Design: A Book of Lenses, Pittsburgh: CRC Press, 2015.

[4] »Experience,« Wikipedia, the Free Encyclopedia, [Mrežno]. Available: https://en.wikipedia.org/wiki/Experience. [Pokušaj pristupa 30. 08. 2021.].

[5] K. Gabler, K. Gray, M. Kucic i S. Shodhan, »How to Prototype a Game in 7 Days,« 2005. [Mrežno]. Available: http://miami.lgrace.com/documents/How%20to%20Prototype%20a%20Game%20in% 20Under%207%20Days.pdf. [Pokušaj pristupa 01. 09. 2021.].

[6] »Bartle taxonomy of player types,« Wikipedia, the Free Encyclopedia, [Mrežno]. Available: https://en.wikipedia.org/wiki/Bartle_taxonomy_of_player_types. [Pokušaj pristupa 08. 09. 2021.].

[7] D. Paez, »How "Grinding" Came to Dominate Video Game Culture,« 2020. [Mrežno]. Available: https://www.inverse.com/gaming/how-grinding-came-to-dominate-videogame-culture. [Pokušaj pristupa 08. 09. 2021.].

[8] B. Lindbergh, »Length Matters,« 2016. [Mrežno]. Available: https://www.theringer.com/2016/8/25/16038806/video-game-length-playtimesf7b8e38f949f. [Pokušaj pristupa 08. 09. 2021.]

[9] »Aesthetics,« Wikipedia, the Free Encyclopedia, [Mrežno]. Available: https://en.wikipedia.org/wiki/Aesthetics. [Pokušaj pristupa 09. 09. 2021.]

Izrada igre preživljavanja u programskom alatu Unity

Dominik Štefičar i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
dsteficar@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu opisat će se implementacija osnovnih mehanika razvoja trodimenzionalne računalne igre preživljavanja u programskom alatu Unity. Opisat će se osnovne mehanike upravljanja igračem, mehanike upravljanja resursima i mehanike izgradnje skloništa.

Ključne riječi: Unity, videoigra preživljavanja, videoigra iz prvog lica, programiranje, algoritmi

Uvod

Razvoj videoigara je dugotrajan i poprilično izazovan zadatak jer zahtjeva multidisciplinarno znanje i mnoge vještine kako bi se uspješno implementirala videoigra. Programski alati kao što je Unity [1] (ili Unreal Engine) uvelike pomažu u razvoju videoigre jer automatiziraju mnoge tehničke aspekte koje nije jednostavno implementirati: iscrtavanje grafike, simulaciju fizike, reprodukciju zvuka i sl. Osim samih alata također postoji i niz materijala iz kojih je moguće naučiti kako implementirati pojedine segmente i mehanike videoigara. S obzirom na sve navedeno, danas i pojedinac može samostalno krenuti u realizaciju izrade videoigre. Izrada prototipa videoigre preživljavanja je glavna tematika ovog stručnog rada.

Korišteni alati i tehnike

Veći dio funkcionalnosti kreiranog prototipa je rađen u programskom alatu Unity [1]. Kao što je već istaknuto, Unity je opremljen s mnoštvom već implementiranih funkcionalnosti koje su potrebne za razvoj videoigre: detekcija kolizije, fizika krutih tijela, uvoz modela, stvaranje izvora svjetlosti i dr. Programski kod je pisan u programskom jeziku C# te je rađen u programskom alatu Microsoft Visual Studio. I Unity i Visual Studio imaju svoje besplatne inačice i upravo su te besplatne inačice korištene za razvoj prototipa videoigre preživljavanja.

Za pripremu 3D modela, kao i njihovo animiranje, korišten je alat Blender [2]. Blender je također besplatan alat koji može stati rame uz rame s komercijalnim alatima za razvoj i animiranje 3D modela. Za pripremu tekstura

za 3D modele korišten je besplatni alat Gimp [3]. Gimp je besplatan alat otvorenog koda za izradu i manipulaciju slike koji pruža veliki raspon besplatni dodataka za olakšano korištenje i postizanje željenih ciljeva. U alati su kreirane razne teksture i ikone unutar videoigre.

Za stvaranje i modificiranje svojstava tekstura korišten je besplatan alat Materialize [4]. Materialize je samostalan besplatan alat otvorenog koda osmišljen za olakšanje procesa stvaranja materijala modela proizvedenih iz tekstura. Učitavaju se odabrane teksture te se koracima izrađuju mape zadane teksture, koje se koriste kako bi se na površini 3D modela kreirale ponavljajuće teksture, uzorci, i specijalni vizualni efekti poput osvjetljenja i odsjaja. Mape se u igrama najčešće koriste za prikaz slike na fizičkoj osnovi kako bi tekstura modela bila realističnija i podražajna na utjecaj svijetlosti [5].

Igrač

Kod izrade videoigre iz prvog lica pozicija igrača se veže izravno uz poziciju kamere kroz koju igrač gleda: kamera treba biti povezana na objekt igrača te pozicionirana na visinu vidnog polja. Na taj način dobivamo efekt igranja iz prvog lica. Određenim svojstvima kamere možemo parametrizirati pogled igrača: svojstvo „clipping planes“ određuje najbližu i najdalju točku iscrtavanja u jedinicama koordinatnog sustava alata Unity dok „occlusion culling“ pomaže optimizaciji videoigre na način da se iscrtavaju samo oni objekti koji su u vidnom polju igrača.

Interakcija između kamere i modela igrača temelji se na pomaku miša u određenom smjeru. Model igrača prati pomake miša: pomak lijevo-desno okreće model igrača po osi X dok pomak gore-dolje rotira pogled po Y osi. Kretanje igrača se izvršava klasično pritiskom tipaka na tipkovnici: tipke W, A, S, D za pomicanje u svim smjerovima i tipka Space za skok. Na model igrača dodana je komponenta „Character Controller“ koja omogućava funkcionalnost kretanja i skakanja: pritiskom na odgovarajuću tipku, registrira se vertiklani, odnosno horizontalni pomak te se na temelju izračunatog vektora u trodimenzionalnom prostoru vrši pomicanje u svim smjerovima.

```
Vector3 forwardMovement = transform.forward * vertInput;
Vector3 rightMovement = transform.right * horizInput;
charController.SimpleMove(Vector3.ClampMagnitude(forwardMovement +
rightMovement, 1.0f) * movementSpeed);
```

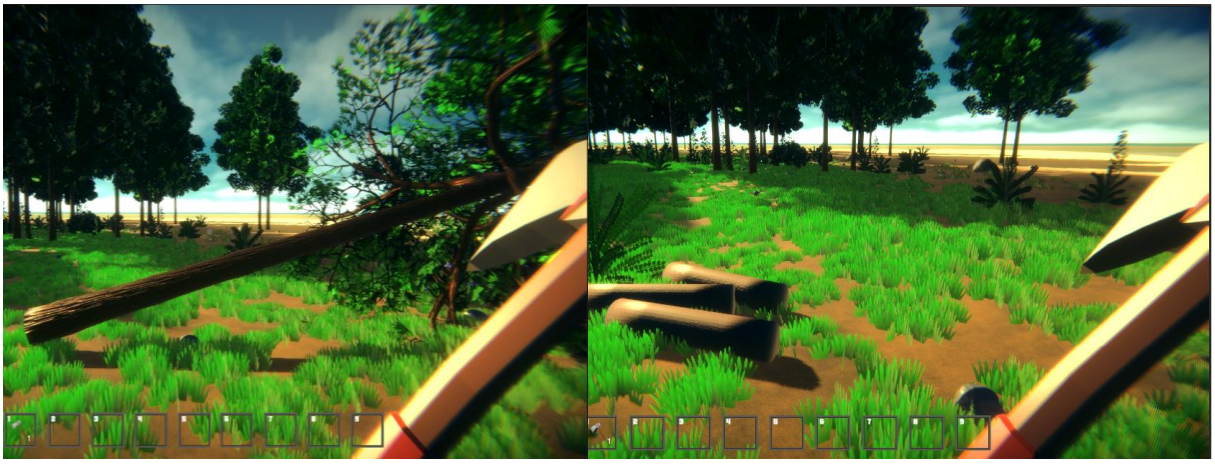
Skakanje se izvršava pritiskom tipke Space pri čemu je prethodno namješten graf putanje objekta kroz fazu skoka.

```
do{
    float jumpForce = jumpFallOff.Evaluate(timeInAir);
    charController.Move(Vector3.up * jumpForce * jumpMultiplier *
                                                    Time.deltaTime);
    timeInAir += Time.deltaTime;
    yield return null;
} while (!charController.isGrounded && charController.collisionFlags
!= CollisionFlags.Above);
```

Okoliš i vegetacija

Još jedan koristan alat u programskom alatu Unity je alat za stvaranje okoliša, tj. terena videoigre. Alatima kao što su kistovi za podizanje/spuštanje terena je moguće oblikovati teren prema željenom izgledu. Također, postoje i kistovi za postavljanje vegetacije stoga je moguće na vrlo intuitivan način kreirati prirodni okoliš koji je često prepun raznih objekata vegetacije.

Sama kreirana vegetacija u ovoj videoigre je interaktivna. Određeni elementi su samo statični vizualni elementi dok s određenim elementima igrač može vršiti interakciju te predstavljaju sastavni dio mehanike videoigre preživljavanja. Igrač mora skupljati odgovarajuće resurse kako bi mogao kreirati određene objekte koji će mu pomoći kako bi prikupio dodatne resurse (npr. kreirati sjekiru kako bi mogao srušiti stablo i prikupiti cjepanice koje kasnije koristi za izgradnju skloništa). Svaki interaktivni objekt na sebi ima svoj sudarač (engl. collider) koji omogućava interakciju na temelju kolizije (npr. kolizija sjekire sa stablom drventa).



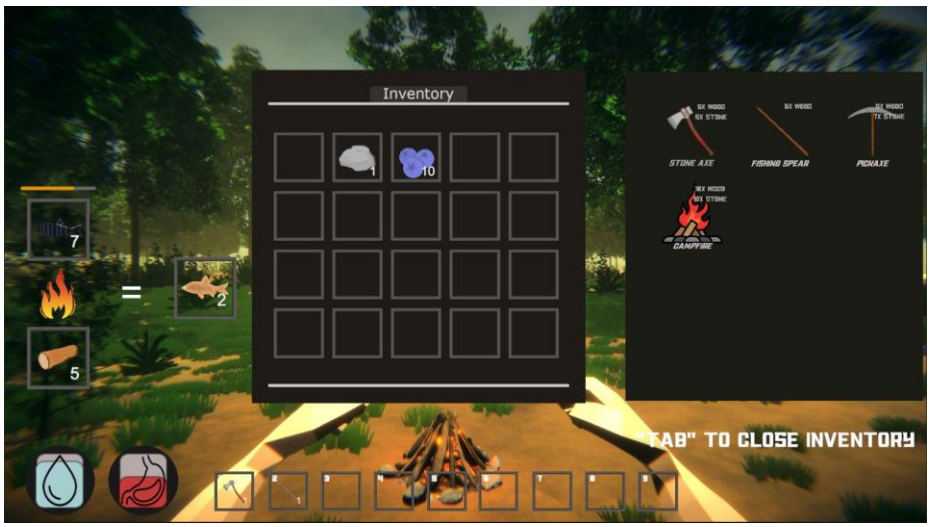
Slika br. 1: Interaktivni okoliš

Upravljanje resursima

Osnovna mehanika kod igara preživljavanja je prikupljanje resursa pomoću kojih igrač izrađuje odgovarajuća sredstava za preživljavanje. Za prikaz korisnih informacija o resursima kreirano je korisničko sučelje u obliku platna

(engl. Canvas) na kojemu se nalaze odgovaraju grafički elementi koji prikazuju količinu i vrstu prikupljenih resursa. Igrač može vidjeti svoj inventar: u inventaru može vidjeti vrstu i količinu prikupljenih resursa, trenutno opremljenu opremu i dr. Tu se nalaze i interaktivne ikone koje omogućavaju izradu odgovarajućeg alata ako igrač ima dovoljno resursa za izradu alata. Također, igrač može vidjeti i ostale elemente mehanika preživljavanja: trenutno stanje gladi i žeđi te popis glavnih ciljeva, tj. zadataka koje treba izvršiti kako bi uspio preživjeti.

Inventar resursa sadrži dvadesetak mjesta u nizu u koje igrač sprema prikupljanje resurse te se prilikom prikupljanja automatski sortiraju na mjesta ovisno o dostupnosti mjesta i maksimalnom kapacitetu. Inventar opreme je uvijek vidljiv na donjem dijelu ekrana te je zamišljeno da igrač ima brzi pristup ovom dijelu kako bi se mogao na jednostavan način opremiti potrebnom opremom za prikupljanje resursa. Lista mogućih alata za izradu vidljiva je na desnom dijelu ekrana u pogledu inventara. Kao što je već napomenuto, ikone su interaktivne te u slučaju da igrač ima dovoljno potrebnih resursa, igrač može izraditi određeni alat.



Slika br. 2: Korisničko sučelje: inventar

Glavni cilj igrača je preživjeti. Životi igrača određeni su količinom hrane i vode koju igrač mora održati na određenoj razini kako ne bi dehidrirao ili umro od gladi. Ako razina hrane ili vode padne ispod minimalne granice, igrač nije uspio preživjeti te će morati igru krenuti igrati od početka. S vremenom ili s odgovarajućom aktivnošću (hodanje, trčanje), igraču se spušta razina hrane i vode u tijelu. Kako bi nadomjestio izgubljeno, igrač može jesti bobice koje može pokupiti na određenih grmovima ili može loviti ribe. Naravno, bobice nadomještavaju mnogo manje status gladi od primjerice ulovljene ribe ali ribu je zato teže uhvatiti. Bobice je jednostavno potrebno uzeti s grma i igrač ih odmah može pojesti. Za hvatanje ribe prvo je potrebno stvoriti alat za hvatanje, nakon toga je ribu potrebno uhvatiti, potrebno je kreirati logorsku

vatru, ribu staviti na logorsku vatru, potrebni su odgovarajući resursi za održavanje vatre te je potrebno odgovarajuće vrijeme kako bi se riba ispekla i bila spremna za konzumiranje.

Umjetna inteligencija riba

Ribe se nalaze u moru u neposrednoj blizini otoka na kojem se igrač nalazi. Kao što smo rekli, igrač prvo mora izraditi koplje kako bi mogao uhvatiti ribu. Ribe se kreću u jatima te imaju kretnje u različitim smjerovima. Za implementaciju mehanike lova riba prvo je potrebno kreirati objekt koji će stvoriti jato riba na određenoj lokaciji. Ribe se zatim prema određenim pravilima ponašanja kreću u moru. Potrebno je odrediti dimenzije objekta unutar kojeg će se ribe stvarati kao i maksimalan broj riba koje se stvaraju u nekom trenutku. Svaka stvorena riba ima svoj model, kostur, animaciju plivanja, kolizijski objekt i skriptu koja je zadužena za ponašanje ribe, tj. umjetnu inteligenciju. Ponašanje ribe se izvodi na sljedeći način: u svakom trenutku provjerava se ima li riba svoj cilj prema kojem se kreće. Ako je riba došla do prethodno zadanog cilja, tada se određuje novi cilj kretanja a on se određuje tako da se odabire nasumična točku u određenoj regiji mora. Smjer i rotacija ribe se definira na temelju trenutne točke na kojoj se nalazi i točke prema kojoj riba ide.

```
float TurnSpeed = currentSpeed * Random.Range(0.7f, 2f);
Vector3 LookAt = waypoint - this.transform.position;
transform.rotation = Quaternion.Slerp(transform.rotation,
        Quaternion.LookRotation(LookAt), TurnSpeed * Time.deltaTime);
```

Izgradnja skloništa

Gradnja skloništa izvodi se korištenjem izbornika za gradnju. Igrač na raspolaganju ima nekoliko vrsta elemenata za izgradnju: temelj, stepenice, zid i krov. Za svaki ovaj element potrebna je određene količina resursa.



Slika br. 3: Izgradnja objekata

Prvo je potrebno izgraditi temelj na koji se postavljaju zidovi na koje se onda postavlja krov. Igrač prvo treba odabrati objekt koji želi izgraditi a zatim odabire poziciju na ekranu gdje želi izgraditi odabrani objekt. Točnu poziciju igrač odabire pomicanjem miša.

Zaključak

Izrada videoigara je zanimljiv, zabavan ali i izazovan zadatak. Dostupni alati i edukacijski materijali uvelike olakšavaju proces izrade videoigre. Uz dovoljno uloženog vremena, svatko može naučiti potrebno znanje i potrebne vještine kako bi izradio svoju vlastiti videoigru. Postoje mnogi besplatni alati koji uvelike olakšavaju proces izrade. S obzirom da su besplatni, oni su zbog toga izuzetno pristupačni te omogućavaju svakome da bez uloženih materijalnih sredstava stekne iskustvo izrade videoigre.

Mnogi koncepti koji su korišteni u razvoju prototipa ove videoigre se mogu iskoristiti i u drugim projektima i drugim videoigrama, čak potpuno drugih žanrova. Svako novo iskustvo nosi novo znanje i vještine i uz dovoljno iskustva, moguće je steći dovoljno znanja kako bi se moglo ući u industriju razvoja videoigara.

Reference

[1] What is Unity? Everything you need to know - Android Authority. Pristupljeno 12. rujna, 2021. https://www.androidauthority.com/what-is-unity-1131558/
[2] About — blender.org. Pristupljeno 10. rujna, 2021. https://www.blender.org/about/
[3] GIMP - GNU Image Manipulation Program. Pristupljeno 12. rujna, 2021. https://www.gimp.org/
[4] Bounding Box Software - Materialize. Pristupljeno 12. rujna, 2021. https://boundingboxsoftware.com/materialize/
[5] What is PBR and What a 3D Artist Should Know - CG Obsession. Pristupljenog 13. rujna, 2021. https://cgobsession.com/what-is-pbr-and-what-a-3d-artist-should-know/

Izrada igre šaha u Unity okruženju

Mario Nižić, Danijel Radošević i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
mnizic@foi.hr, darados@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu prikazana je izrada računalne igre šaha u 2D okruženju korištenjem programskog alata Unity. Na početku rada bit će dane osnovne informacije o igri šah nakon čega će biti govora o izradi programskog rješenja. posebno poglavlje se bavi vizualnim dijelom igre a glavni dio bavi se implementacijom pravila igre: pravila upravljanjem figura i međusobnom interakcijom.

Ključne riječi: Unity, C#, šah, računalna igra, 2D, programiranje

Uvod

Vjerojatno svatko tko je ikada igrao videoigre je barem jednom u životu pomislio na ideju samostalnog razvoja i implementacije vlastite igre koja bi bila plod vlastite mašte i koja bi bila uređena u potpunosti prema vašim željama. Realnost je da kod velike većine ljudi to ostane na razini ideje u glavi zato što je proces razvoja računalnih igara relativno kompleksan i zahtjeva široko znanje iz različitih područja: primarno iz područja programiranja, matematike ali i iz umjetničkih područja kao što je grafički dizajn, 3D modeliranje i sl. Također, tu je i područje dizajna same videoigre, definiranje pravila, balansa, priče, likova, konteksta i sl.

Danas je ipak mnogo lakše pristupiti problematici implementacije vlastite računalne igre zato što postoje brojni alati koji nam u tome mogu pomoći a jedan od njih je i programski alat Unity [1]. Alat automatirizra i rješava mnoge probleme „niže razine“ što ostavlja kreatoru onda više vremena da se bavi suštinskim stvarima oko implementacije suštinskih mehanika i pravila videoigre. Razina znanja koja je potrebna više ne mora biti na visokoj i naprednoj razini: dovoljno je poznavati osnove programiranja i kreator može krenuti u proces izrade svoje videoigre te eventualne nedostatke u znanju pohvatati u procesu izrade. Uvelike pomaže i to da na Internetu imamo kvalitetnu dokumentacije korištenja alata ali i mnoga rješenja za probleme s kojima su se drugi već susreli.

U ovom radu opisat će se popularna igra šah te proces implementacije ove igre u programskom alatu Unity. Igra je s jedne strane relativni jednostavna: imamo figure koje se pomiču po ploči. No postoji i niz nekih specifičnih situacija koje zahtijevaju posebnu pažnju.

Šah

Šah je jedna od najstarijih i najraširenijih igara na ploči. Ime je dobila po perzijskoj riječi šah što znači vladar, kralj ili car. Igra se na šahovskoj ploči odnosno šahovnici koja se sastoji od osam redova numeriranih od 1 do 8 te osam stupaca označeni slovima od a do h tako da svako od 64 polja ima jedinstvenu oznaku stupca i retka (npr. e4 – peti stupac i četvrti redak). Na početku partije se na jednoj strani nalazi 16 bijelih, a na drugoj strani 16 crnih figura. Na svakoj strani imamo po jednog kralja, jednu damu, dva topa, dva lovca, dva konja i osam pješaka [2].

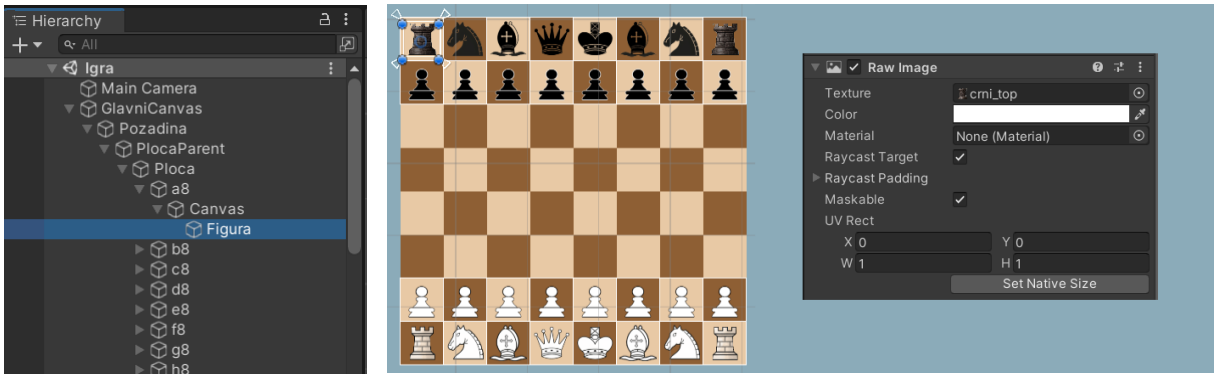
Kralj je najvažnija i najdragocjenija figura u šahu. Cilj igre je zarobiti protivničkog kralja tako da je protivniku nemoguće pobjeći ili obraniti se. Ako kralja napada neka figura odnosno prijeti uzimanjem, tada se kaže da je kralj u šahu i igrač tu prijetnju mora otkloniti idućim potezom. Ako je to neizvedivo, tada je kralj matiran i igra je završena [3].

Postoje tri načina obrane od mata. Prvi je da se napadačka figura eliminira, drugi je da se postavi vlastita figura na liniju napada tako da je pritom napadnuta ta figura umjesto kralja i treći način je uzmicanjem kralja na polje koje nije pod napadom [4].

Ploča

Igraća ploča kreirana je tako da je napravljena prazna „slika“ pod nazivom „PlocaParent“ u obliku kvadrata veličine zamišljene ploče, međutim funkcionalno gledajući, važniju ulogu sadrži podelement „Ploca“ koji posjeduje komponentnu „Auto Grid Layout Group“ [5]. Pomoću te javno dostupne skripte za Unity UI, možemo skalirati ploču na 8 redova i 8 stupaca te automatski dobiti adekvatnu veličinu polja u odnosu na element roditelja.

Zatim elementu „Ploca“ dodajemo 64 elementa koji predstavljaju polja, a nazvani su prema pozicijama šahovske notacije. Slova predstavljaju stupac, a brojevi redove. Primjerice, kod polja „a8“, „a“ označava prvi stupac slijeva, a „8“ posljednji redak odozdo odnosno najgornji redak. „Auto Grid Layout Group“ tih 64 polja grupira u mrežu veličine 8x8. Kako bi se postigla tema drvene ploče, za crna polja korištena je nijansa smeđe boje (#8E5F34), a za bijela nijansa bež (engl. beige) to jest blijedo pješčane boje (#E8C9A5). Bitno je i napomenuti da na početku igre 32 polja sadrže i figure.



Slika br. 1: Hijerarhija elementa „PlocaParent“ i igrača ploča

Pojedincima kojima je šahovska notacija strana će biti jasnije ukoliko uvide da figura na polju „a8“ predstavlja početnu poziciju crnog topa s damine strane. Također, možete uočiti da sva polja od „a8“ do „h7“ sadrže „djecu“ te da ta „djeca“ predstavlja čitavu kolekciju crnih figura, dok primjerice polje „a6“ ne sadrži nijedan podelement jer je s početkom partije to je polje prazno.

Upravljanje i logika igre

Temelj šaha predstavljaju polja i figure odnosno međudjelovanje figura i njihov utjecaj nad poljima ploče. Iako se šah naizgled čini igrom od koje se zahtjeva samo implementiranje mehaničke strane odnosno gdje se figura smije pomaknuti, a gdje ne smije, iznad te sfere postoji jedan upravljački aspekt koji sve te funkcionalne dijelove spaja u jednu cjelovitu sliku i čija uloga je konstantno motriti događanja na ploči te intervenirati ukoliko situacija zahtjeva šire tumačenje od onog implementiranog kroz sama polja i same figure.

Cjelovitu logičku stranu ploče je pokrivena skriptom „Igra“ koja ima razne zadaće te je najveća skripta u cijelom programu. Za potrebe ovog rada izdvojit će se dvije najvažnije metode u skripti pod nazivima „ProvjeriStanjePartije()“ i „MoguceBlokiratiSahFigurom()“. „ProvjeriStanjePartije()“ vraća vrijednost enumeracije „StanjePartije“ te za ulogu ima ispitati kakvo je trenutno stanje na ploči tako da analizira pozicije figura: npr. je li kralj u šahu i, ukoliko jest, je li moguće iz tog šaha izaći, a ako nije, partija je završena jer igrač više nema validni potez koji može izvršiti. „MoguceBlokiratiSahFigurom()“ vraća tip podataka bool odnosno odgovara na pitanje je li potezom blokiranja putanje napada na kralja moguće šah spriječiti. Sastoji od tri argumenata. Prvi argument je figura „blokFigura“. Taj argument predstavlja figuru za koju želimo provjeriti ima li mogućnost blokiranja šaha ili ne. Drugi argument je lista „figureKojeNapadajuKralja“ koja sadrži sve figure koje daju šah kralju. U većini slučajeva to će biti jedan figura. Treći argument je „napadnutiKralj“, pomoću koje imamo mogućnost pristupa poziciji napadnutog kralja na ploči,

ali i njegovim legalnim potezima kako bi se ustanovilo je li, ukoliko je već nemoguće blokirati šah figurom, moguće pomaknuti kralja na sigurno polje.

Figure i mehanika

Ovo poglavlje fokusira se na skripte „Polje“, „Figura“ te podklase skripte „Figura“, točnije, na svaku figuru ponaosob. Skripta „Polje“ vezana je za svaki pojedinačni element definiran u prethodnom poglavlju te predstavlja veoma bitan faktor kod legalnosti poteza određenih figura. Točnije, potezi koje sadrže figure u svojim listama legalnih poteza su zapravo polja na koja se te figure mogu pomaknuti.

Skripta „Figura“ služi kao osnovna klasa svim figurama za koje će se kasnije pojedinačno izvesti njihove distinktivne klase. Kao i uobičajeno kod nasljeđivanja, sve figure dijele neke zajedničke atribute i metode te su iste implementirane u ovoj skripti. Podklase skripte „Figura“ su skripte „Bijeli pijun“, „Crni pijun“, „Konj“, „Lovac“, „Top“, „Dama“ i „Kralj“.

Bijeli i crni pješaci su po funkcionalnostima identični uz jednu jedinu razliku a to je da se bijele figure kreću prema „gore“, a crne prema „dolje“. Eliminacija protivničke figure se pješakom vrši dijagonalno prema naprijed, osim u slučaju kad protivnički pješak preskoči branjeno polje. Tada pješak može uzeti protivničkog potezom „en passant“. Također, dolaskom do kraja ploče, pješak se mora promovirati u novu figuru, najčešće u damu iako je moguća promocija i u druge vrste figura.

Skripta „Konj“ ima najkraću i najjednostavniju implementaciju od svih figura. Iako su kretnje konjem najupečatljivije te ujedno i najnepredvidljivije na šahovskoj ploči, konj ne sadrži nikakve posebne poteze i specifične metode. Kreće se dva polja ravno potom lijevo ili desno, u obliku slova „L“. Ima mogućnost preskakanja drugih figura, bile one protivničke ili vlastite.

Skripta „Lovac“ specifična je zato što lovac teoretski može pokrivati mnogo polja istovremeno. Međutim, problem koji njihovo kretanje predstavlja je koliki je zapravo doseg dijagonale preko koje se lovčeva domena prostire. Budući da nije moguće preskakati figure, lovčeva kretnja limitirana je do polja na kojem se nalaze druge figure.

Skripta „Top“ dijeli velike sličnosti sa skriptom „Lovac“. Dok se lovac kreće dijagonalno, top se kreće ravno u svim pravcima. Također kao i kod lovca, nastaje problem kolika je zapravo domena kretanja topa budući da ne može preskakati figure te je njegova domena ograničena ili rubom ploče ili figurom koja stoji kao meta ili prepreka.

Skripta „Dama“ najduža je od svih skripti figura, ali s obzirom da predstavlja figuru koja je kombinacija lovca i topa, ista sadrži sve koncepte identične onima implementiranim u prethodne dvije skripte.

Skripta „Kralj“ predstavlja najvažniju figuru na ploči, figuru oko koje se cijela igra temelji i figure koja nije najjača na ploči, ali je najdragocjenija. Kralju je dozvoljeno kretati se za jedno polje u svim smjerovima, međutim u kombinaciji s topom, kralj ima mogućnost posebnog poteza pod nazivom rokada. Rokadom se kralj pomiče za dva mjesta ulijevo ili za dva mjesta udesno, a da se pritom top sa svoje početne pozicije pomakne, ukoliko je riječ o lijevoj rokadi, na polje kralju s desne strane, a ukoliko o desnoj, kralju s lijeve strane.



Slika br. 2: Kraj partije

Zaključak

U ovom radu ukratko je opisan proces izrade igre šaha u Unity okruženju. Objašnjena je igra šaha kako bi se čitatelj upoznao s jednostavnim pravilima te se potom obradio dio rada implementiran u samom Unity-u opisom vizualne implementacije šahovske ploče. Najvažnije i najbitnije koncepte implementirane u programskom kodu u kratkim crtama su objašnjeni prvo u upravljačko-logičkom segmentu rada, a zatim i u mehaničkom.

Reference

[1] Unity [Online]. Dostupno na: https://unity3d.com/get-unity/download [Pristupljeno: 05-vel-2021]
[2] Hrvatska enciklopedija, mrežno izdanje, šah. Leksikografski zavod Miroslav Krleža, 2021. [Online]. Dostupno na: https://www.enciklopedija.hr/Natuknica.aspx?ID=59291 [Pristupljeno: 21-srp-2021]
[3] Hrvatski šahovski savez, Pravila šaha FIDE. [Online]. Dostupno na: https://zlz-sport.hr/wp-content/uploads/Pravila-saha-FIDE-2014-v1.03-1.pdf [Pristupljeno: 21-srp-2021]
[4] hr.wikipedia.org, „Šah – Wikipedija“. [Online]. Dostupno na: https://hr.wikipedia.org/wiki/%C5%A0ah [Pristupljeno: 21-srp-2021]
[5] „Auto scaling grid layout code“ [Online]. Dostupno na: http://www.justapixel.co.uk/2014/12/08/ugui-auto-scaling-grid-layout-code/ [Pristupljeno: 02-lip-2021]

Izrada 2D platformske videoigre u Godotu

Dominik Kavur, Danijel Radošević i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
dkavur@foi.hr, darados@foi.hr, mlkoneck@foi.hr

Sažetak

Tema ovog rada je izrada 2D platformske igre u alatu Godot. Kreirana igra implementira osnovne mehanike platformera, prema uzoru na videoigru Super Mario. Ukratko će bit spomenut povijesni kontekst razvoja videoigara te će poseban kratak osvrt biti dan na važnost pozadinske glazbe i zvučnih efekata. Nakon toga će se ukratko opisati programski alat Godot te njegov programski jezik GDscript. Ukratko će se istaknuti određene prednosti i nedostatci programskog alata Godot. Posljednje, ukratko će biti opisana implementacija osnovnih mehanika tipičnog platformera.

Ključne riječi: videoigre, 2D platformska videoigra, Godot, GDscritp, programiranje, algoritmi

Uvod

Od 1950-tih pa sve do danas, popularnost videoigara je u neprestanom porastu te sve više i više ljudi igra videoigre. Na početku kao i danas, videoigre su bile te koje su predstavljale mogućnosti novog sklopovlja računala. Jednako kao i danas, igre su uvijek izvlačile najviše iz hardvera koji ih je pogonio. Zlatno dobar videoigara je započelo pojavom arkadnih igara koje su se igrale u javnih igraonicama, barovima i restoranima. Konzole su omogućile igranje videoigra u udobnosti vlastitog doma. Jedan od prvih žanrova videoigara je i žanr platformskih igara. Tehnološki razvoj je omogućio igranje videoigara na velikom broju različitih uređaja: osobna računala, konzole, tableti, pametni telefoni i dr. [2] [3] [4] [5].

2D platformer

Jedan od prvih žanrova videoigara uopće, 2D platformer se temelji na dvodimenzionalnog grafici te kontroli lika u dvodimenzionalnom prostoru. Igrač se najčešće kreće s lijeva na desno i prolazi svijetom igre. Osim kretanja, igrač najčešće može skakati, penjati se i pucati. Temeljni izazov razina videoigre je prelazak preko platformi između kojih se nalaze provalije. Platforme često nisu statične i to često zahtjeva dobru koordinaciju igrača kako bi uspio savladati neki dio razine. Glavni cilj je doći do kraja razine a osim

savladavanja platformi igrač se često bori i protiv neprijatelja koje treba izbjegavati ili eliminirati. Također, često igrač ima mogućnost skupljanja odgovarajućeg resursa, najčešće u obliku novčića. Tipične mehanike igranja su: kretanje, skakanje, napadanje, korištenje raznih pojačanja (engl. power-ups) i sl.

Glazba i zvučni efekti u video igrama

Segment koji je oduvijek bio neizostavan element je pozadinska glazba i zvučni efekti videoigre. Prvi zvučni efekti su bili 8-bitni zvukovi koji su se u videoigri pojavljivali i aktivirali u ključnim trenutcima kao što je prelazak razine, smrt igrača ili neprijatelja, ispucavanje metka, skupljanje novčića i sl. Zvučni efekti tog vremena su postali toliko voljeni da se i dan danas stvaraju videoigre koje koriste te 8-bitne zvučne efekte [6].

Kako je tehnologija napredovala tako je rastao i kapacitet medija za pohranu videoigara a samim time i količina i kvaliteta multimedijskog sadržaja. Mogućnosti reprodukcije zvuka s jednog kanala su se proširile na više kanala i sama reprodukcija zvuka u videoigrama je postajala sve sofisticiranija. Kako je rasla rezolucija ekrana tako je rasla i „rezolucija“ glazbenih podloga i zvučnih efekata. Mnoge pozadinske glazbe su postale vrlo prepoznatljive i popularna i izvan svijeta videoigara: svima je poznata originalna pozadinska glazba kultnog platformera Super Mario. Jednako tako i mnogi drugi naslovi su postali vrlo popularni, poput glazbe iz videoigara Halo ili Half Life. Također, mnoge igre su postale popularne i radi vrlo kvalitetnih zvučnih efekata, primjerice videoigra Call of Duty koja je uključivala vrlo realistične zvučne efekte oružja što je zasigurno doprinijelo boljem iskustvu igranja videoigre [6].

Godot engine

Programski alat Godot je pogon za razvoj videoigara otvorenog koda (engl. open source) koji postaje sve popularniji. Vrlo je pristupačan za početnike, zauzima vrlo malo prostora na disku za razliku od nekih drugih vrlo popularnih alata za razvoj videoigara kao što su Unity ili Unreal Engine. Mnoge stvari se mogu riješiti kroz samo sučelje i samim time je vrlo jednostavan za korištenje. Sam programski alat je napisan u C++ programskom jeziku te nudi svoj interni programski jezik GDscript koji je vrlo sličan programskom jeziku Python. Postoji i posebna inačica Godota u kojoj se može programirati logika igara i u programskom jeziku C#. Velika prednost Godaota je i njegova portabilnost [1].

Glavne značajke Godota su rad s čvorovima i scenama. Čvorovi predstavljaju zasebne jedinice namijenjene za određenu funkciju: igrač, kamera, pozadina, zvuk, pozicija na razini, kolizija, animacija i dr. Svaka scena videoigre se sastoji

od niza čvorova. Svaki čvor sadrži specifične opcije i gotove funkcionalnosti za interakciju s drugim čvorovima [1]. Scena predstavlja skup čvorova koji čine smislenu cjelinu. No postoji velika fleksibilnost jer je scenu kao takvu moguće i primjerice instancirati unutar druge scene. Tako je primjerice moguće kreirati scenu nekog neprijatelja u kojoj se nalazi sva grafika, animacije i logika ponašanja neprijatelja. Tada je vrlo jednostavno moguće instancirati objekte neprijatelja kada god je to potrebno [1].

Takav pristup je jako dobar jer se na početku razvoja prvo stvaraju elementi videoigre koji su višestruko iskoristivi a nakon toga se oni koriste u samoj videoigri. Također je moguće odvojiti vizualni dio od funkcionalnog što je vrlo dobar pristup za percepciju strukture i također je pristupačno onima koji tek uče osnove izrade videoigara.

Igra Skeletal Ruins

U prototipu koji je kreiran u sklopu ovog rada implementirane su neke osnovne karakteristike žanra platformera: 2D svijet koji je podijeljen u niz razina, razine koje se sastoje od niza platfori, lika koji upravlja igrač te neprijatelja koji pokušavaju zaustaviti igrača u prelasku razina.



Slika br. 1: Izgled igre Skeletal Ruins

Igrač upravlja glavnim likom vidoigre na tipičan način kako je to implementirani i u velikoj većini videoigara: korištenjem tipki W, A, S i D te se koristi tipka Space za skakanje. Osim običnog skoka u igri je implementirana i mogućnost dvostrukog skoka (engl. double jump). Kao i u mnogim drugim igrama ovog tipa, kretanje igrača je moguće kontrolirati i u zraku.

Igrač igru započinje s 3 života koja su vidljiva iznad lika igrača. Igrač može izgubiti život kolizijom s neprijateljskim jedinicama. Životni bodovi igrača se ne mogu nadoknaditi. Ako igrač izgubi sva 3 životna boda, tada započinje trenutnu razinu ali samo s jednim životnim bodom. Ovakav sustav tjera igrača na povećani oprez: igrač mora dobro planirati i čuvati životne bodove kako bi mogao savladati razine videoigre.

Igrač

Na sljedećoj slici mogu se vidjeti čvorovi koji su sastavni dio igrača.

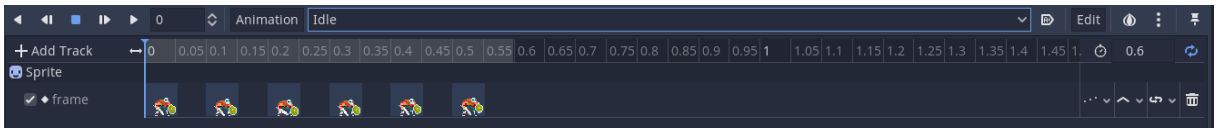


Slika br. 2: Čvorovi u Godotu

U hijerarhiji čvorova, može se vidjeti kako se svi čvorovi igrača nalaze kao podređeni čvorovi glavnog čvora igrača imena „Player“. Svaki čvor ima svoj meta naziv ali moguće je i svakom čvoru dodijeliti i vlastiti naziv. Meta naziv glavnog čvora imena „Player“ je „KinematicBody2D“. Ova vrsta čvora predstavlja 2D objekt na kojeg se dodaju ostali potrebni čvorovi. Za izgled se koristi čvor „Sprite“ putem kojeg se određuje grafički izgled igrača. Čvor „AnimationPlayer“ je čvor koji je zadužen za animiranje igrača. Ukratko, svaki čvor predstavlja dio funkcionalnosti lika igrača.

Animacije

Animacije je moguće kreirati u samom alatu Godot i vrlo je jednostavno kreirati vlastite animacije. Dva tipična načina izrade animacija je korištenje sličica animacije koje se reproduciraju u slijedu ili je moguće kreirati vlastitu animaciju translacijom pojedinih objekata koji sačinjavaju lika videoigre. U kreirano prototipu se koristi niz sličica koje predstavljaju animaciju lika.



Slika br. 3: Kreiranje animacija u Godotu

GDScript skripta

Skripta koja je zadužena za logiku ponašanja igrača izgleda ovako:

```
extends KinematicBody2D

const GORE = Vector2(0,-1)
const MAKS_BRZINA = 130
const SILA_SKOKA = 390
const AKCELERACIJA = 10

var kretnja = Vector2()
var pogled_desno =  true
var skok = 0
var maksSkok = 1

func _ready():
	pass

func izlaz_iz_igre():
	if Input.is_action_pressed("End"):
		get_tree().change_scene("res://MainMenu.tscn")

func padanje(): # Padanje i ograničavanje brzine padanja
	kretnja.y += 19
	if kretnja.y > 199:
		kretnja.y = 199

func pogled(): # Okretanje sprite-a
	if pogled_desno == true:
		$Sprite.scale.x = 1 #Normalna slika sprite-a
	else:
		$Sprite.scale.x = -1 #Zrcaljena slika sprite-a
```

```
func kretanje(): # Kretanje igrača
	kretnja.x = clamp(kretnja.x,-MAKS_BRZINA,MAKS_BRZINA)

	if Input.is_action_pressed("right"):
		kretnja.x += AKCELERACIJA
		pogled_desno = true
		$AnimationPlayer.play("Walk")
	elif Input.is_action_pressed("left"):
		kretnja.x -= AKCELERACIJA
		pogled_desno = false
		$AnimationPlayer.play("Walk")
	else:
		kretnja.x = lerp(kretnja.x,0,0.3)
		$AnimationPlayer.play("Idle")

	if skok < maksSkok:
		if Input.is_action_just_pressed("jump"):
			kretnja.y = -SILA_SKOKA
			skok += 1

	if is_on_floor() == true:
		skok = 0

	if !is_on_floor():
		if kretnja.y < 0:
			$AnimationPlayer.play("Jump")
		elif kretnja.y > 0:
			$AnimationPlayer.play("fall")

	kretnja = move_and_slide(kretnja, GORE)

func _physics_process(delta):
	Input.set_mouse_mode(Input.MOUSE_MODE_HIDDEN)
	izlaz_iz_igre()
	padanje()
	pogled()
	kretanje()
```

Funkcija „_physics_process(delta)“ se izvršava svaki puta kada se osvježi sličica videoigre što znači da će se izvoditi otprilike 60 puta u sekundi te predstavlja ugrađenu funkciju za tu svrhu u programskom alatu Godot. U toj funkciji se pozivaju sve preostale funkcije koje su zadužene za kretanje igrača: funkcija padanje() je primjena sile gravitacije na lika, funkcija pogled() iscrtava ispravan smjer gledanja igrača, funkcija kretanje() je zadužena za hodanje, skakanje i detekciju nalazi li se igrač na podlozi ili ne. Kretanje po osi X je kretanje ulijevo ili udesno, dok je kretanje po Y osi kretanje prema gore ili dolje.

Metoda „clamp“ je metoda koja ograničava brzinu kretanja između definirane minimalne i maksimalne vrijednosti. Metoda „lerp“ omogućava postepeno zaustavljanje igrača na način da se trenutna brzina kretnje postepeno interpolira prema vrijednosti 0: svaku deltu brzina se smanjuje za 3%.

Neprijatelji

Neprijatelji te kreću po mapi ulijevo i udesno, a igrač ih mora izbjegavati ukoliko ne želi izgubiti život. Neprijatelji su izrađeni kao zasebna scena te su instancirani u svakoj razini određeni broj puta.

Napredak i poboljšanja

U kreiranom prototipu implementirane su neke osnovne funkcionalnosti jednostavne 2D platformske videoigre. Ovaj prototip kao takav se može unaprijediti na niz različitih načina. Prvo, u grafičkom smislu moguće je napraviti brojna poboljšanja: bolja grafika koja bi stvorila i bolji vizualni doživljaj. Animirana pozadina kao i animirani objekti na sceni mogu dodatno oživjeti scenu i učiniti ju stvarnijom i „življom“. Također, sam dizajn razina bi se mogao unaprijediti: stvoriti izazovnije prepreke kao i pomične platforme, rotirajuće platforme i dr. Također, više različitih neprijatelja bi svakako poboljšalo sveukupni dojam: neprijatelji koji lete, neprijatelj koji mogu ispaljivati projektile na igrača i sl. Tu je naravno i element priče koji bi se mogao nadograditi: stvorili prepoznatljive likove, priču koju bi igrač mogao pratiti i koja bi igrača tjerala u sljedeći izazov.

Zaključak

2D patformeri su stari kao i same videoigre te su ponovno postali poprilično aktualni pojavom „indie“ videoigra koje su stari provjereni žanr obogatili novim suvremenim elementima. 2D platformeri i maju relativno jednostavne mehanike igranja i to je ono što je igračima danas vrlo interesantno, pogotovo u kontekstu igranja videoigara na mobilnim uređajima. Kada igrač poželi odigrati nešto „usput“ a da nije neka zbilja jednostavna igra, 2D paltformer je izvrstan izbor.

Godot je vrlo pristupačan i jednostavan alatu putem kojeg je relativno jednostavno implementirati osnovne mehanike 2D platformera. Dokumentacija alta je zbilja dobra te omogućava brzo učenje. Ono što je možda jedan nedostatak je nedovoljno edukacijskim materijala koji objašnjavaju neke naprednije koncepte no vjerojatno s vremenom će se pojaviti i novi, bolji edukativni materijali. Sistemski zahtjevi su također izuzetno niski što čini ovaj alat također vrlo praktičnim i prikladnim za razvoj jednostavnijih oblika videoigara.

Još jedna specifična prednost korištenja alata Godot je ako kreirat komercijalni proizvod u potpunosti možete zadržati sav zarađeni iznos, za razliku od bespaltnih alata kao što su Unity ili Unreal Engine koji nakon određene razine zarade uzimaju određena sredstva.

Reference

[1] GodotEngine, Godot Docs, https://docs.godotengine.org/en/stable/ (dostupno 28.11.2021.)
[2] Wikipedia The Free Encyclopedia, „First generation of video game consoles“, https://en.wikipedia.org/wiki/First_generation_of_video_game_consoles (dostupno 28.6.2021.)
[3] Wikipedia The Free Encyclopedia, „Second generation of video game consoles“, https://en.wikipedia.org/wiki/Second_generation_of_video_game_consoles (dostupno 28.6.2021.)
[4] Wikipedia The Free Encyclopedia, „Third generation of video game consoles“, https://en.wikipedia.org/wiki/Third_generation_of_video_game_consoles (dostupno 28.6.2021.)
[5] Wikipedia The Free Encyclopedia, „Fourth generation of video game consoles“, https://en.wikipedia.org/wiki/Fourth_generation_of_video_game_consoles (dostupno 28.6.2021.)
[6] „The Evolution of Video Game Music“, https://www.npr.org/templates/story/story.php?storyId=89565567 (dostupno 30.11.2021.)

Izrada „pixel art“ igre uloga u programskom alatu Unity

Danijel Tokić i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
dtokic@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu opisana je implementacija osnovnih mehanika izrade računalne igre uloga u programskom alatu Unity. Na početku rada opisani su korišteni alati te se ukratko daje opis žanra igara uloga. Nakon toga slijedi opis osnovnih mehanika upravljanja igrača, upravljanja resursima te interakcija s drugim likovima.

Ključne riječi: Unity, računalne igre, videoigre, igra uloga, algoritmi, programiranje, pixel art

Uvod

Igre uloga su jedan od najpopularnijih žanrova videoigara. Igre uloga na vrlo balansiran način kombiniraju elemente priče, akcije, savladavanja prepreka, zagonetki, upravljanje resursima i dr. [1]. Drugim riječima, u ovim videoigrama se isprepliću gotovo svi elementi dizajna videoigra koje možemo vidjeti u drugim žanrovima. Zbog svoje kompleksnosti i složenosti, postoji veliki broj visoko budžetnih igara koje su upravo ovog žanra [2]. Najčešća tematika igara ovog žanra su svjetovi fantazije ili znanstvena fantastika upravo zato što je ključni element žanra istraživanje stoga je kontekst svijeta vrlo značajan za videoigre ovog žanra.

Korišteni alati

Za izradu prototipa videoigre uloga korišten je programski alat Unity. Unity je jedan od najpopularnijih razvojnih okolina (engl. game engine) za razvoj videoigara na tržištu. Alat je besplatan za korištenje, jednostavan je i intuitivan te postoji puno edukativnih materijala koji pomažu u savladavanju rada u tom alatu [3]. Unity se može koristiti za razvoj 2D i 3D videoigra, kao i igara virtualne stvarnosti (VR) i proširene i prilagođene stvarnosti (AR). Programska logika videoigre se piše u programskom jeziku C# [4]. Za pisanje programskog koda korišten je Microsoftov alat Visual Studio [5]. Nakon uspješnog povezivanja s programskim alatom Unity, Visual Studio predstavlja odličnu radnu okolinu za pisanje programskog koda. Bojanje ključnih riječi i

Intellisense uvelike olakšavaju pisanje programskog koda. Unity je moguće povezati i s nekim drugim alatima osim Visual Studija.

Za izradu grafičkih elemenata videoigre korišten je alat Pixilart [6]. To je besplatna aplikacija za izradu pixel art grafike. Pixel art je forma digitalne grafike gdje je osnovni element piksel. Ovaj grafički stil svoje podrijetlo vuče iz vremena 8 bitnih i 16 bitnih računala gdje je ova vrsta grafike bila tehničko ograničenje. S obzirom na popularnost ovakve vrste grafičke forme, razvio se cijeli grafički stil koji i dana ima određenu popularnost [7]. Aplikacija Pixilart posjeduje sve potrebne alate za izradu grafičkih elemenata navedenog stila: rad sa slojevima, kreiranje animacija sličicu po sličicu, kreiranje liste slika animacije i sl.

Žanr igre uloga

Kao što je već navedeno, žanr igara uloga igrača smješta najčešće u neki izmišljeni svijet koji igrač istražuje. Među glavnim predstavnicima žanra nalaze se videoigre poput serijala „The Witcher“, serijala „The Elder Scrolls“, serijala „The Legend of Zelda“ i dr. [2]. U videoigri „The Witcher 3: Wild Hunt“ igrač je u ulozi lika koji već dobro poznaje svijet u kojem se nalazi dok u videoigri „The Elder Scrolls V: Skyrim“ igrač igru započinje kao lik koji nema posebnu ulogu u svijetu u kojem se nalazi i tek treba otkriti što ga čeka. U svakom slučaju, za igrača je svijet igre nepoznati i sam igrač treba upoznati svijet u kojem se nalazi i likove s kojima će se susretati kako bi otkrio priču koja prati glavnog lika videoigre.

Temeljna karakteristika igara uloga je da igrač istražuje svijet te u tom istraživanju rješava razne zadatke, skuplja predmete/resurse, skuplja sve jača oružja, i sam postaje sve jači i jači te stječe s vremenom razne sposobnosti koje mu mogu pomoći u savladavanju neprijatelja i u prolasku kroz svijet u kojem se nalazi.

Osnovne funkcionalnosti igre - igrač

Osnovne funkcionalnosti svih videoigara su funkcionalnosti koje se vežu za kretanje igrača. Također, tu su i osnovne mehanike interakcije. U igrama uloga često je jako značajno i samo sučelje videoigre na kojemu se nalaze relevantne informacije vezano za igrača, za trenutno stanje videoigre, pozicije u odnosu na svijet igre i sl. Na sučelju kreiranog prototipa nalaze se tipične informacije vezane za ovaj žanr igara: u gornjem lijevom uglu nalazi se avatar igrača, njegovi životni bodovi i razina mane koja se koristi za napad, kao i informacija o trenutnom napretku razvoja lika: trenutna razina i trenutno iskustvo.



Slika br. 1: Sučelje

Koncept iskustva je jako značajan u žanru igara uloga jer predstavlja temeljnu mehaniku putem koje igrač napreduje u videoigri: skupljanjem iskustva igrač postaje sve jači te otključava nove mogućnosti i sposobnosti. Iskustvo se stječe rješavanjem zadataka ili savladavanjem protivnika.

```
IEnumerator ResourcesOverTime () {
      while (true) {
            if (Resource < playerData.MaxResource)
                  Resource += ResourcePerSec;
            if (HP < playerData.MaxHP && !InCombat)
                  HP += HPPerSec;
            yield return new WaitForSeconds(1);
      }
}
```

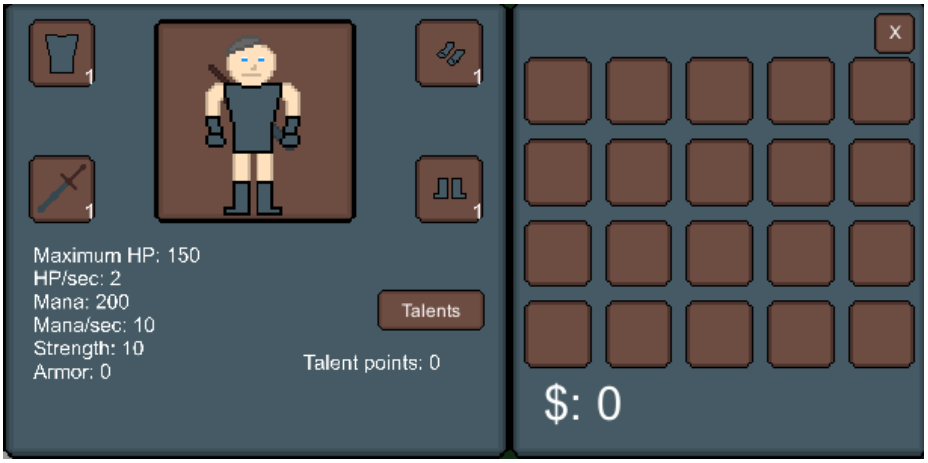
Tijekom igre, dok igrač nije u borbi, vraćaju mu se postepeno životni bodovi kao i razina mane. Funkcija ResourcesOverTime() prikazuje tu mehaniku regeneracije. U funkciji se nalazi beskonačna petlja koja svaku sekundu vraća igraču određenu razinu resursa. Ako je određeni resurs ispod maksimalne razine količine tog resursa, tada se resurs uvećava za odgovarajuću definiranu vrijednost.

U gornjem desnom uglu sučelja nalazi se gumb putem kojeg se može igra pauzirati. Pritiskom na ovaj gumb zaustavlja se izvođenje procesa koji predstavljaju aktivnosti svijet u kojem se igrač nalazi. Tu je i gumb za prikaz trenutnih zadatak (engl. quests) koje igra može i treba izvršiti kako bi napredovao kroz igru.

U donjem dijelu korisničkog sučelja s lijeve strane nalazi se kontrola putem koje igrač upravlja kretanjem svoga lika: pomicanjem „gljive“ prema rubu lik

igrača će se kretati u definiranom smjeru. Na sredini nalazi se mjesto gdje igrač može staviti neki objekt za brzo korištenje, primjerice napitak za vraćanje životnih bodova. S desne strane nalaze se kontrole za različite vrste napada.

Kod igara uloga igrač osim određenih sposobnosti može nositi i brojne predmete a također se može opremiti raznom opremom koja mu može pomoći u borbama. Na slici br. 2. vidi se inventar u kreiranom prototipu videoigre gdje se vidi na lijevoj strani trenutno opremljena oprema igrača zajedno s dodatnim karakteristikama. S desne strane mogu se vidjeti objekti koje igrač sa sobom nosi.



Slika br. 2: Inventar

Resursi igrača

Prolaskom kroz svijet videoigre, igrač skuplja razne resurse kao što je novac koji može upotrijebiti kako bi kupio u trgovini određene objekte, bilje koje može koristiti kao recepte za stvaranje napitaka i sl. Resursi se mogu pronaći u svijetu videoigre a također se mogu steći rješavanjem zadataka. Neki resursi se mogu pronaći samo jednom dok neki resursi se mogu pronaći višekratno. Primjerice, bilja koju smo pokupili će nakon nekog vremena izrasti pa će biti opet dostupna igraču.

Resurse se koriste za kreiranje predmeta koji igraču mogu pomoći u savladavanju zadataka i prelasku videoigre: kreiranje oklopa i oružja ili napitaka i sl. U igrama ovog tipa često postoje brojni resursi. Igrač neke resurse može izravno iskoristiti dok druge može prodati u trgovinama i zamijeniti za novce koje opet može koristiti za prikupljanje resursa koji su mu značajni.

Likovi

Svijet videoigre bi bio vrlo monoton i statičan da u njemu ne postoje i drugi likovi. U ovim igrama možemo vidjeti likove koji imaju raznolike uloge: likovi koji su borci, trgovci, proizvođači i sl. Ovisno o ulozi, likovi imaju odgovaraju sposobnosti koje igrač može iskoristiti kako bi postigao neki željeni rezultat. Tako je primjerice u kreiranom prototipu kreirana uloga alkemičara koji može kreirati specifične napitke. Alkemičar može izraditi za igrača određeni napitak no igrač će trebati za to izdvojiti određene resurse: novac kako bi napitak kupio, dati odgovarajuće elemente kako bi se napitak izradio a također će trebati i pričekati određeno vrijeme kako bi se napitak i napravio.

Priča je također jedan od ključnih elemenata ugara uloga. U prototipu su kreirani razni likovi s kojima igrač može razgovarati te na taj način otkrivati informacije o svijetu u kojem se nalazi, otključavati nove zadatke i sl.

Neprijatelji i borba

Osim prijateljskim likova sastavni dio svijeta su i neprijateljski likovi s kojima se igrač može boriti i koje je potrebno savladati kako bi se ostvario neki željeni cilj. Mehanika napada ovisi o klasi igrača. U kreiranom prototipu kreirane su 3 različite klase: ratnik, strijelac i čarobnjak. To su tipične klase kod ovakvih vrsta igara. Kada igrač dođe u neposrednu blizinu s neprijateljskim likom, neprijatelj će krenuti napadati igrača. Kako bi se igrač obranio, igrač treba napasti i savladati neprijatelja. Svaki napad igrača košta određenu količinu mane. Također, kako bi igrač mogao izvesti neku vrstu napada, mora imati i opremljeno odgovarajuće oružje. Sama mehanika napada je implementirana putem sudarača (engl. colliders): primjerice, kada se mač igrača nađe u koliziji s tijelom neprijatelja, uspješno je zadan udarac neprijatelju koji će mu oduzeti odgovarajuću razinu životnih bodova.

```
private void OnTriggerEnter2D (Collider2D collision) {
      if (CheckCondition(collision.gameObject)) {
            DamageTarget(collision.gameObject);
            Attack.Cancel(false);
            Destroy(gameObject);
      }
      else if (collision.CompareTag("Environment")) {
            Attack.Cancel(false);
            Destroy(gameObject);
      }
}
```

U navedenom isječku koda vidljiva je funkcija OnTriggerEnter2D() koja se automatski poziva kada objekt na kojemu je pridružena skripta u kojoj se

nalazi ova funkcija dođe u koliziju s nekim drugim objektom koji na sebi ima komponentu sudarača. Prvo se provjerava ima li objekt s kojim se dogodila kolizija ispravnu oznaku: u slučaju napada igrača objekt treba imati oznaku neprijatelja i obrnuto. Ako je uspješno izvršen napad, izvršava se šteta nad objektom kolizije, napad se nakon toga prekida. Napad se također prekida i prilikom napada na objekt koji ima oznaku okoline (npr. ispucavanje strijele u zid).

Šteta koja se zadaje objektu s kojim je nastala kolizija se računa na temelju štete oružja te količini obrane koju protivnik ima. Često se kod izračuna štete uzimaju razna svojstva u obzir a također postoji često i djelomična nasumičnost koja se ugrađuje kako bi se dobila i neka realističnost da svaki udarac ne nanosi točno istu razinu štete. Ponekad se osim štete prenose napadima i posebni efekti koji traju određeno vrijeme (npr. svaku sekundu neprijatelj će izgubiti neku malo količinu životnih bodova – efekt zaraze). Također moguće je i postaviti određene pozitivne zaštitne efekte na igrača: povećana zaštita od određene vrste napada i sl.

Zaključak

Izrada videoigara je dugotrajan i složen posao, kako s dizajnerske strane gledišta tako i iz tehničke strane. Ako jedna osoba izrađuje videoigru mora imati multidisciplinarno znanje i vještine: izrada grafičkih elemenata, poznavanje koncepata dizajna videoigara, mehanika i sl. Također mora znati programirati programsku logiku videoigre i dr. Videoigre uloga su zahtjevnije i zbog toga što uključuju niz različitih elemenata u odnosu na neke jednostavnije žanrove koje se temelje na manjem broju mehanika igranja. Programski alat Unity uvelike olakšava određene faze razvoje jer automatizira izradu i implementaciju mnogih osnovnih funkcionalnosti videoigre. Veliki izazov je kreirati iskustvo igranja koje će biti jedinstveno, zanimljivo i koje se neće ponavljati već će svakim trenom biti nadograđeno s nekim novim elementom kako igranje videoigre ne bi vrlo brzo dosadilo igraču.

Reference

[1] Role-playing game. https://en.wikipedia.org/wiki/Role-playing_game (Dostupno: 2.9.2021.)
[2] The Most Popular RPG Video Games Right Now. https://www.ranker.com/list/most-popular-rpg-video-games-today/ranker-games (Dostupno: 2.9.2021.)
[3] J. Brodkin: How Unity3D Became a Game-Development Beast. https://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/ (Dostupno: 1.9.2021.)

[4] Unity Technologies, MonoBehaviour [Unity Documentation]. https://docs.unity3d.com/ScriptReference/MonoBehaviour.html (Dostupno: 1.9.2021.)
[5] Microsoft, Visual Studio 2019, https://visualstudio.microsoft.com/vs/ (Dostupno: 1.9.2021.)
[6] B. Ware: An art community for everyone! https://www.pixilart.com/about (Dostupno: 1.9.2021.)
[7] Pixel art. Wikipedia. https://en.wikipedia.org/wiki/Pixel_art (Dostupno: 1.9.2021.)

Izrada videoigre preživljavanja u alatu Unreal Engine 4

Emil Mihalić, Danijel Radošević i Mladen Konecki
Fakultet organizacije i informatike, Sveučilište u Zagrebu
emilmihalic3@gmail.com, darados@foi.hr, mlkoneck@foi.hr

Sažetak

U ovom radu opisat će se određeni aspekti implementacije igre preživljavanja u programskom alatu Unreal Engine 4. Primarni fokus bit će na mehanikama preživljavanja što su temeljne mehanike žanra. Dijelovi programske logike bit će implementirani putem nacrta (engl. Blueprints). Određeni segmenti su implementirani u programskom jeziku C++.

Ključne riječi: Unreal Engine 4, C++, nacrti, videoigra preživljavanja, programiranje, algoritmi, umjetna inteligencija

Uvod

Industrije zabavnih sadržaja iz godine u godinu postaju sve sofisticiranije i naprednije i nude razne oblike multimedijskih sadržaja koji pojedincu mogu pružiti zabavu i zadovoljstvo. Ogromna industrija koja ima veliku ulogu u tome je i industrija videoigara. Po svom opsegu, industrija videoigara je u 2020. godini ostvarila prihod od 159,3 milijarde američkih dolara [1]. Industrija videoigara se počela razvijati 1970-tih godina te je do današnjeg dana postala jedna od najvećih industrija zabave na svijetu [2]. Postala je veća od glazbene i filmske kino industrije te televizijske industrije filmova i serija [3].

Razvijati videoigre nije nimalo jednostavan zadatak, posebice danas, kada se standard kvalitete digao na vrlo visoku razinu. Pojedincima koji žele svoje ideje pretvoriti u interaktivne žive svjetove u obliku videoigara mogu pomoći pogoni za razvoj videoigara kao što su na primjer programski alati Unity ili Unreal Engine 4 [4][5]. Pogoni za razvoj videoigara već imaju implementirane mnoge funkcionalnosti koje svaka videoigra mora imati. Korištenjem pogona za razvoj videoigara pojedinac se ne mora toliko fokusirati na tehničke aspekte razvoja videoigre već se može fokusirati na sam sadržaj videoigre koju želi izraditi.

U ovom radu će se opisati određene mogućnosti programskog alata za razvoj videoigara Unreal Engine 4 te će se putem njega kreirati prototip videoigre preživljavanja. Glavni fokus stavljen je na logiku ponašanja igrača, upravljanje oružjem, na umjetnoj inteligenciji neprijateljskih jedinica, skupljanje resursa za

preživljavanje i mehaniku dolaska neprijatelja u valovima. Glavni fokus je stavljen na programsku logiku videoigre: modeli i vizualni efekti su preuzeti s Interneta za potrebe izrade ovog rada te se njihovo stvaranje neće opisivati.

Tema videoigre

Glavna ideja kreiranog prototipa videoigre je kreirati akcijsku videoigru koja će u sebi sadržavati elemente dva osnovna podžanra akcijskih videoigara: elemente igara preživljavanja i elemente igara pucanja [6]. Osnovne mehanike igranja se temelje na refleksima, brzim reakcijama igrača na opasnosti iz okoline te na koordinaciji između ruku i očiju igrača: igrač će morati prepoznati prijetnju i onda preciznim pokretima tu prijetnju eliminirati.

Igrač ima mogućnost slobodnog kretanja po mapi na kojoj se nalazi. Glavni cilj videoigre je preživjeti nalete neprijatelja koji dolaze u unaprijed definiranim valovima. Svaki val progresivno postaje sve teži i teži tako što dolazi sve više i više protivnika. Što se tiče elemenata žanra preživljavanja, u prototip su ugrađene mehanike gladi i žeđi. Osim toga, igrač ima i ograničenu municiju što je također jedno tipično ograničenje u akcijskim igrama.

Kako bi igrač preživio što duže, igrač će morati savladavati nadolazeće neprijatelje, a pritom skupljati resurse kako bi zadovoljio potrebe za hranom i vodom. Osim toga, potrebno je sakupljati i municiju te je moguće izgubljene životne bodove nadomjestiti pronalaskom paketa prve pomoći. Atmosfera u videoigri ima elemente horora primarno kroz dizajn neprijateljskih jedinica koje imaju izgled zombija. Osnovna inspiracija za izradu ovog prototipa dolazi iz popularnih sličnih videoigara kao što su *DayZ* i *Scum* [7][8].

Kretanje igrača

```
void MoveForward(float Value);
void MoveRight(float Value);

void ASCharacter::MoveForward(float Value)
{
AddMovementInput(GetActorForwardVector() * Value);
}
void ASCharacter::MoveRight(float Value)
{
AddMovementInput(GetActorRightVector() * Value);
}

PlayerInputComponent->BindAxis("MoveForward", this,
&ASCharacter::MoveForward);
PlayerInputComponent->BindAxis("MoveRight", this,
&ASCharacter::MoveRight);
```



Slika br 1: Dodavanje kontrola za kretanje

Kao i kod manje-više svih videoigara, prva osnovna mehanika koja nam je važna je kretanje igrača. Kao što je vidljivo na temelju prikazanog programskog koda, igraču je omogućeno kretanje u svim smjerovima: naprijed, natrag, lijevo i desno. Kombinacijom kretanja u više smjerova možemo dobiti fine pokrete u točno željenom smjeru. U postavkama projekta definirane su i tipke na tipkovnici koje će se mapirati uz ove osnovne mehanike kretanja.

```
PlayerInputComponent>BindAxis("LookUp", this,
&ASCharacter::AddControllerPitchInput);
PlayerInputComponent>BindAxis("Turn", this,
&ASCharacter::AddControllerYawInput);
```



Slika br. 2: Dodavanje kontrola za gledanje

Osim samog kretanja, igrač ima i mogućnost gledanja u svim smjerovima, a ta mehanika se veže uz pomicanje miša. Tako igrač lijevom rukom pomiče svog lika pritiskom na tipke A, S, D i W dok desnom rukom vrši rotiranje pogleda korištenjem miša. Osim toga implementirane su također dodatne vrlo standardne mehanike kretanja: čučanj, skakanje i trčanje. Čučanj se implementira tako da promijenimo poziciju igrača po Z osi. Skakanje je također promjena pozicije po Z osi uz primjenu osnovnih zakona fizike (gravitacija koja nas „vuče“ natrag prema dolje). Trčanje je implementirano tako da mijenjamo brzinu kretnje igrača. Za sve ove mehanike na model igrača dodane su potrebne vizualne komponente: animacije pokreta, razni drugi vizualni i zvučni efekti. Implementirana je i mehanika životnih bodova koje igrač gubi u trenutku napada neprijatelja, a životne bodove može vratiti skupljanjem resursa za povratak životnih bodova.

Oružje

Kako bi igrač preživio valove neprijatelja, neće se moći samo kretati po mapi i izbjegavati neprijatelje. Stoga je implementirana i mehanika korištenja oružja. Osnovna mehanika je naravno ispucavanje metaka iz oružja. Najlogičniji način kako implementirati ovu mehaniku je korištenjem „Line trace“ metode. Ovom metodom kreiramo liniju koja se kreira u centru kamere koja prati igrača. Prvi objekt s kojim se ta linija nađe u koliziji je objekt koji će biti pogođen metkom koji se ispaljuje iz oružja. Metak kao takav zapravo ne postoji: postoji vizualni efekt koji prikazuje (simulira) ispaljivanje metka, ali ništa se zapravo ne instancira prilikom korištenja oružja već se detekcija kolizije temelji na koliziji te linije i objekata na sceni. Ovakav način „pucanja“ se može implementirati na različite načine, a u ovom prototipu koristi se vrlo jednostavan način: ravna linija se iscrtava od sredine ekrana prema ravno, u smjeru pogleda igrača koji je definiran pokretom miša. Ovisno o vrsti objekta s kojim je linija u koliziji, vršimo odgovarajuću programsku logiku: ako je linija u koliziji s neprijateljem, tada će mu se nanijeti odgovarajuća razina štete.

```
virtual void Fire();

AActor* MyOwner = GetOwner();
if (MyOwner) {
    FVector EyeLocation;
    FRotator EyeRotation;
    MyOwner->GetActorEyesViewPoint(EyeLocation, EyeRotation);
    FVector ShotDirection = EyeRotation.Vector();
    FVector TraceEnd = EyeLocation + (ShotDirection * 10000);
    FCollisionQueryParams QueryParams;
    QueryParams.AddIgnoredActor(MyOwner);
    QueryParams.AddIgnoredActor(this);
    QueryParams.bTraceComplex = true;
    QueryParams.bReturnPhysicalMaterial = true;

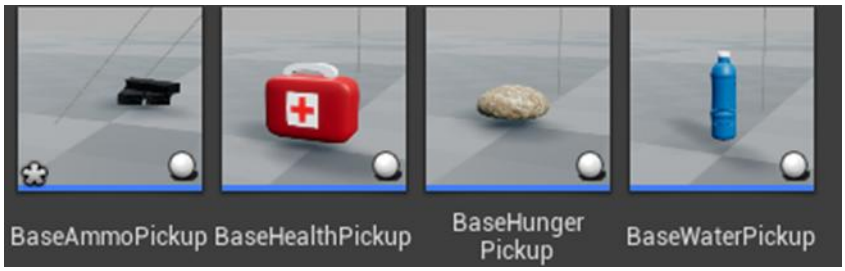
    FHitResult Hit;
    If (GetWorld()->LineTraceSingleByChannel(Hit,EyeLocation,
         TraceEnd, COLLISION_WEAPON, QueryParams)) {  }
}

virtual FVector GetPawnViewLocation() const override;
FVector ASCharacter::GetPawnViewLocation() const {
    if (CameraComp) {
        return CameraComp->GetComponentLocation();}
    return Super::GetPawnViewLocation();
}
```

Ova funkcija za pucanje je nadograđena s nekim tipičnim funkcionalnostima za neke dodatne mogućnosti pucanja: automatsko pucanje, blaga devijacija

(nasumičnost) u putanji ispucavanja metaka, ograničen broj municije u magazinu oružja, punjenje magazina. Za sve navedene mehanike dodane su odgovarajuće animacije te vizualni i zvučni efekti.

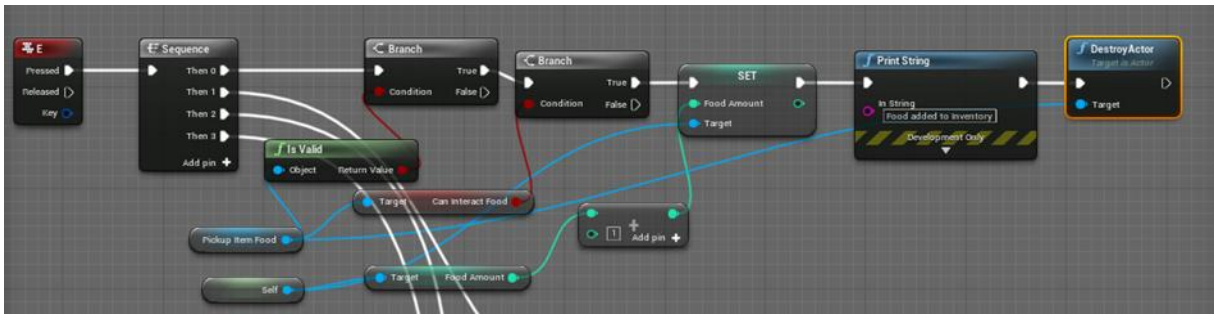
Skupljanje resursa



Slika br 3: Klase za svaki resurs

Videoigre preživljavanja su karakteristične po tome što imaju određene mehanike koje se vežu uz preživljavanje: najčešće se temelje na nekim osnovnih ljudskim potrebama kao što su potreba za hranom, pićem, odmorom i sl. U kreiranom prototipu su implementirane dvije najčešće mehanike, a to su mehanike gladi i žeđi. Glad i žeđ polako s vremenom rastu, a kako bi nadomjestio izgubljeno, igrač mora skupljati resurse hrane i pića. Osim hrane i pića, igrač treba skupljati i metke koje troši korištenjem oružja. Također, u slučaju da neprijatelj uspije doći do igrača i nanijeti mu štetu, igrač skupljanjem kutije prve pomoći može nadoknaditi odgovarajuće izgubljene životne bodove.

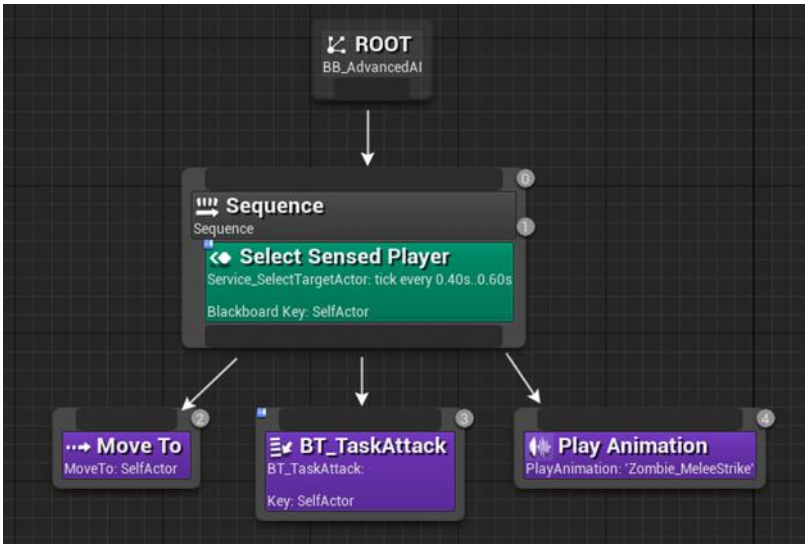
Za skupljanje objekata se također koristi „Line trace“ metoda pomoću koje prepoznajemo objekt koji se nalazi ispred igrača. Ukoliko je objekt dovoljno blizu igrača i ako igrač gleda u pravcu tog objekta tada su zadovoljeni uvjeti za interakciju s tim resursom. Ova mehanika je implementirana vizualnim skriptiranjem. Slika ispod je nacrt konačnog koraka interakcije: prije toga je bilo potrebno provjeriti definirane uvjete za interakciju.



Slika br 4: Interakcija sa resursom

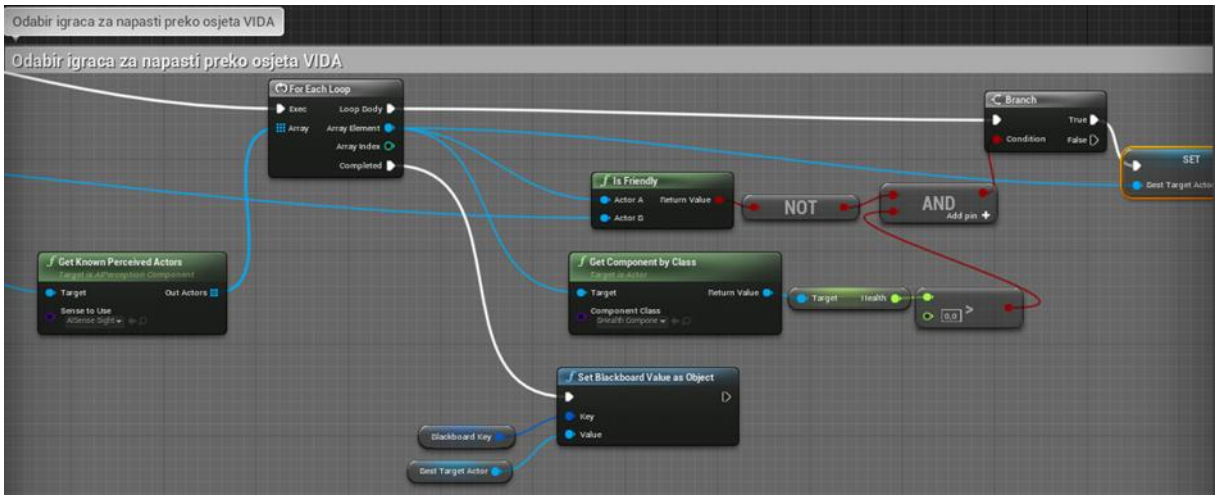
Umjetna inteligencija neprijatelja

Kako bi videoigra bila izazovna, u videoigri se nalaze neprijatelji koji će pokušati nanijeti štetu igraču i u konačnici pokušati mu oduzeti sve životne bodove. Tematika i motivi zombija su česti u videoigrama, posebice u nekim horor ili apokaliptičnim scenarijima. Umjetna inteligencija ponašanja neprijatelja je ostvarena korištenjem stabla ponašanja. Ponašanje zombija je implementirano na sljedeći način: neprijatelj pokušava „osjetiti" igrača, vidom ili sluhom. Nakon što neprijatelj otkrije poziciju na kojoj se igrač nalazi, prvo će se početi kretati prema poziciji igrača. Nakon što dođe u neposrednu blizinu igrača, izvršava se napad koji igraču radi štetu. Prilikom hodanja ili napada zombija izvršavaju se odgovarajuće animacije pokreta.



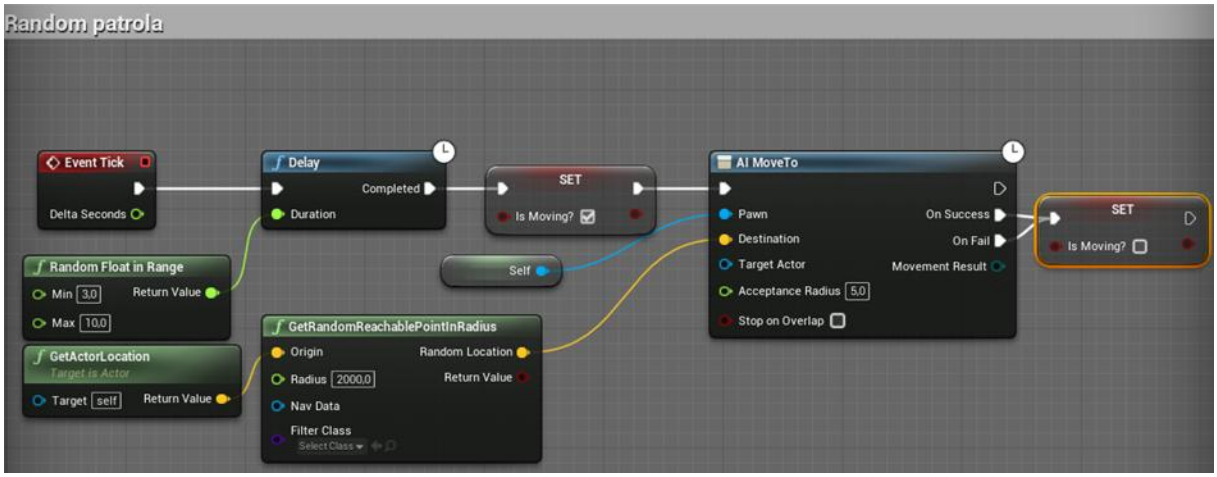
Slika br 5: Stablo ponašanja

Neprijatelji i igrač imaju neke zajedničke karakteristike: komponentu životnih bodova, detekciju kolizije i sl. Za razliku od igrača, koji se naravno kreće u skladu s ulaznim podatcima koji dolaze s tipkovnice i miša, kretanje neprijatelja je unaprijed definirano stablom ponašanja. Procedura „osjeta" igrača se provodi na sljedeći način: ako igrač dođe u vidno polje zombija, zombi će prepoznati igrača i krenuti u napad. Osim osjeta vida zombiji imaju i osjet sluha koji ima dosta veći radijus detekcije: zombi može „čuti" neku akciju koja se događa na mapi (npr. hodanje, trčanje, pucanje metaka). Nakon što zombi čuje neki zvuk, usmjerit će pažnju prema tom dijelu mape te će provjeriti je li možda negdje u blizini igrač. Stoga igrač mora biti oprezan što radi: izbjeći vidno polje zombija je jednostavnije nego izbjeći osjet sluha. Također, kada zombiji ne bi reagirali na zvuk tada i njihovo ponašanje ne bi bilo toliko sofisticirano i zapravo bi bilo vrlo jednostavno izbjeći napade neprijatelja – trčanjem bi se moglo pobjeći iz njihovog vidnog kruga i na taj način jednostavno izbjeći napade.



Slika br 6: Odabir mete pomoću osjetila vida

Neprijatelji se stvaraju u valovima. Svaki val je teži od prethodnog. Bez obzira na osjetila vida i sluha neprijatelja, videoigra bi mogla biti poprilično pasivna: igrač ne radi buku i izbjegava vidna polja neprijatelja. Kako bi se izbjegao taj scenarij, neprijatelji kada nemaju informaciju o poziciji igrača patroliraju mapom tako da odabiru nasumične točke na koje će se kretati. Nakon što dođu do točke prema kojoj su išli, tu provedu nekoliko sekundi te nakon toga odabiru novu nasumičnu točku prema kojoj se kreću. Na sljedećoj slici se može vidjeti implementacija mehanike patroliranja neprijatelja po mapi.



Slika br 7: Nasumično patroliranje neprijatelja

Valovi neprijatelja

Za kraj, objasnit ćemo i način na koji funkcionira mehanika stvaranja valova neprijatelja. Jednom kada su neprijatelji u potpunosti implementirani, sada je potrebno kreirati stvaranje neprijatelja na određeni način. Na mapi su određene točke stvaranja neprijatelja. Na početku vala svaki neprijatelj se stvara na nasumično odabranoj točki od svih definiranih točaka stvaranja koje su

razbacane po mapi. Postoji jedno ograničenje: ako se kod neke točke nalazi igrač tada se neprijatelj ne može stvoriti na toj točki. Na taj način se izbjegava scenarij gdje se neprijatelj stvara u neposrednoj blizini igrača. Uvjet kada se stvara novi val neprijatelja je da su svi neprijatelji na mapi eliminirani. Svaki idući val sadrži sve više i više neprijatelja kako bi valovi bili sve izazovniji za savladavanje.

```
void ASGameMode::StartWave() {
	WaveCount++;
	NrOfBotsToSpawn = 20 + (WaveCount*2);
	GetWorldTimerManager().SetTimer(TimerHandle_BotSpawner,this,
	&ASGameMode::SpawnBotTimerElapsed, 1.0f, true, 0.0f);
	SetWaveState(EWaveState::WaveInProgress);
}
void ASGameMode::EndWave() {
	GetWorldTimerManager().ClearTimer(TimerHandle_BotSpawner);
	SetWaveState(EWaveState::WaitingToComplete);
}
void ASGameMode::SpawnBotTimerElapsed() {
	SpawnNewBot();
	NrOfBotsToSpawn--;
	if (NrOfBotsToSpawn <= 0) {
		EndWave();
	}
}
void ASGameMode::CheckWaveState() {
	bool bIsPreparingForWave = GetWorldTimerManager().IsTimerActive
				(TimerHandle_NextWaveStart);
	if (NrOfBotsToSpawn > 0 || bIsPreparingForWave) return;
	bool bIsAnyBotAlive = false;
	for (FConstPawnIterator It = GetWorld()->GetPawnIterator(); It;
							++It) {
		APawn* TestPawn = It->Get();
		if (TestPawn == nullptr || TestPawn->IsPlayerControlled()){
			continue;
		}
		USHealthComponent* HealthComp = Cast<USHealthComponent>
	(TestPawn->GetComponentByClass
				(USHealthComponent::StaticClass()));
		if (HealthComp && HealthComp->GetHealth() > 0.0f) {
			bIsAnyBotAlive = true;
			break;
		}
	}
	if (!bIsAnyBotAlive) {
		SetWaveState(EWaveState::WaveComplete);
		PrepareForNextWave();
	}
}
```

```
void ASGameMode::PrepareForNextWave() {
      GetWorldTimerManager().SetTimer(TimerHandle_NextWaveStart,this,
      &ASGameMode::StartWave, TimeBetweenWaves, false);
      SetWaveState(EWaveState::WaitingToStart);
}
```

Programski kod koji je iznad naveden je zaslužan za programsku logiku stvaranja, trajanja i završetka valova neprijatelja. Svaka metoda ima svoju odgovarajuću funkcionalnost: početak novog vala, završetak vala, provjeru trenutnog stanja, učitavanje idućeg vala i sl.

Zaključak

Izrada ovog prototipa implementira osnovne funkcionalnosti videoigre preživljavanja iz trećeg lica. Opisano je i demonstrirano kako implementirati određene funkcionalnosti klasičnim pisanjem programskog koda, ali i kako implementirati određene funkcionalnosti korištenjem nacrta. Ovisno o kontekstu, moguće je koristiti jedan ili drugi način, ovisno o tome što se korisniku čini praktičnije.

Izraditi prototip videoigre nije jednostavan zadatak: prototip mora imati niz funkcionalnosti koje ima i cjelovita videoigra. Ako se uzme u obzir da u praksi nije potrebno raditi samo programsku logiku već i sve ostale aspekte i segmente videoigre (npr. vizualne i zvučne elemente, animacije, efekte i dr.) tada je jasno zašto je potrebno uložiti puno truda kako bi se razvila kvalitetna, zanimljiva i zabavna videoigra.

Reference

[1] Tom Wijman. The World's 2.7 Billion Gamers Will Spend $159.3 Billion on Games in 2020; The Market Will Surpass $200 Billion by 2023. 2020. (Pristup: 11.11.2021.) Dostupno na: https://newzoo.com/insights/articles/newzoo-games-market-numbers-revenues-and-audience-2020-2023/

[2] Video game industry. Wikipedia. (Pristup: 11.11.2021.) Dostupno na: https://en.wikipedia.org/wiki/Video_game_industry

[3] Maaani. Gaming Industry is Greater than Movies and Music Industry. (Pristup: 11.11.2021.) Dostupno na: https://www.oneangrygamer.net/2021/09/gaming-industry-bigger-than-movies-and-music/

[4] Unity. (Pristup: 11.11.2021.) Dostupno na: https://unity.com/

[5] Unreal Engine. (Pristup: 11.11.2021.) Dostupno na: https://www.unrealengine.com/en-US/

[6] List of video game genres. Wikipedia. (Pristup: 12.11.2021.) Dostupno na: https://en.wikipedia.org/wiki/List_of_video_game_genres

[7] DayZ. Steam. (Pristup: 13.11.2021.) Dostupn na: https://store.steampowered.com/agecheck/app/221100/
[8] SCUM. Steam. (Pristup: 13.11.2021.) Dostupno na: https://store.steampowered.com/app/513710/SCUM/

STRUČNA KONFERENCIJA

RAČUNALNE IGRE 2021

31. prosinca 2021.