UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marko Kladnik

Primerjava igralnih pogonov Unity in Unreal Engine

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Igor Rožanc

Ljubljana, 2015

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marko Kladnik

Primerjava igralnih pogonov Unity in Unreal Engine

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Igor Rožanc

Ljubljana, 2015

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Naslov: Primerjava igralnih pogonov Unity in Unreal Engine

Tematika naloge:

Igralni pogoni so programerska orodja, ki omogočajo učinkovito izgradnjo računalniških iger za različne platforme. V diplomski nalogi predstavite značilnosti in uporabo dveh popularnih igralnih pogonov Unity in Unreal Engine. Z obema pogonoma razvijte enako računalniško igro ter na podlagi tega opravite primerjavo obeh orodij po več izbranih kriterijih, ki so pomembni za razvijalca računalniških iger.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Marko Kladnik sem avtor diplomskega dela z naslovom:

Primerjava igralnih pogonov Unity in Unreal Engine

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 14.9.2015 Podpis avtorja:

Zahvaljujem se svoji družini in Simoni za vso pomoč in podporo med študijem. Zahvala gre tudi viš. pred. dr. Igorju Rožancu za vso pomoč pri izdelavi diplomske naloge.

Kazalo

Povzetek

Z naraščanjem popularnosti računalniških iger raste tudi popularnost orodij (t.i. igralnih pogonov), ki so namenjena za njihovo ustvarjanje. Diplomska naloga strnjeno predstavi spremembe, ki so vodile do množičnega razmaha iger in njihovo zgodovino. Osrednji del je namenjen predstavitvi igralnih pogonov Unity in Unreal Engine. Unity je najbolj uporabljan igralni pogon, ki ga je leta 2005 predstavilo podjetje Unity Technologies. Unreal Engine je igralni pogon, ki omogoča ustvarjanje iger s čudovito grafiko. Izdalo ga je podjetje Epic Games leta 1998. Pogona sta namenjena lažjemu ustvarjanju in oblikovanju računalniških iger. V nadaljevanju je predstavljena izdelava iger v vsakem izmed njiju, čemur sledi primerjava obeh z različnih vidikov. Primerjali smo grafiko, dostopnost kode, ustvarjanje novih objektov in še mnoge druge vidike. Primerjavo spremlja tudi ocenjevanje lastnosti obeh igralnih pogonov. Ugotovili smo, da ima vsak izmed njiju svoje prednosti in slabosti. Unity je primernejši za izdelavo dvodimenzionalnih in mobilnih iger, medtem ko je Unreal Engine boljša izbira za izdelavo tridimenzionalnih iger.

Ključne besede: Unity, Unreal Engine, igralni pogon, primerjava, računalniške igre

Abstract

With the increasing popularity of computer gaming, the growth of game engines popularity comes as no surprise. The thesis shortly presents the changes that led to games as we know them today, followed by presentation of their history. The main focus is on the presentation of Unity and Unreal Engine. Unity is the most used game engine which was developed by Unity Technologies in 2005. Unreal Engine is known for its amazing graphic. It was developed by Epic Games in 1998. Their main functions enable developers to create games in an easier way. What follows is a presentation of the games developed using both engines, extended by the comparison between the two from different points of view. We compared and assessed source code access, creation of new game objects, graphics and other characteristics. Both engines have advantages and disadvantages. Unity is a better choice when developing two-dimensional and mobile games, while Unreal is more suitable for developing three-dimensional games.

Key words: Unity, Unreal Engine, game engine, comparison, computer games

1 Uvod

Računalniške igre so vedno imele posebno mesto v mojem življenju. S prijatelji smo večino prostih dni preživeli za računalniki in se zabavali ob prvih igrah. Takrat se nismo ukvarjali z mislijo o ustvarjanju iger in nismo vedeli, kako zelo zahtevna naloga je razvijanje iger. Desetletje kasneje je namen iger še vedno enak, igre in način ustvarjanja teh pa se je močno spremenil. Za igranje igre ni več potrebna polna soba opreme, ampak zadošča že navaden računalnik, ki ga najdemo v skoraj vsakem gospodinjstvu. Ustvarjanje iger je postalo enostavnejše s pojavom igralnih pogonov (angl. game engine). Sam termin računalniški pogon se je uveljavil v letih pred prelomom tisočletja, ko smo se prvič srečali s situacijo, kjer razvijalci iger niso razvili sami od začetka, ampak so nadgradili in spremenili že obstoječe igre. Na tej podlagi so nastali prvi igralni pogoni.

Unity, Unreal Engine, CryEngine in HeroEngine so le nekateri izmed veliko igralnih pogonov, ki obstajajo danes [39]. Pogosta težava, s katero se srečajo ustvarjalci pri uporabi igralnih pogonov je njihova cena, saj so zaradi nje pogosto nedostopni. Marca letos je Unreal Engine postal v celotni brezplačen in tako dostopen širši množici uporabnikov. Najbolj uporabljen igralni pogon [36]. Unity pa je z izdajo zadnje različice svojega programa občutno razširil vsebino, ki je v brezplačni različici in na tak način postal dostopnejši.

Pravijo, da je izbrati med različnimi igralnimi pogoni tako, kot bi se odločali med Coca-Colo in Pepsijem [17]. Da bi to lažje storili, smo v diplomski nalogi predstavili prednosti in slabosti dela v Unity in Unreal Engine, ki je prav tako eden najboljših in najbolj razširjenih igralnih pogonov [53].

Najprej smo se posvetili sami zgodovini iger, saj je pomembno razumeti, kako smo se znašli na točki, kjer smo. O obeh programih sem se najprej naučil osnove, kdo in kdaj ju je izdal, katere so očitne prednosti in slabosti. Najboljši način, da resnično spoznamo igralni pogon je seveda ta, da se lotimo ustvarjanje igre z njegovo pomočjo. Nobenega izmed igralnih pogonov prej nismo uporabljali, zato smo se z velikim veseljem in polni pričakovanja lotili ustvarjanja iger in tako je v Unity nastala igra Space Shooter, kjer z raketo uničujemo komete in poskušamo preleteti čim daljšo pot, v Unreal Engine pa Endless Running, kjer tečemo po neskončno dolgi progi, na naši poti pa se izogibamo oviram in pobiramo kovance. Dejansko ustvarjanje iger nam je pomagalo spoznati oba igralna pogona. Zaradi lažje primerjave smo v obeh igralnih pogonih ustvarili igro Busy Road, kjer se z avtomobilom umikamo kamnom na cesti in poskušamo prevoziti čim večjo razdaljo. Primerjali smo ju z različnih vidikov, za katere menimo, da so pomembni pri ustvarjanju iger – ustvarjanje objektov, grafika, dostop do izvirne kode,

razhroščevanje ipd. Na ta način smo zagotovili objektivnejši pogled, ki bo razvijalcem iger olajšal izbiro.

2 Zgodovina računalniških iger

Težko razumemo položaj v katerem smo, če ne poznamo poti, ki nas je tja pripeljala. Na tem mestu so kratko predstavljene spremembe v razvoju računalniških iger in njihov razvoj.

Spremembe v razvoju računalniških iger lahko v grobem razdelimo v več skupin [59].

a) *Spremembe na področju strojne opreme za igranje iger*

   Razvoj računalnikov in opreme je v zadnjih približno petdesetih letih doživel velik razcvet, kar je posledično pomenilo tudi velik napredek na področju iger. Od računalnikov z minimalno zmogljivostjo, spominom in grafiko smo prešli k hitrim in zmogljivim računalnikom, med katerimi so nekateri namenjeni posebej igranju iger.

b) *Spremembe na področju igralnih naprav*

   Razvoj je velik tudi na področju samih igralnih naprav. Prve igralne konzole so imele posebne vrtljive gumbe, majhno številko tipk ali posebne preproste igralne palice. Danes je situacija drugačna – igre lahko igramo z večimi igralnimi palicami hkrati in imamo veliko gumbov. Napredek je opazen tudi na področju zaznavanja gibanja telesa, kar seveda tudi močno vpliva na same računalniške igre.

c) *Spremembe na področju programske opreme, ki je namenjena ustvarjanju računalniških iger*

   Prvi programerji so napisali kodo v zbirnem jeziku (pogosto rečemo v strojni kodi) in so vsak piksel oblikovali sami. Danes imamo na voljo kopico programov, ki so ustvarjanje iger olajšali, saj so objekti že oblikovani in samo pisanje kode močno olajšano.

d) *Spremembe podjetij, ki se ukvarjajo z računalniškimi igrami*

   Sprva so računalniške igre ustvarjali posamezniki, danes pa na večini modernih iger delajo cela podjetja ali vsaj večje skupine posameznikov. Sredstva namenjena razvoju so iz nekaj tisoč dolarjev za igro narasla na več milijonov.

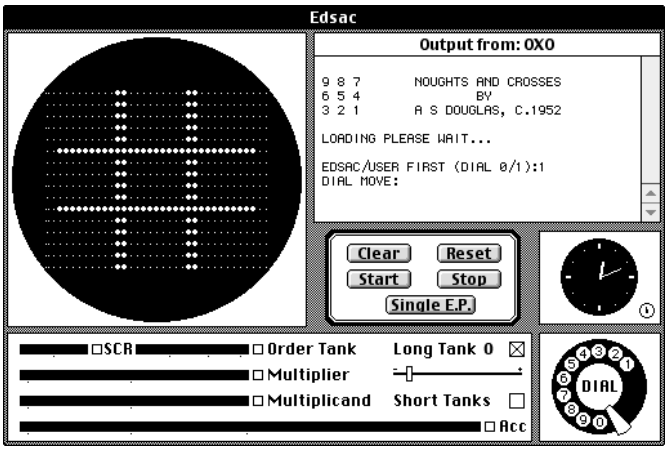
e) *Spremembe v starosti, spolu in ostalih lastnostih igralcev iger*

   Če so včasih računalniške igre igrali predvsem mlajši moški, danes to zagotovo ne drži več. Število ženskih igralk se približuje številu moških igralcev, starost pa ne predstavlja več nobene omejitve, saj imamo tako igre za otroke kot igre za odrasle.

f) *Spremembe v samem ustvarjanju in izgledu iger*

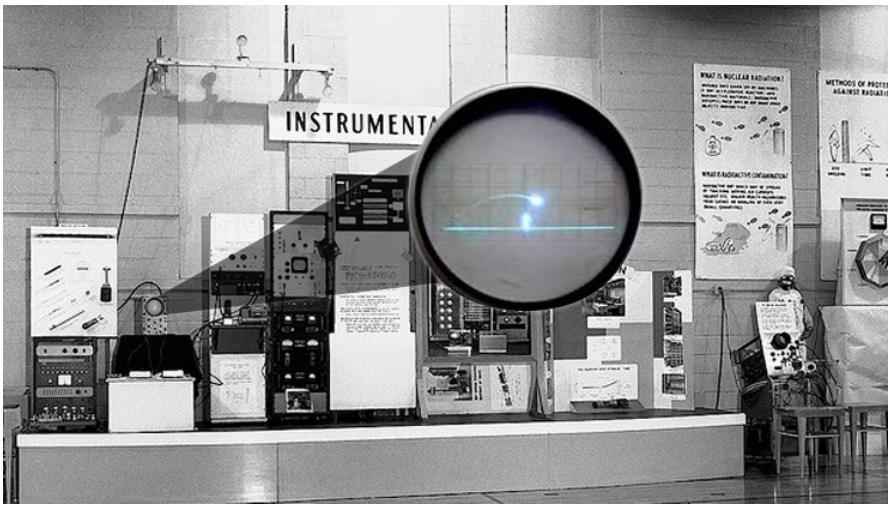
   Iz enostavnih dvodimenzionalnih iger smo prešli na zahtevnejše igre. Igre se prilagajajo lastnostim igralcev in postajajo vedno bolj dostopne prav vsakemu posamezniku.

Prvo računalniško igro je ustvaril Alexander Douglas leta 1952 in jo poimenoval OXO [32]. Njen izgled vidimo na sliki 1. Že iz samega imena lahko vidimo, da je bila to igra križec-krožec. Nastala je v sklopu njegovega diplomskega dela, ki ga je pisal na temo interakcije med ljudmi in računalniki.



Slika 1: Izgled prve računalniške igre [16].

Mnogi menijo, da je delo nuklearnega fizika Williama Higginbothama prva prava računalniška igra. Ta je leta 1958 ustvaril igro Tennis for Two, katere namen je bil poučiti igralce o vplivu gravitacije. Težava s prvimi igrami je bila ta, da je bilo zanje potrebne ogromno opreme, kar je bila ovira za spregled resničnih zmožnosti, ki so jih igre že takrat imele. Opremo potrebno za igro Tennis for Two vidimo na sliki 2.



Slika 2: Oprema za igro Tennis for Two [43].

V tem obdobju je bila ustvarjena tudi igra Spacewar!, ki jo je ustvaril študent MIT-a Steve Russell in je osnova za marsikatero kasnejšo igro [30]. V sedemdesetih se je prvič pojavilo razmišljanje o komercializaciji iger in desetletje, ki je sledilo, mnogi označujejo za zlato dobo arkadnih iger. Prva popularna arkadna igra je nastala leta 1972 v podjetju Atari [7]. Imenovala se je Pong[1] in je bila velik uspeh, k čemur priča tudi to, da se prva arkadna naprava ni pokvarila zaradi slabega delovanja mehanizacije same naprave, ampak ker je bilo v njej preveč kovancev. Izgled igralne naprave in same igre prikazuje slika 3.

Slika 3: Prva igralna naprava Pong in izgled Pong igrice.

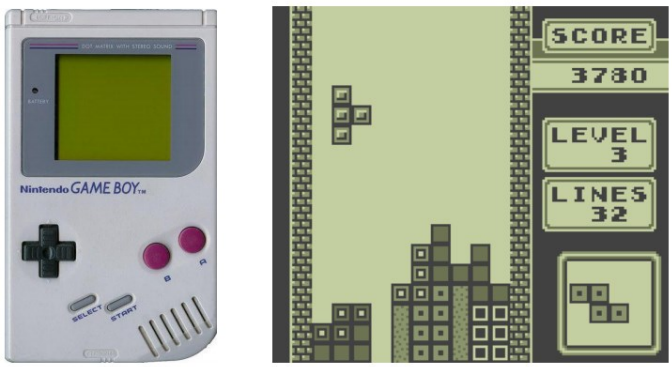
Kmalu so podjetju Atari sledili tudi drugi in nastalo je mnogo novih zanimivih iger. Igra Space Wars je tista, kjer so prvič uporabili vektorsko grafiko. Leta 1978 se pojavi prva barvna igrica Space Invaders. 1980 je leto, ko je bila ustvarjena še danes popularna igra Pac-Man [7].

V osemdesetih so se pojavile konzole, ki so imele možnost povezave z televizorjem. Prva je bila Odyssey in je bila prodana v več kot 100.000 izvodih. Pomembnejša med konzolami je bila konzola podjetja Atari, ki se je imenovala Atari 2600. Vidimo jo lahko na sliki 4. Ko so konzoli priložili še igro Space Invaders, so dosegli rekordno prodajo; prodanih je bilo več kot 30 milijonov konzol, najbolj igrana igra na tej konzoli pa je bil Pac-Man. Konzola je imela le en kilobajt spomina, kar je oteževalo razvoj zahtevnejših iger.

[1] http://www.ponggame.org/

Slika 4: Igralna konzola Atari 2600 [42].

V devetdesetih se je pojavilo mnogo podobnih iger in konzol. Dotlej vodilni Atari je zaradi slabih investicij izgubil svoj položaj na trgu. Na trg računalniških iger je močno vplival tudi pojav računalnikov za domačo rabo. Ti so z več spomina, grafike in možnostjo zvoka ponujali boljšo izkušnjo igranja iger kot konzole. Pomemben napredek je bil predvsem to, da so računalniki omogoči shranjevanje napredka in igre ni bilo potrebno začeti vedno znova. Diskete so omogočile lahko shranjevanje in prenašanje iger. Na novo oblikovan trg je omogočil vzpon novih podjetij, med katerimi je še zlasti uspelo podjetje Nintendo, ki je leta 1985 predstavilo svoj Nintendo Entertainment System (NES) [20], ki je bil dan na trg skupaj z igrico Super Mario Bros in je bil takojšen velik uspeh. NES je imel namesto igralne palice ali obračajočih se gumbov smerne tipke in je popolnoma spremenil način igranja iger. Nintendo je svoj položaj na trgu leta 1989 še dodatno okrepil, ko je izdal prvi ročni igralni sistem – The Game Boy. Tetris [10] je bila prednaložena igra in skupaj sta predstavljala zmagovalno kombinacijo. Sliko prvega Game Boy-a in Tetrisa prikazuje slika 5.



Slika 5: The Game Boy in originalna različica Tetrisa.

Ob prelomu tisočletja sta bila najpomembnejša igralca na trgu še vedno Nintendo in Sega, ki je Nintendovim izumom vztrajno sledila. Prehitela ga je na področju konzol, ko je prva izdala 16-bitni sistem imenovan Genesis [38] z hitrejšim procesorjem in 64 kilobajti spomina. Zaslon je imel mnogo boljšo grafiko, ki je omogočila, da je Saga predstavila nov lik in novo igro Sonic The Hedgedog. Med letoma 1994 in 1996 se je pojavila nova generacija konzol in takrat se je trgu pridružil tudi Sony s svojim prvim PlayStationom [37]. Procesorji so postali 32 ali 64 bitni in so imeli že do 4 megabajte spomina. Strojna oprema je omogočala 3D grafiko, vendar v zelo omejeni obliki. PlayStation, ki ga lahko vidimo na sliki 6, je doživel velik razmah predvsem zato, ker je bilo enostavno programirati igre zanj.

Slika 6: Prvi PlayStation [46].

V tem času so tudi računalniki postali zmogljivejši. Grafika je bila boljša, imeli so več spomina, trdi diski so omogočali shranjevanje podatkov igric in imeli so hitrejše procesorje. Prednost je bila tudi v tem, da so omogočali predvajanje glasbe in videov z CD-jev. Pojavile so se igrice, ki so delovale na principu Full Motion Video (FMV) in so igralcu omogočale okoljske zvoke in polnejše doživetje ob igranju. Prednost računalnikov je tudi v uporabi tipkovnice in miške, saj so zato lahko nastale igre, kjer je bilo potrebno označevati točne na zaslonu in igre, kjer je bilo potrebnih veliko različnih ukazov. Tipkovnice so bile nujne za različne simulatorje letenja. Mnogi računalniki so imeli modeme, kar je omogočilo igranje iger med igralci na različnih lokacijah in je bilo podlaga za vzpon množičnih spletnih iger z večimi igralci, ki igrajo domišljijske vloge. (angl. massive multiplayer role playing online games) (MMORPG).

Slabost računalnikov je bila ta, da je bila namestitev iger precej zahtevna, saj je večina uporabljala MS-DOS. Ta težava je bila odpravljena leta 1995, ko se je pojavil Windows 95, kar je imelo pozitiven vpliv na razvoj iger za okolje Windows.

Pojavile so se težnje po ustvarjanju 3D iger, vendar večina računalnikov ni imela ustreznih komponent. Programerji so težavo rešili tako, da so prikazali navidezni 3D svet – med prvimi in najuspešnejšimi sta bila John Carmack in John Romero, ki sta leta 1993 ustvarila igro Doom [8], ki jo mnogi smatrajo za prvo prvoosebno strelsko igro, čeprav v resnici to ni. Ko so 3D grafične kartice postale standardni del računalnikov, so bile težave na tem področju manjše, vendar še ne popolnoma odpravljene, saj je bilo programiranje za toliko različnih grafičnih kartic velik zalogaj.

Pojavile so se mnoge spremembe tudi na področju konzol in prenosnih igralnih sistemov (npr. Nintendo je leta 1995 izdal barvni Game Boy), vendar se osredotočimo predvsem na spremembe na področju računalniških iger, ki jih je ob bilo kar nekaj prelomu tisočletja.

Še vedno so igre na računalnikih spremljale stare težave – programerji so imeli težave z ustvarjanjem iger zaradi široke palete različnih grafičnih kartic, različnih hitrosti procesorjev, pomanjkanja spomina in ostalih komponent. Težko je namreč izdelati igro, ki bo delovala na vseh različno zgrajenih in različno zmogljivih računalnikih. Pred programerji je bila težka odločitev, odločiti so se morali ali želijo narediti igro, ki bo delovala le na najboljših računalnikih ali igro, ki bo delovala na večini srednje dobrih. Večina iger (recimo Age of Empires [1] in Command & Conquer [4]) se je odločila za slednjo verzijo in tako dobila več igralcev kot recimo Crysis [6], ki je deloval le na računalnikih z dobro grafično kartico in je tako dosegel le najbolj zagrizene igralce.

Težava se je pojavila tudi zaradi enostavnega prepisovanja iger in piratstva. Oboje je povzročilo upad zaslužka pri računalniških igrah. Pomembno mesto med igrami ima igra The Sims, ki je bila prvič izdana leta 2000. Izgled igre je prikazan na sliki 7. Predstavljala je popolnoma nov način igranja iger in je bila sprva skoraj zavrnjena, saj se je podjetju zdelo, da tak način igranja ni Zabaven. Njihovo zmoto dokazuje dejstvo, da je kljub piratskim verzijam prodanih več kot 75 milijonov kopij.

Razmahnile so se tudi MMORPG igre. Najpopularnejša igra je World of Warcraft, ki jo lahko vidimo na sliki 7 [57]. Ob koncu leta 2010 je imela več kot dvanajst milijonov igralcev [50].

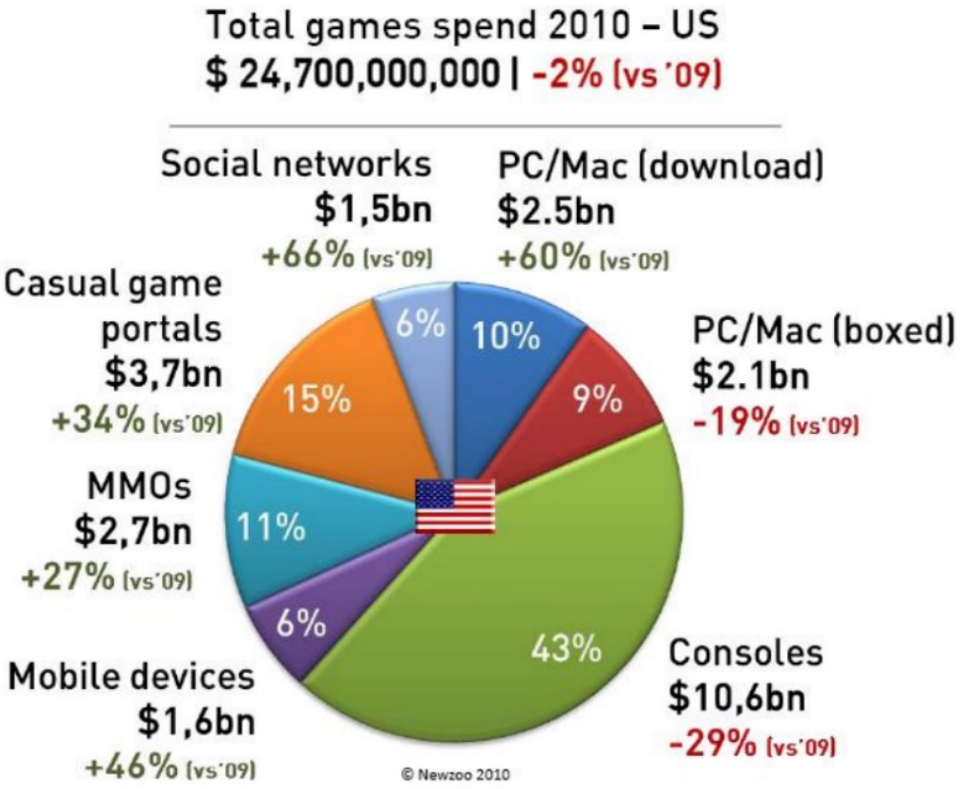
Slika 7: Dve izmed najbolje prodajanih računalniških igric The Sims (levo) in World of Warcraft (desno).

Ob prelomu tisočletja so se pojavile tudi tako imenovane vsakodnevne/navadne igre (angl. casual games). Vedno več ljudi je imelo doma računalnike z internetno povezavo in želeli so igrati preproste, hitre in kratke igre. Te igre so pogosto napisane v Flashu in so igrane kar preko internetnega brskalnika. Eden izmed tipičnih in hkrati zelo uspešnih primerov je Bejeweled [2], ki se je pojavila leta 2001. Z pojavom socialnih omrežij so se pojavile tudi igre, ki so igrane na socialnih omrežjih in s pomočjo prijateljev na socialnih omrežjih. Najbolj tipičen primer je Farmwille [9], ki ima več kot 80 milijonov aktivnih igralcev [49].

Le omenimo naj, da so ljudje igre začeli igrati tudi na mobilnih telefonih, zaradi česar so se razvile popolnoma nove vrste iger. Tudi tukaj so imeli programerji težave zaradi različnih vrst telefonov in včasih je bilo potrebnih več sto različic iste igre. Področje mobilnih iger je doživelo razmah predvsem s pojavom pametnih telefonov, ki so popolnoma spremenili svet mobilnih iger. Dodaten napredek na tem področju je pomenil tudi pojav tabličnih računalnikov.

Svet računalniških iger se je začel enostavno in v črno beli grafiki, danes pa je eno najbolj zahtevnih in razvitih področij, kar jih poznamo. Igralna industrija je ogromna in ljudje vlagajo vedno več časa in denarja v igre.

Spodnja slika prikazuje porabo sredstev v različnih kategorijah za leto 2010 v Združenih državah Amerike. Vidimo lahko ogromen porast iger igranih na socialnih omrežjih in mobilnih telefonih. Vedno več iger je kupljenih preko interneta in naloženih na računalnike, kot kupljenih v specializiranih trgovinah. Velik upad je tudi na področju iger za konzole [12].



Slika 8: Poraba sredstev glede na različne kategorije iger [1].

3 Igralni pogoni

Igralni pogon (angl. game engine) je programska oprema, ki je namenjena lažjemu razvoju in ustvarjanju iger. Pogon skrbi za osnovne funkcije igre, kot so uporabljanje grafike, predvajanje zvoka, podpora simulaciji fizikalnega obnašanja objektov, podpora animacijam in podobno. Prednost je v tem, da razvijalcem ni potrebno skrbeti za osnovne funkcionalnosti, kot je recimo risanje objektov, in lahko svoj trud vložijo v edinstvene in zahtevnejše dele svoje igre [15]. Igralnih pogonov je danes na voljo mnogo, vse več je tudi brezplačnih, na katere se bomo osredotočili v tej diplomski nalogi. CryEngine 3 [5], Source 2 [48], HeroEngine [13], Unity in Unreal Engine so le nekateri izmed njih. Naša primerjava poteka med pogonom Unity in Unreal Engine. Unity smo izbrali zato, ker je najbolj uporabljen igralni pogon [36]. Unreal Engine pa je del primerjave zato, ker je eden najbolj razširjenih pogonov in omogoča ustvarjanje iger z čudovito grafiko; pogosto je uporabljen za ustvarjanje resničnostnih fotografij [52].

3.1 Igralni pogon Unity

Slika 9: Logotip pogona Unity.

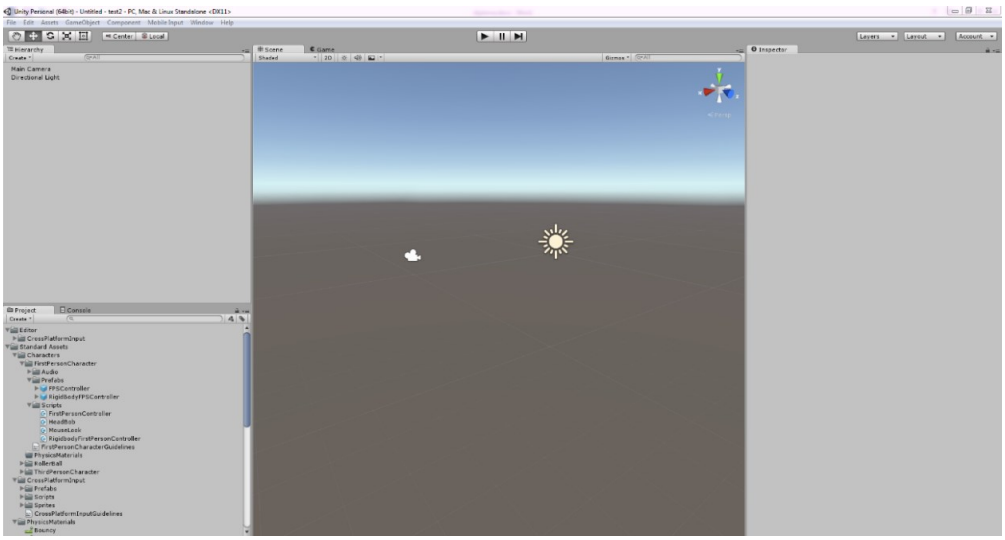
Unity, katerega logotip prikazuje slika 9, je igralni pogon, ki ga je podjetje Unity Technologies prvič predstavilo javnosti leta 2005 na Applovi konferenci. Prva v Unity narejena igra je GooBall [23]. Takrat je Unity omogočal le izdelavo iger za OS X, danes pa se je področje njegove uporabe močno razširilo, saj obvladuje kar 45% trga in ga uporablja skoraj vsak drugi razvijalec računalniških iger. Igre, ki so ustvarjene s pomočjo Unity igra več kot 600 milijonov igralcev, na več kot 20ih [22] različnih platformah, ki jih lahko ločimo na več kategorij:

a) mobilne platforme: iOS, Android, Windows Phone 8, BlackBerry 10 in Tizen [51],
b) namizne platforme: Windows, Mac in Linux,
c) igralne konzole: Wii [56], Xbox [58] in PlayStation [29],
d) spletne platforme: vsi internetni brskalniki in WebGL,
e) navidezna resničnost: Oculus Rift [21], Gear VR [11] in Microsoft Hololens [19].

Unity se pojavlja v dveh različicah in sicer kot Unity Pro in Unity Personal. Slednji je popolnoma brezplačen za uporabo tako za posameznike, kot za podjetja z manj kot 100,000$ letnega bruto prihodka. Od marca 2015 je Unity Personal popolnoma brezplačen in nam že brezplačna različica omogoča skoraj vse, za kar je sicer potrebno pri Unity Pro odšteti 75$ na mesec ali pa 1500$, v kolikor se odločimo za enkratno plačilo. Razlike med brezplačno in plačljivo verzijo so predvsem na področju dodatnih funkcij, kjer so uporabniki brezplačne različice prikrajšani le za možnost spremembe barve v urejevalniku iz bele v črno, igre ustvarjene v brezplačni različici pa imajo vedno ob zagonu prikazan logotip pogona Unity. Ostale plačljive funkcije so namenjene predvsem olajšavi dela in lahko tudi brez njih Unity popolnoma brez težav uporabljamo [35]. Za delo v tej diplomski nalogi je bil uporabljen Unity 5.1.2.

Pri programiranju v Unity nismo omejeni le na programski jezik, ki je vgrajen v program, ampak lahko za delo uporabljamo JavaScript, C# ali Boo [3]. Dejstvo, da so vsi trije jeziki skriptni, omogoča prilagodljivo načrtovanje, hitro izgradnjo in hitre iteracije [24].

Program si namestimo in že je pripravljen za uporabo. Za delo z pogonom Unity je potrebno ustvariti račun, s pomočjo katerega se kasneje vpišemo v program in dostopamo do svojega dela. Pri ustvarjanju projekta si izberemo, koliko dimenzionalno igro bomo ustvarili. Sam čas ustvarjanja projekta je povezan s tem, koliko elementov smo izbrali na začetku: več elementov pomeni daljši čas. Ko je projekt pripravljen je začetni igralni prostor popolnoma prazen, imamo le prazen prostor in tudi najbolj začetne in osnovne elemente postavimo sami. Prikaz osnovnega prostora lahko vidimo na sliki 10.



Slika 10: Osnovni prostor v pogonu Unity.

V pomoč pri delu nam je spletna trgovina, v kateri so na voljo različne teksture, objekti, efekti in podobno.

3.2 Igralni pogon Unreal Engine

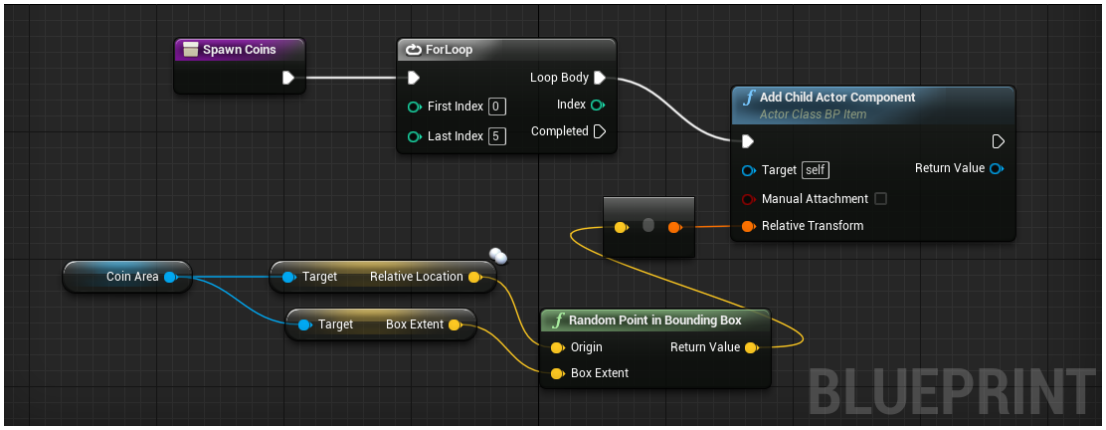
Slika 11: Logotip pogona Unreal Engine.

Unreal Engine, ki ima logotip kroga s črko U (vidimo ga na sliki 11) smo prvič spoznali leta 1998, ko je podjetje Epic Games predstavilo igro Unreal. Unreal je igra, ki ima prvoosebnega strelca in sprva se je Unreal Engine uporabljal predvsem za ustvarjanje tovrstnih iger. Nastal je z namenom ustvarjanja 3D iger in je znan po tem, da omogoča ustvarjanje iger z osupljivo grafiko, katere primer vidimo na sliki številka 12 [25].

Slika 12: Prikaz grafičnih zmožnosti Unreal Engine [47].

Od prvega dne je bilo izdanih več različic. Najnovejša je Unreal Engine 4, ki je bila izdana maja 2012. Kljub temu, da so od izdaje minila že tri leta, Epic Games še vedno aktivno delajo na programu in izdajajo nove različice in nadgradnje k obstoječemu programu. Za delo v tej diplomski nalogi je bil uporabljen Unreal Engine 4.8.3. Cena programa je bila 19$ na mesec, od marca 2015 pa je program popolnoma brezplačen. Omogočen je dostop do izvirne kode in vseh funkcij, ki jih program nudi. Edina obveznosti uporabnika je ta, plačuje avtorski honorar v primeru, ko z igro zasluži vsaj 3000$ v enem trimesečju. [26]

Unreal Engine je poseben tudi zaradi načina ustvarjanja iger. Za delo lahko uporabljamo programski jezik C++, možno pa je ustvariti celotno igro brez ene same vrstice kode, saj omogoča ustvarjanje iger s pomočjo posebnih diagramov (angl. blueprint). Primer diagrama vidimo na sliki 13. Diagrami so velika prednost za uporabnike, ki niso vešči programiranja, imajo pa tudi svoje pomanjkljivosti, ki se kažejo predvsem pri ustvarjanju zahtevnejših iger. Pri teh je potrebna natančnost, ki je diagrami ne omogočajo.



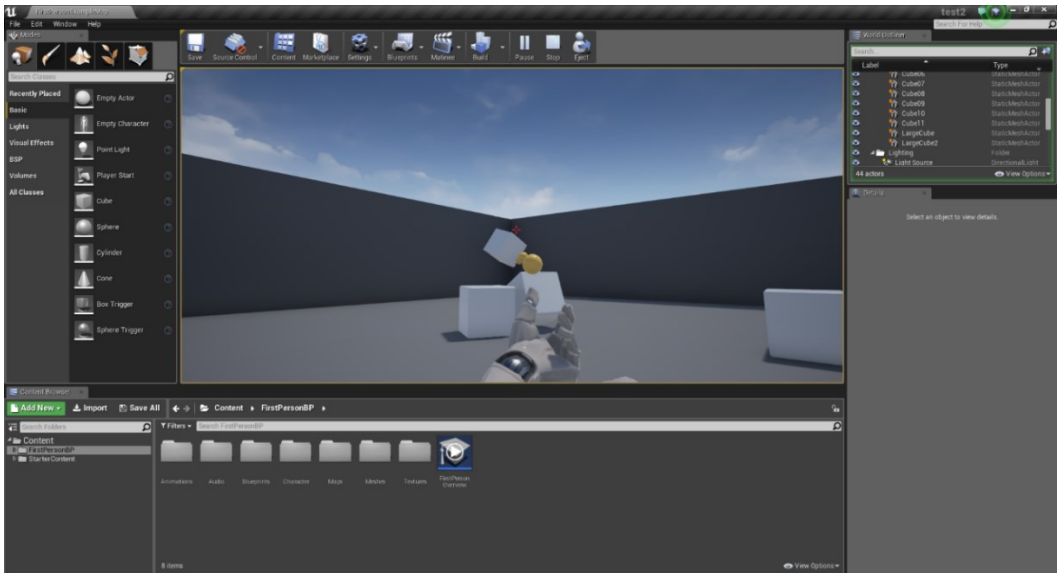
Slika 13: Primer ustvarjanje iger s pomočjo diagrama.

Pogon omogoča ustvarjanje iger za različne platforme, vendar je znan po tem, da podpira le najnovejše različice teh platform. Pri namiznih platformah omogoča ustvarjanje iger za Microsoft Windows, Linux in OS X. Pri igralnih konzolah sta podprta PlayStation 4 [29] in Xbox one [58], podprti mobilni platformi sta Android in iOS. Možno je tudi ustvarjanje iger za naprave, ki ustvarjajo navidezno resničnost kot so Oculus Rift [21], Ouya [28] in HTC Vive[14]. Pozabili niso niti na možnost ustvarjanja internetnih iger [27].

Pogon namestimo tako, da si najprej prenesemo sprožilnik (angl. launcher) znotraj katerega imamo dostop do pogona, spletne trgovine, nadgradenj, posodobitev in podobnega. Sprožilnik

nam tudi omogoča, da imamo hkrati naloženih več verzij programa. Ima to prednost, da na enem mestu najdemo vse povezano s programom, vendar pa prinaša tudi slabosti, med drugim to, da je zagon daljši in naš računalnik bolj obremenjen.

Ko se odločimo za ustvarjanje novega projekta v pogonu Unreal Engine nam program takoj ponudi predloge iger, glede na to, kakšno vrsto igre bomo ustvarjali. Kako izgleda začetni osnovni prostor lahko vidimo na sliki 14. Izberemo tudi način ustvarjanja projekta, kjer se odločimo med programskim jezikom C++ ali izdelovanjem diagramov. Določimo platformo, za katero bomo ustvarjali igro. Okno za določitev osnovnih izbir je jasno in pregledno, zato se tudi nepoznavalec brez težav znajde. Po določenih izbirah se odpre začetno igralno polje, kjer je osnova za igro že pripravljena. Primer je ustvarjanje igre, kjer streljamo: tam nas program postavi v mapo z objekti in nam v roko da pištolo. Omenjen primer vidimo na sliki 14. Hkrati so znotraj projekta samodejno ustvarjane mape, povezane s projektom. Vidimo dosedanjo kodo in kakšen je razpored elementov. Oboje je velika prednost in pomoč pri nadaljnjem delu.



Slika 14: Prikaz osnovnega prostora v Unreal Enginu.

V pomoč pri delu nam je v pomoč tudi spletna trgovina.

4 Ustvarjanje iger

Program najbolje spoznamo tako, da se sami lotimo dela z njim. Zaradi pomanjkanja znanja smo najprej ustvarili igri s pomočjo vodenih videov, ki so namenjeni začetnikom in njihovemu prvemu srečanju s pogonom. Tako sta nastali igri Space Shooter v pogonu Unity, kjer se igralec z raketo izogiba kometom in jih uničuje. V pogonu Unreal Engine smo ustvarili igro Endless Running, katere cilj je preteči čim dlje in pobrati čim več kovancev. Na poti se moramo izogibati oviram, preskakovati luknje in zavijati.

Začetnemu spoznavanju je sledilo ustvarjanje lastne igre Busy Road. Igralec vozi avto, se izogiba drugim avtom in poskuša prevoziti čim dlje. Igro dela zahtevnejšo dejstvo, da se igralčev avto premika vedno hitreje. Igro Busy Road smo ustvarili v obeh igralnih pogonih in se tako podrobneje seznanili z njima.

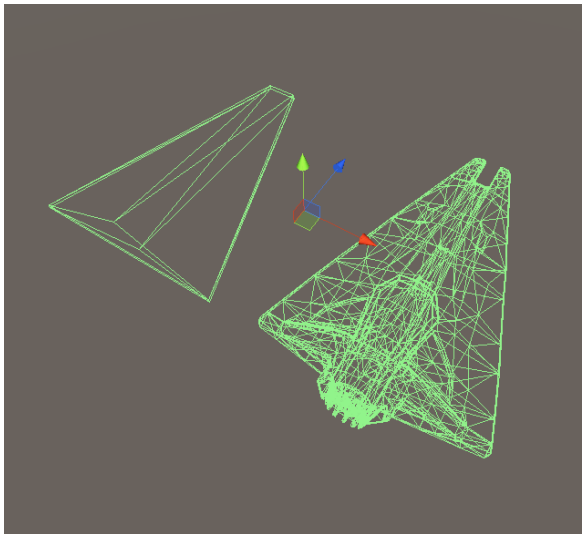
Za avtomobilsko igro smo se odločiti zato, ker lahko pri njenem ustvarjanju uporabimo mnogo različnih elementov, hkrati pa je ustvarjanje dovolj enostavno za začetnika. Igra je zanimiva, prednost je dejstvo, da je ne moremo dokončati. Vedno se lahko trudimo za boljši rezultat in jo tako igramo dalje.

4.1 Razvoj igre Space Shooter v pogonu Unity

Najprej nas video[2] seznani s tem, kako ustvariti projekt. Video je namenjen popolnim začetnikom in z njegovo pomočjo lahko vsak še tako neizkušen uporabnik razume osnovne pojme in ustvari svojo prvo igro. Začnemo s prenosom potrebnih sredstev (angl. assets) (objekti, teksture, zvoki, …) in uvozom teh sredstev v pogon Unity. Na tak način se nam olajša delo, saj ni potrebna izdelava teh objektov in se lahko osredotočimo takoj na izdelavo igre. Če bi začeli popolnoma nov svoj projekt bi ta sredstva morali ustvariti sami ali pa bi jih morali kupiti preko spletne trgovine. Na tej točki izberemo tudi platformo, za katero igro izdelujemo. Space Shooter je namenjena delovanju v internetnem brskalniku. Ko nastavimo osnovne nastavitve je čas za postavljanje objektov. Najprej iz mape sredstev v polje prenesemo model rakete, ki bo naš osnovni in najpomembnejši element. Želimo doseči, da komet ob dotiku rakete raznese. To dosežemo tako, da modelu rakete dodamo trkalnik (angl. collider). Tako naša raketa dobi obrobo, na kateri se za vsako sliko sproti preveri, če je prišlo do trka. Obrobo ustvari program tako, da sledi vsem linijam na objektu in na tak način ustvari obrobo. Takšna obroba je za naše potrebe prezapletena, saj mora računalnik sproti preračunati, če je prišlo do trka za vsako izmed linij. Stvar rešimo tako, da sami ustvarimo enostavnejšo obrobo objekta, ki ima le nekaj linij.

[2] https://unity3d.com/learn/tutorials/projects/space-shooter-tutorial

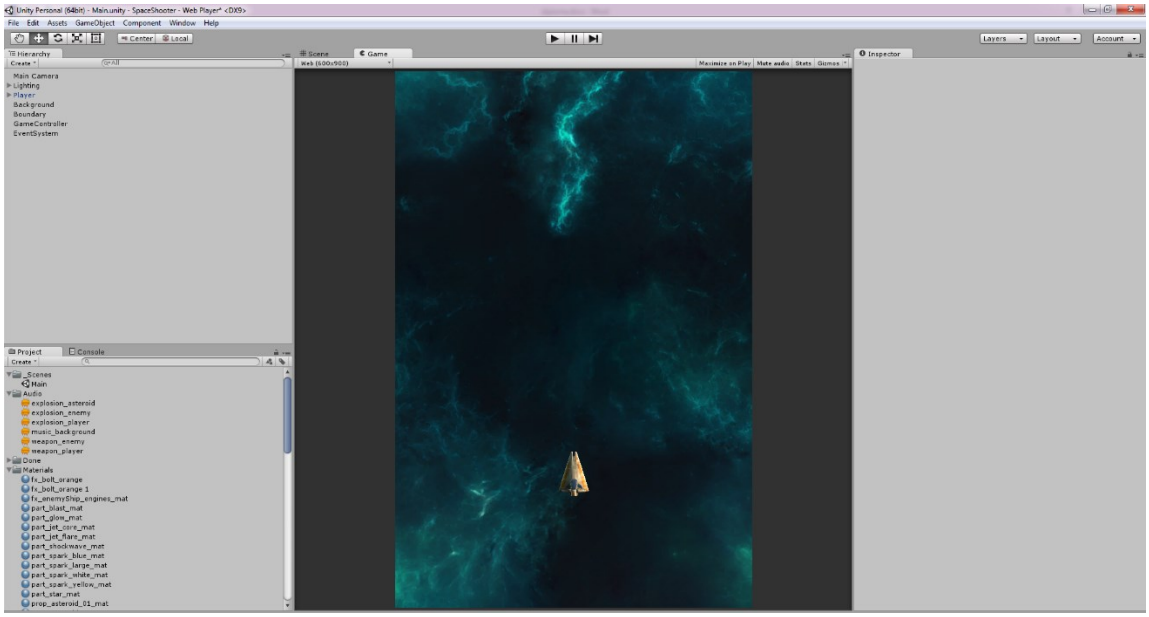
Na tak način je izračun dosti lažji in hitrejši. Težavo bi lahko rešili tudi tako, da bi preko našega objekta postavili obrobo kocke ali podobnega telesa. V tem primeru se pojavi druga težava, saj bi s tem izgubili na natančnosti. Naša prva rešitev je idealna, saj ohranimo pravo mero natančnosti (sledimo namreč obliki objekta), vendar ne pretiravamo s podrobnostmi obrobe. Primerjavo med obrobama prikazuje slika 15.



Slika 15:Primerjava enostavne in zahtevnejše obrobe objekta

Ker bo igra igrana s ptičje perspektive, je potrebno pravilno postaviti kamero. V izhodiščem položaju je kamera zadaj in malo nad raketo, mi pa jo želimo nad raketo. Kamero prestavimo s potegom. Sedaj moramo samo še prestaviti raketo na dno našega zaslona, saj bo tam najboljša izhodiščna pozicija. Tudi urejanje pozicij elementa je preprosto, saj ga enostavno premikamo po x, y in z osi.

Pomembna je tudi ureditev osvetlitve. Začnemo tako, da ozadje spremenimo v črno in izklopimo ambientalno osvetlitev, ki je prisotna že od začetka. Odstranimo jo zato, da ne moti naše osvetlitve. Urejevanje svetlobe je v pogonu Unity dokaj enostavno, saj jo uredimo s premikanjem kurzorjev in premikanjem vira svetlobe. V igri želimo, da svetloba prihaja od zgoraj desno, saj tako ustvarimo vtis vožnje proti soncu. Ko smo zadovoljni s pozicijo vseh luči dodamo še ozadje. Tega izberemo iz mape in potegnemo na polje. Postavimo ga daleč v ozadje, da ne moti ostalih objektov. Izgled igre na tej točki nazorno pokaže slika 16.



Slika 16: Objekt na željenem položaju in z željeno osvetlitvijo.

Sledi premikanje rakete ob pritisku smernih tipk. To dosežemo tako, da objektu dodamo skripto, ki preverja za pritisk smernih tipk. Potrebne skripte napišemo v programskem jeziku, ki ga sami izberemo. Odločili smo se za C#. Del kode pa izgleda tako:

```
void FixedUpdate () {
    float moveHorizontal = Input.GetAxis ("Horizontal");
    float moveVertical = Input.GetAxis ("Vertical");

    Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
    rigidbody.velocity = movement * speed;

    rigidbody.position = new Vector3 (
        Mathf.Clamp (rigidbody.position.x, boundary.xMin, boundary.xMax),
        0.0f,
        Mathf.Clamp (rigidbody.position.z, boundary.zMin, boundary.zMax)
    );

    rigidbody.rotation = Quaternion.Euler (0.0f, 0.0f, rigidbody.velocity.x
* -tilt);
}
```

Kadar pride do pritiska, se raketa premakne v željeno stran za določeno število enot. Pomembno je, da postavimo meje znotraj katerih se lahko premika. Gibanju rakete dodamo še nagib, raketo nagnemo v desno, ko se premika desno in levo, ko se premika levo. Samo dodajanje nagiba ima zgolj estetski učinek. Nagib je lažje urejati zaradi funkcije, ki omogoča lahke prehode, saj ne želimo, da se nagib takoj spremeni iz 0° v 45°.

Naša raketa potrebuje možnost streljanja. Začnemo tako, da najprej ustvarimo nov objekt, kar je možno storiti tudi s kratico CRTL+SHIFT+N. Kratica je praktična, saj nam omogoča hitro delo s programom. Ustvarimo nov štirikoten objekt, na katerega damo teksturo naboja, ki ga imamo že pripravljenega. Glede na to, da je naboj na črnem ozadju, je potrebno spremeniti senco na objektu (angl. shader) in na tak način črno spremeniti v prozorno. Naboj potrebuje trkalnik, saj je pomembno, kdaj zadane komet. Na tej točki ponovimo postopek obrisovanja objektov. Tukaj je naše delo dosti lažje, saj okrog objekta naboja postavimo valj. Taka rešitev je hitra in učinkovita. Naboju dodamo hitrost s spremenljivko s pomočjo spodnje kode:

```
public class Mover : MonoBehaviour {
    public float speed;

    void Start () {
        rigidbody.velocity = transform.forward * speed;
    }
}
```

Nabojev bomo imeli več, zato ga shranimo. Na tak način zagotovimo, da so vsi naboji enaki. Naslednji korak je izstrelitev nabojev z lokacije rakete ob pritisku na tipko. Lokacijo nastavimo pred samo raketo, da ustvarimo realističen videz. Streljanje omogočimo s skripto in na tak način zagotovimo, da se ob vsakem pritisku na tipko ustvari nov objekt naboja, ki se premika z določeno hitrostjo. Pomembno je tudi, da je med posameznimi izstreljenimi naboji zamik.

```
void Update () {
    if (Input.GetButton("Fire1") && Time.time > nextFire)    {
        nextFire = Time.time + fireRate;
        Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
    }
}

void FixedUpdate () {
    float moveHorizontal = Input.GetAxis ("Horizontal");
    float moveVertical = Input.GetAxis ("Vertical");

    Vector3 movement = new Vector3 (moveHorizontal, 0.0f, moveVertical);
    rigidbody.velocity = movement * speed;

    rigidbody.position = new Vector3 (
        Mathf.Clamp (rigidbody.position.x, boundary.xMin, boundary.xMax),
        0.0f,
        Mathf.Clamp (rigidbody.position.z, boundary.zMin, boundary.zMax)
    );

    rigidbody.rotation = Quaternion.Euler (0.0f, 0.0f, rigidbody.velocity.x
* -tilt);
}
```

Pojavi se problem, saj pri streljanju naboji potujejo v neskončnost. To rešimo tako, da igri postavimo robove. Okrog igralne površine postavimo kocko, kateri dodamo trkalnik. S skripto dosežemo, da se vsak objekt, ki zapusti trkalnik, uniči.

Sedaj potrebujemo samo še nekaj, kar bodo naši naboji uničevali. Ponovno ustvarimo nov objekt, tokrat komet. Model kometa že imamo, zato je naše delo olajšano. Znajdemo se na že znani točki, ko moramo kometu dodati trkalnik. Postopek poenostavimo tako, da okrog kometa postavimo valj, saj sta obliki zelo podobni. Kometu dodamo še hitrost in možnost premikanja proti igralcu. Naslednji korak je zapis skripte, s katero poskrbimo, da se ob stiku naboja in kometa oba uničita. Ta skripta preverja ali je prišlo do stika med trkalnikom naboja in trkalnikom kometa.

```
void OnTriggerEnter(Collider other) {
    if (other.tag == "Boundary") {
        return;
    }
    Destroy(other.gameObject);
    Destroy(gameObject);
}
```

Z željo, da igra izgleda čim bolj realistično in estetsko, dodamo kometom naključno rotacijo in efekt eksplozije ob uničenju kometa in igralca.

Sedaj smo na točki, ko imamo vse potrebne elemente, za našo igro. Potrebujejo je še nekaj, kar jih bo združilo v celoto. Zato ustvarimo novo skripto, ki bo predstavljala krmilnik za igro. Naloga krmilnika za igro je ta, da naključno izbere lokacijo, kjer nastane komet in na tak način ustvarja valove kometov. S tem je igra skoraj končana in potrebuje le še zvok, ki igra v ozadju, zvok ob izstrelitvi naboja in zvok ob eksploziji. Pomembno je, da je glasnost zvokov usklajena, pri tem se osredotočimo predvsem na to, da je zvok v ozadju tišji od zvokov za zvočne efekte. Zvok dodamo tako, da ga preprosto potegnemo v objekt, tudi ostale nastavitve zvokov so zelo preproste. V našem primeru so zvoki že bili na voljo. Potrebno jih je bilo samo še uvoziti in dodati objektom.

```
void Update () {
    if (Input.GetButton("Fire1") && Time.time > nextFire) {
        nextFire = Time.time + fireRate;
        Instantiate(shot, shotSpawn.position, shotSpawn.rotation);
        audio.Play ();
    }
}
```

Igra seveda potrebuje štetje točk, saj lahko na tak način tekmujemo. Točke igralcu dodelimo za vsak uničen komet, stanje točk pa je prikazano v zgornjem levem kotu. Prikaz omogočimo tako, da ustvarimo nov objekt imenovan risalna površina (angl. canvas). Ustvarjanje tega objekta je enostavno; potrebna sta le dva klika. V ta objekt lahko dodamo razne stvari, v našem primeru je to besedilo. Dodeljevanje točk ustvarimo tako, da ob vsakem trku v komet povečamo globalno spremenljivko za določeno število. Igro med igranjem lahko vidimo na sliki 17. V zgornjem levem kotu vidimo tudi prikaz točk.

Slika 17: Space Shooter med igranjem. Vidimo lahko eksplozijo kometa.

Igra se konča, ko igralčevo raketo zadane komet. Takrat pride do eksplozije in napisa `Game Over` na sredini zaslona. Igra je popolnoma zaključena, ko z igralne površine pade še zadnji komet. Takrat se nam v zgornjem desnem kotu pojavi možnost ponovnega igranja.

Naša igra je dobila končno obliko. Sedaj jo moramo samo še dokončno zgraditi. Takšno lahko zaženemo brez pogona Unity kar v internetnem brskalniku [55].

4.2 Razvoj igre Engless Running v pogonu Unreal Engine

Ustvarili bomo igro, kjer igralec teče v neskončno dolgo naravnost, pri teku pa se izogiba oviram in pobira kovance. Cilj igre je pobrati čim več kovancev in priteči čim dlje[3].

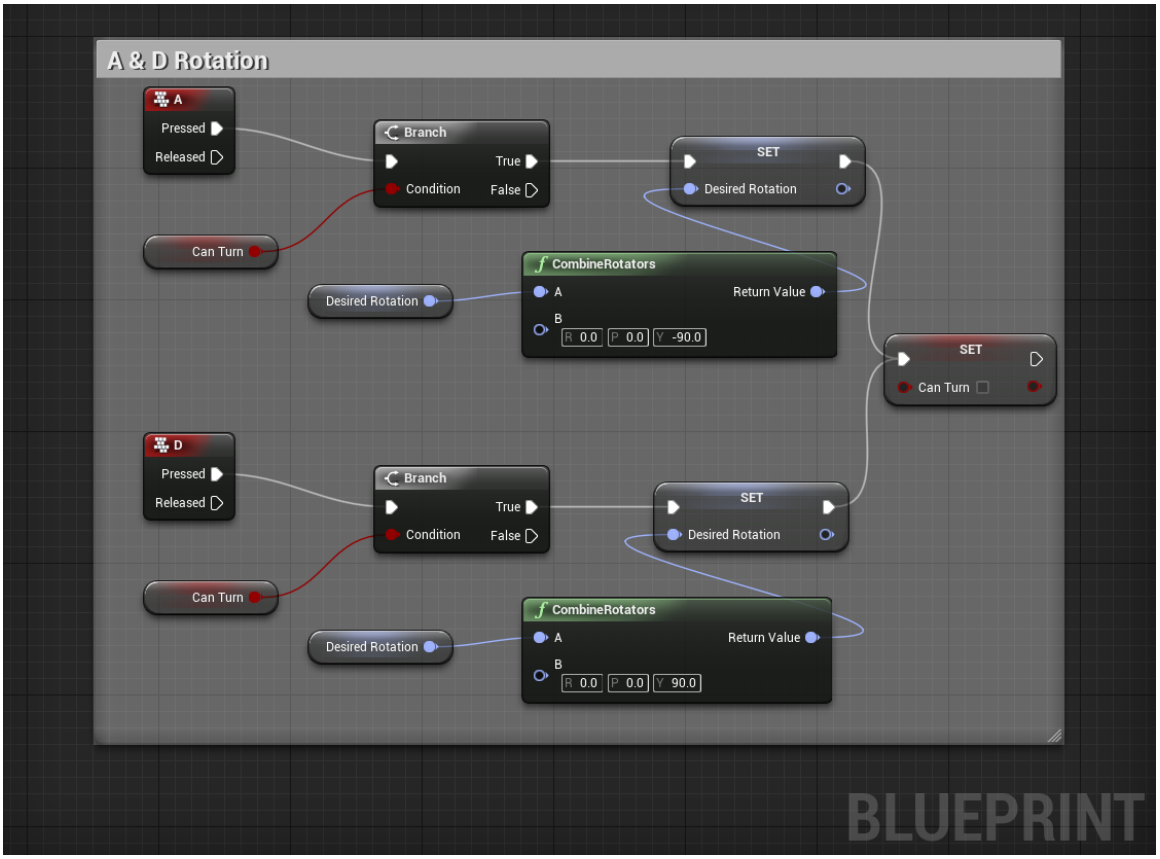
Začnemo tako, da najprej ustvarimo nov projekt. V pogonu Unreal Engine že obstajajo predloge, zato kar uporabimo predlogo za tretjeosebno igro. Začnemo v prostoru, kjer že imamo osnovne objekte. Med njimi je objekt igralca, s katerim se lahko premikamo in skačemo po osnovni mapi. Prikaz prostora je na sliki 18.

Slika 18: Elementi, ki jih pogon Unreal Engine postavi ob začetku novega projekta.

Začetek ustvarjanja igre je dosti lažji zaradi vseh elementov, ki jih že imamo. Pomembno je, da si osnovne nastavitve uredimo tako, da bodo ustrezale našim potrebam. Najprej odstranimo možnost premikanja kamere z miško, saj želimo imeti fiksno kamero, ki bo postavljena za igralcem in malo nad njim. Kamero premikamo tako, da jo označimo in povlečemo na željeno mesto. Odstranimo tudi možnost premikanja igralca z miško, zato je potrebno da na tej točki uredimo premikanje s smernimi tipkami. Premiki tukaj niso ustvarjeni s skriptami, ampak vse stvari urejamo preko diagramov. Za odstranitev premikanja z miško, smo del diagrama, ki je urejal premikanje, izbrisali in ga ponovno ustvarili takšnega, kot smo želeli sami. Miško smo

[3] https://wiki.unrealengine.com/Videos/Player?series=PLZlv_N0_O1gbY4FN8pZuEPVC9PzQThNn1

odstranili zato, ker želimo, da je kamera fiksna in premikanje vedno usmerjeno naravnost. Igralec spreminja smer le s smernimi tipkami, sicer je premikanje naprej samodejno. Ko pritisnemo smerno tipko se igralec pomakne bolj levo ali desno. Če pritisnemo smerno tipko na robu vogala, se celoten igralni prostor zavrti za pravi kot. Del diagrama, ki je ustvarjen za premikanje, vidimo na sliki 19.



Slika 19: Prikazan je del diagrama, ki skrbi za premikanje igralca. Vidimo lahko, da ni potrebno programiranje, saj vso delo poteka z ustvarjanjem elementov v načrtu.

Pri premikanju je pomembno, da označimo, kje na mapi so vogali. Označevanje poteka tako, da v vogalih postavimo navidezne kocke, ki nam služijo kot sprožilec za omogočanje obračanja. Ko igralec vstopi v kocko, se spremenljivka spremeni iz 0 v 1 in igralcu je omogočen obrat za 90°.

Nadaljnje se lotimo ustvarjanja tal. Začnemo tako, da naredim nov objekt, na katerega dodamo teksturo tal, zraven pa dodamo še zidove, ki preprečujejo padec z mape. Cilj je, da se ta objekt nastaja en ob drugem in na tak način ustvari pot za igralca. Pri prehodu skozi objekt želimo, da se zadnji izmed ustvarjenih elementov zbriše in ustvari en nov element na začetku. Na tak način

zagotovimo, da ima igralec vedno primerno dolgo pot. Postopek brisanja starih objektov in ustvarjanja novih je vezan na sprožilec, ki ga namestimo na rob objekta. Ko ga igralec prečka, se postopek brisanja in ustvarjanja elementov začne.

Naša osnovna pot je na tak način ustvarjena. Ker želimo težavnejšo igro, dodamo na pot še ovire. Ovir ne ustvarjamo sami, ampak lahko za oviro uporabimo kamen, ki je že del začetnega paketa, ki nam ga program dodeli ob začetku novega projekta. Nastanek ovir želimo nadzorovati, zato že v naprej označimo mesta na našem objektu tal, kjer želimo, da ovire nastajajo. Ko objekti tal nastajajo in tvorijo pot, se na vsakem objektu pojavi ena ovira na enem izmed predvidenih mest. Mesto, kjer se pojavi, je izbrano naključno.

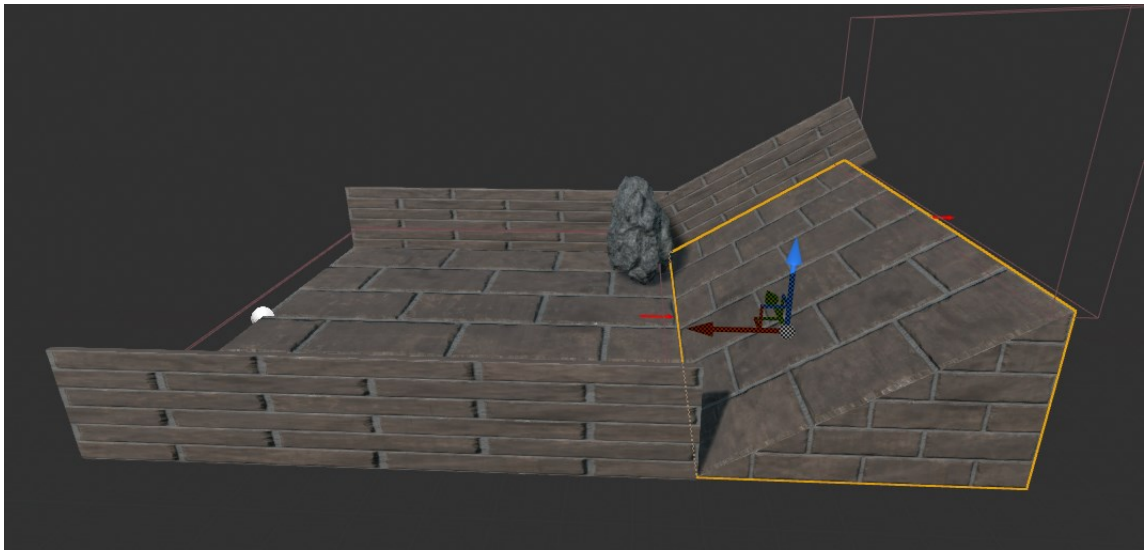
Ovire smo postavili na pot zato, da ob trku uničijo igralca. Če je do trka prišlo preverjamo s pomočjo trkalnika, ki ga moramo dodati na ovire. Dodajanje trkalnika je enostaven proces, objekt označimo in že se nam pojavi možnost `add collider`. Tudi tukaj imamo več različnih možnosti povezanih s natančnostjo našega trkalnika. Izberemo srednjo, ki potrebno natančnost, brez pretirane obremenitve računalnika. Trkalnika na igralca ni treba dodati, saj je že avtomatsko prisoten. Igralcu dodamo v načrt možnost smrti, ki se zgodi ob trku v oviro. Ob trku pride do eksplozije, za katero imamo efekt že na voljo v začetnem paketu. Pomembno je tudi, da po trku onemogočimo nadaljnje premikanje. Premikanje igralca po poti in med ovirami vidimo na sliki 20.

Slika 20: Igralec teče po poti zgrajeni iz objektov tal, na katerih na naključnih mestih nastajajo ovire.

Želimo, da igralec na poti pobira kovance, tak način pridobiva točke. Kovance naredimo tako, da najprej ustvarimo nov objekt v obliki valja in ga postavimo pokonci. Dodamo mu teksturo zlata, kar mu daj sijaj. Sijaj izboljšamo tako, da pod objekt postavimo luč, ki sveti v objekt. Tako ustvarimo videz pomembnega objekta, ki ga ni moč spregledati. Objektu dodamo enakomerno rotacijo in ga naredimo še privlačnejšega. Kovanci nastanejo ob nastanku tal, pojavijo se na naključnih lokacijah. Naša edina skrb je, da določimo, koliko kovancev se lahko pojavi na posameznem objektu tal. Tudi kovancem dodamo trkalnik, ki pa ne vodi do eksplozije, ampak do tega, da kovanec ob dotiku izgine in štejemo ga za pobranega, na kar nas opozori zvok. Za vsak pobran kovanec igralec dobi določeno število točk. Ni dovolj, da igralec točke dobi, treba jih je tudi prikazati.. Prikažemo jih tako, da dodamo besedilo, ki vsebuje prej omenjeno spremenljivko.

Naše ovire in kovanci so nared. Sedaj je potrebno določiti, da na vsakem objektu tal nastanejo prvi ali drugi, nikoli pa ne oboji hkrati. To dosežemo tako, da računalnik za vsak objekt tal naključno izbere celo število nič ali ena. Če je izbrana nič, nastanejo ovire, če je izbrana ena, nastanejo kovanci.

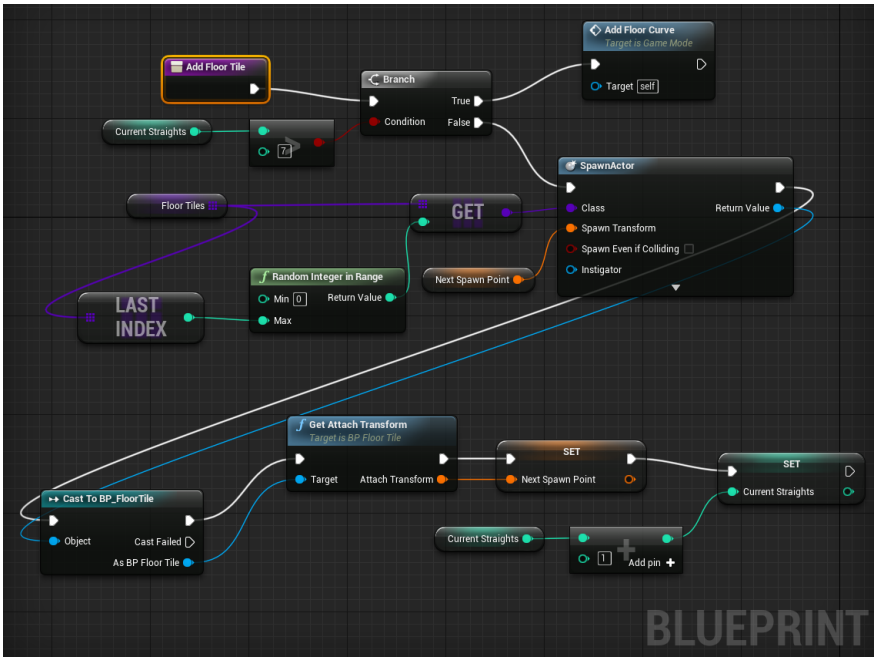
Objekt tal je pripravljen, da začnemo spreminjati njegovo lego. Želimo doseči, da se ne premikamo samo ravno, ampak tlom dodamo nagib navzgor in navzdol. Tako dosežemo razgibanost in naredimo igro bolj zanimivo. To dosežemo tako, da diagram trenutnih tal podvojimo in objektu tal dodamo prizmo. Nastane objekt viden na sliki 21.



Slika 21: Na desni strani je objekt tal, ki smo mu na desni dodali prizmo.

Postopek ponovimo in prizmo postavimo tako, da je naklon navzdol. Dobimo še en novi objekt tal. Pomembno je, da prilagodimo mesto, kjer se naklonu priključi nov objekt tal. To točko moramo premakniti s konca ravnega dela tal (sedaj je ta del na sredini objekta), na konec celotnega objekta (torej na konec rampe). Do sedaj je bil naš diagram tak, da je omogočal le dodajanje ravnih objektov tal. Z dodanimi objekti ustvarimo tabelo iz katere pogon sproti izbira vrsto tal, ki jih bo zgradil. Dodatna možnost v tabeli je možnost luknje, ki jo igralec mora preskočiti. Pogon tako izbira med štirimi različnimi možnostmi. Te možnosti seveda niso nujno potrebne, vendar naredijo igro bolj dinamično in zanimivejšo, hkrati pa tudi zahtevnejšo.

Igro naredijo bolj dinamično tudi zavoji. Najprej ustvarimo vogale na način, da podvojimo diagram za objekt tal, ki mu sedaj dodamo še steno. Tako se steni dotikata in tvorita kot 90°. Na tak način naredimo še dva dodatna objekta tal, kjer en predstavlja točko zavoja v levo, drugi točko zavoja v desno. Na ta objekt dodamo še navidezno kocko, ki je sprožilec za spremembo spremenljivke, ki omogoča obračanje. Prestaviti moramo tudi točko, kjer se gradijo novi objekt tal. Ta točka je sedaj ni več na sprednji strani objekta, saj je tam stena, ampak je na levi ali desni. Načelno bi lahko ta dva objekta dodali v našo tabelo, iz katere program naključno izbira objekte tal, vendar tega ne moremo storiti, saj obstaja možnost, da bi program štiri ali večkrat zapored izbral levi zavoj in bi prišlo do prekrivanja poti. Tej težavi se izognemo tako, da ustvarimo novo tabelo, v kateri sta le elementa tal z levim in desnim zavojem. Diagram nato postavimo tako, da program postavi sedem elementov iz prve tabele, nato pa enega iz druge tabele. Diagram za nastajanje objekta tal prikazuje slika 22.



Slika 22: Prikaz dela diagrama za izbor in nastanek objekta tal.

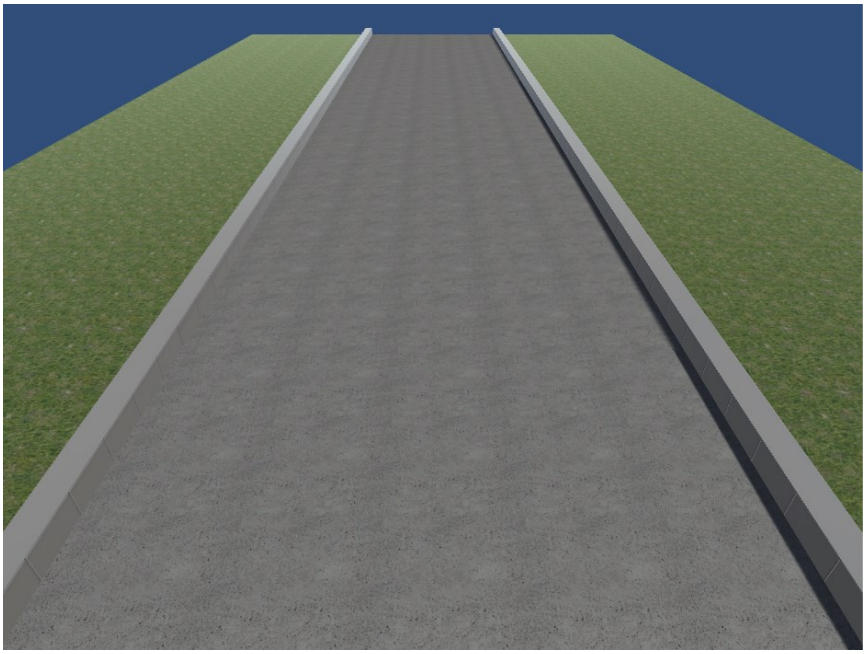
Igra potrebuje le še to, da igralec umre, če se zaleti v zid pred sabo. Na vse zidove dodamo trkalnik in preverimo, če je ob trku kot med igralcem in zidom 90°. Če je, igralca ubijemo.

Naša igra je s tem dokončana, potrebno jo je samo še zgraditi. Izgradnja je končana s pritiskom na gumb, potrebno je le izbrati platformo, na kateri želimo da igra deluje, in lokacijo, kamor nam pogon Unreal Engine igro izvozi [54]. Dokončano igro vidimo na sliki 23.

Slika 23: Dokončana igra Endless Running.

4.3 Razvoj igre Busy Road v pogonu Unity

Začeli smo z ustvarjanjem novega projekta in za platformo izbrali Windows. Za igranje potrebujemo neskončno dolgo cesto. To ne moremo ustvariti tako, da bi naredili osnovni element, ki bi ga dodajali k trenutni cesti, saj pogon Unity deluje na osnovi plavajoče vejice. Pri večjih številih bi zaradi plavajoče vejice izgubili natančnost, gre namreč za igro, kjer je razdalja pomembna in bi lahko bila neskončno velika. Temu smo se izognili tako, da smo cesto naredili fiksno. Igralec stoji na mestu in se premika le levo in desno. Vtis premikanja ima zaradi premikajočih tekstur. Za vsak okvir (angl. frame) premaknemo teksture za neko v naprej določeno število. Večja kot je številka, hitreje se premikajo. Cesto smo olepšali z ograjo in travnatimi površinami. Omenjene teksture smo pridobili s pomočjo spletne trgovine, kjer so bile na voljo brezplačno in smo imeli možnost izbirati med mnogimi. Igra je dobila bolj resničen videz in postala zanimivejša. Izgled igre na tej točki vidimo na sliki 24.



Slika 24: Začetni izgled ceste.

Na cesto smo postavimo avte dveh oblik; en predstavlja igralca, drugi predstavljajo ovire. Modele avtov, ki smo jih želeli, v pogonu Unity ni bilo, zato smo jih prenesli z interneta in jih dodali v pogon. Preden smo avte postavili na cesto, smo na njej določili pet položajev na katerih je lahko avto. Tako so nastali vozni pasovi, med katerimi avto skače, ko se izogiba oviram. Ločili smo jih z belimi črtami. Najprej smo na cesto postavili igralčev avto, ki začne vedno na srednjem pasu. Premikanje med pasovi smo omogočili s pritiskom na smerno tipko. Pomembno je, da ob pritisku pride do premika le za en pas v željeno smer. To smo dosegli s spodnjo kodo. Koda skrbi tudi za to, da igralec ne zapusti cestišča.

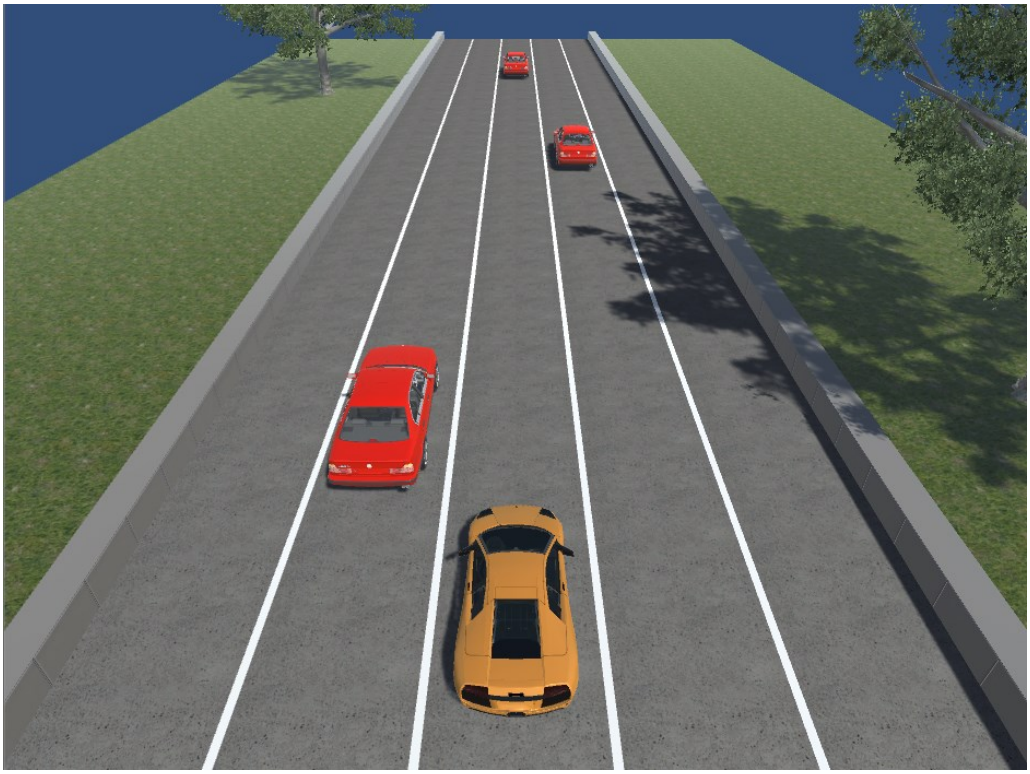
```
void FixedUpdate () {
    float x = player.position.x;
    if (Input.GetKeyDown("a") && player.position.x > -3)
        x = player.position.x - 2f;
    else if (Input.GetKeyDown("d") && player.position.x < 3)
        x = player.position.x + 2f;

    player.position = new Vector3(x, player.position.y,
player.position.z);;
}
```

Sledilo je postavljanje ovir. To dosežemo s skripto, ki na določen časovni interval na enega izmed pasov postavi avto. Hitreje kot se premikamo, krajši je časovni interval. Igra tako pridobi na težavnosti. Potrebno je, da se avti premikajo proti igralcu. Premikanje ustvarimo tako, da objektu ob postavitvi dodamo skripto, ki ga vsako sliko v sekundi premakne bližje nam. Hitrost

premikanja objektov mora biti manjša od hitrosti premikanja tekstur, saj na tak način ustvarimo videz premikajočih se avtov.

Za izboljšanje igre smo ob robu ceste dodali še drevesa. Model dreves smo prenesli z interneta in s skripto dosegli, da se pojavljajo vsakih nekaj sekund na eni izmed strani. Želeli smo doseči resnični videz. Igra sedaj izgleda tako, kot prikazuje slika 25.



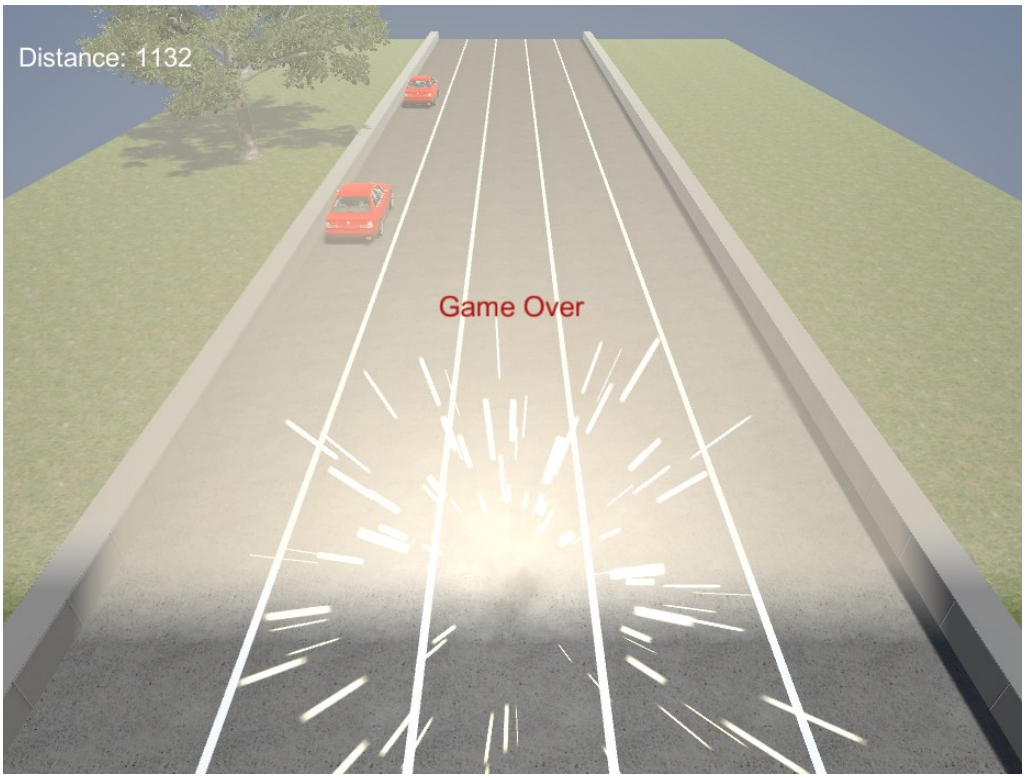
Slika 25: Izgled igre z ovirami in drevesi.

Prišli smo do točke, kjer uredimo preverjanje trkov. Na ovire in igralčev avto dodamo trkalnik. Dodamo ga tako, da preko modela avta dodamo kvader, za katerega se preverja trk. Kvader čim bolj prilagodimo obliki avta. Ob trku dodamo na igralčevi lokaciji učinek eksplozije in uničimo oviro ter igralca. Učinek eksplozije je del začetnega paketa, do katerega nam pogon Unity omogoči dostop z začetkom novega projekta.

Ovire, ki se jim izognemo, potujejo mimo nas v neskončnost. To je nepotrebna obremenitev računalnika, zato se temu želimo izogniti. Za našo progo dodamo velik kvader, ki prekriva celotno igralno površino. Kvader skrijemo in naredimo nevidnega. Dodamo mu trkalnik in ko pride do trka kvadra in ovire, naš avto in oviro uničimo.

Potrebno je urediti še štetje točk, v našem primeru je to prevožena dolžina. Na zaslon dodamo besedilo, ki prikazuje prevoženo razdaljo. Razdaljo štejemo tako, da v glavni skripti dodamo spremenljivko `distance`, ki se povečuje glede na premik tekstur. Na tak način dobimo dejansko razdaljo, kljub temu, da igralec dejansko stoji na mestu.

Dodamo še besedilo `Game Over`, ki se pojavi le takrat, ko pride do trka. Takrat se nehajo premikati teksture in pojavljati ovire. Igre je ob trku konec in igra izgleda približno tako, kot jo vidimo na sliki 26.

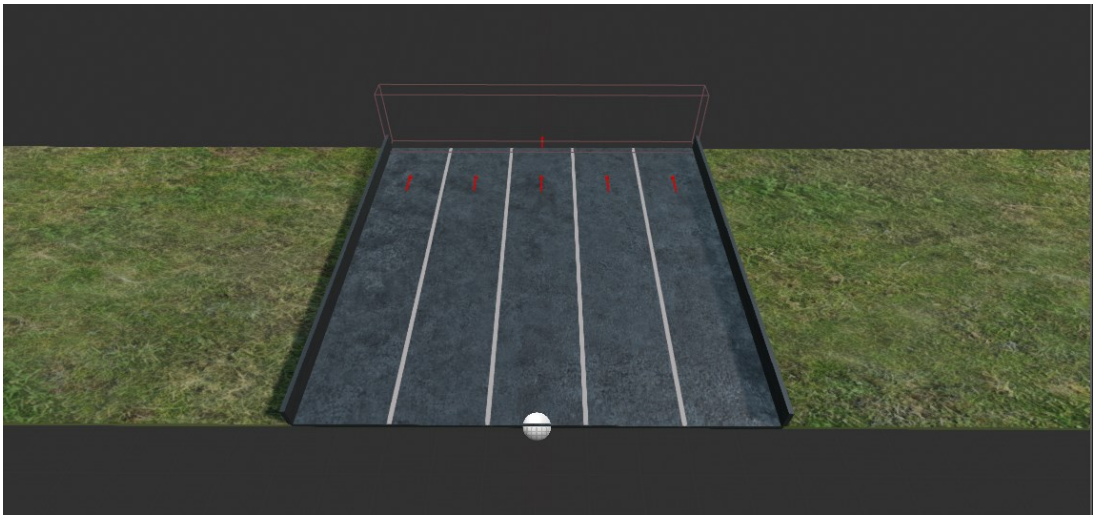


Slika 26: Prikaz igre ob trku, ki igro konča.

4.4 Razvoj igre Busy Road v pogonu Unreal Engine

Začeli smo z ustvarjanjem novega projekta. Izbrali nismo nobene izmed predlog, čeprav bi imeli možnost izbrati avtomobilsko igro, kjer že imamo model avta. Brez predloge smo začeli zato, da bi bila izdelava igre čim bolj podobna izdelavi igre v pogonu Unity. Delo je potekalo s pomočjo diagramov, ki so zelo primerni za začetnike.

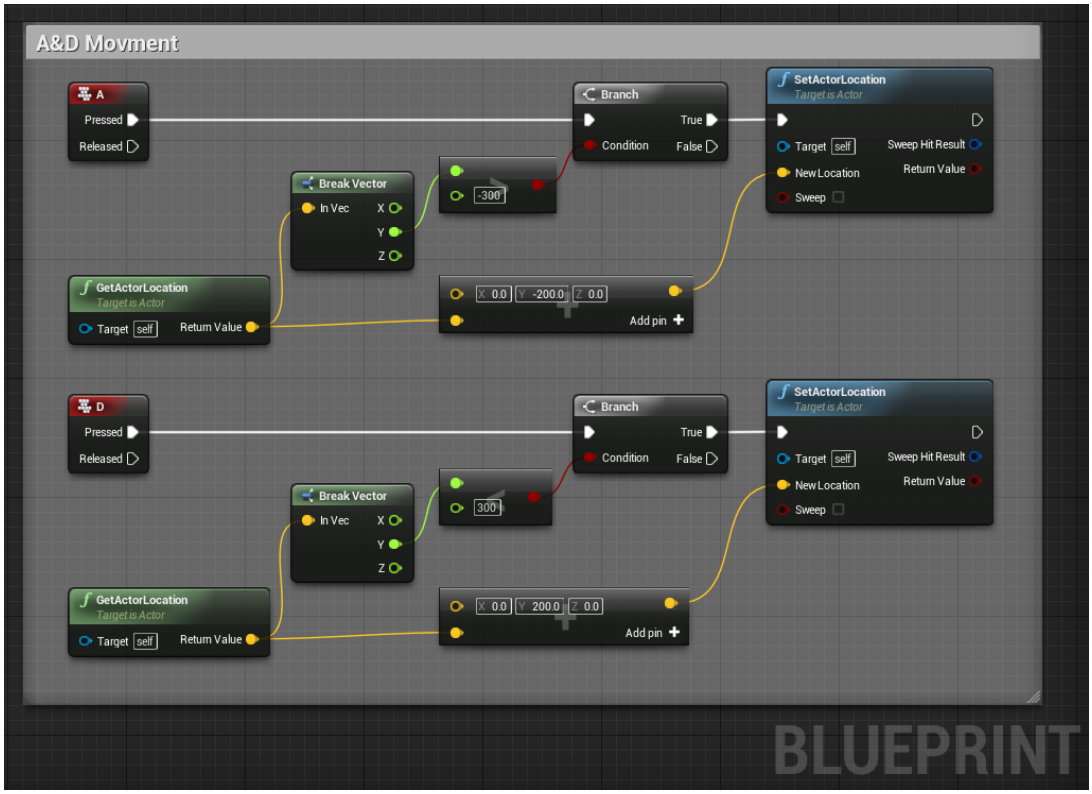
Najprej smo ustvarili objekt ceste. Na sredini je dejanska cesta razdeljena na pet pasov, ob straneh imamo travo, ki jo od ceste ločuje ograja. V pogoni Unreal Engine cesta nastaja med vožnjo, igralec ni statičen. Ker pogon ne uporablja principa plavajoče vejice, ampak uporablja kar navadna števila, ne pride do izgube natančnosti, zato je postopek ustvarjanja ceste podoben kot v igri Endless Running. Ko se premikamo pred nami nastajajo novi objekti, za nami pa se stari brišejo. Vedno ohranjamo določeno dolžino. Nov objekt se doda na stično točko, ki jo označimo na osnovnem objektu. Do izgradnje pride, ko prevozimo kvader, ki smo ga postavili na rob ceste in mu dodali trkalnik. Ob trku se sproži potek izgradnje.



Slika 27: Osnovni objekt ceste.

Slika 27 prikazuje osnovni objekt ceste in že v naprej določene točke, kjer bomo postavili ovire. Vse teksture, ki jih vidimo, so že bile na voljo v sklopu začetnega paketa, kar je naredilo postopek ustvarjanja osnovnega objekta dosti lažji.

Začeli smo z ustvarjanjem novega diagrama za igralca, v katerega smo povlekli model avta. Dodali smo mu trkalnik, ki ga pogon zgradi sam. Uredili smo premikanje avtomobila, pri katerem smo bili pozorni na to, da je premikanje možno le med pasovi in po vnaprej predvidenih mestih. Premikanje je omogočeno s smernimi tipkami. Diagram premikanja prikazuje slika 28.



Slika 28: Diagram premikanja avta.

Določili smo, da se igralcu poveča hitrost ob vsakem premiku skozi kvader ob koncu objekta ceste. Na tak način igri dodamo težavnost in zanimivost.

Potrebujemo tudi ovire. Avto, ki ga uporabimo za oviro smo prenesli z interneta. Diagram ustvarimo tako, da pogon na vsakem objektu ceste ustvari eno oviro. Oviram dodamo trkalnik in preverjamo ali je prišlo do trka z igralcem. Ob trku uničimo igralčev avto in dodamo učinek eksplozije.

Za olepšanje igre smo dodali še drevesa, ki so vezana na osnovni objekt ceste. V diagramu zapišemo, da se drevo pojavi naključno na levi, desni ali nikjer. Dobimo razgibano pokrajino, ki izgleda resnično.

Štetje razdalje je urejeno tako, da spremljamo premikanje naravnost in ga zapišemo kot igralčeve točke. Prikažemo jih s pomočjo besedila v zgornjem levem kotu, kot lahko vidimo na sliki 29.

Slika 29: Končna oblika igre Busy Road.

Igra se konča, ko pride do trka med oviro in igralcem. Izpiše se `Game Over` in po nekaj sekundah se igra samodejno začne od začetka.

5 Primerjava posameznih elementov

Že pri delu s pogonoma so bile opazne razlike, zato smo se lotili podrobnejše primerjave. Primerjali smo skoraj vse elemente, s katerimi smo se srečali pri ustvarjanju iger in so pomembno vplivali na naše delo. Znotraj kategorij smo vsakemu izmed pogonov dali oceno s pomočjo Likertove lestvice [18], kjer je 5 najvišja ocena in 1 najnižja ocena. Ocenam smo dodali uteži, saj vse kategorije niso enako pomembne. Manj pomembne kategorije smo pomnožili z 0,75, srednje pomembne so ohranile svojo vrednost, najpomembnejše pa smo pomnožili z 1,25. Seštevek ocen in strnjeni rezultati so predstavljeni na koncu poglavja.

Seznam primerjanih elementov lahko vidimo spodaj.

1. *Namestitev*
   Primerjali smo tehnične podatke in sistemske zahteve, ki jih imata pogona. Posvetili smo se tudi postopku namestitve. Višjo oceno je dobil pogon, ki ima nižje sistemske zahteve in enostavnejši postopek namestitve.

2. *Izgled in preglednost pogonov*
   Ta kategorija je zelo pomembna za delo s programom, saj boljša preglednost pomeni hitrejše in enostavnejše delo. Izgled je mnogokrat odločilen element, kadar smo v dilemi. Zaradi subjektivne narave te kategorije smo jo označili kot manj pomembno.

3. *Ustvarjanje projekta in začetni paket*
   Pomembnost te kategorije je velika predvsem za začetnike, ki še nimajo veliko izkušenj z igralnimi pogoni in delajo osnovne ter enostavne igre. Dejstvo je, da naprednejši uporabniki ne uporabljajo predlog in se manj zanašajo na vsebino začetnih paketov. Uporabljajo lastne teksture in modele, ter vso delo opravijo sami. Zaradi omenjenega smo kategorijo označili kot manj pomembno.

4. *Ustvarjanje kode*
   Način ustvarjanja kode je bistvenega pomena pri ustvarjanju iger zato je ta kategorija označena kot bolj pomembna. Osredotočili smo se na število programskih jezikov, ki jih pogon podpira. Višjo oceno je dobil pogon, ki je uporabniku ponuja več možnosti.

5. *Grafika in svetloba*
   Osredotočili smo se na delo z orodji, ki jih posamezen pogon ponuja. Primerjali smo kvaliteto grafike v izgrajeni igri.

6. *Trkalnik*
   Pozornost smo usmerili na način ustvarjanja trkalnika. Višjo oceno je dobil pogon, ki nam delo s trkalnikom bolj olajša in nam nudi čim več različnih možnosti.

7. *Podprte platforme*
   Ta kategorija je pogosto bistvenega pomena pri izbiri pogona, zato smo jo označili kot bolj pomembno. Višjo oceno je dobil pogon, ki podpira širši obseg platform.

8. *Spletna trgovina*
   Primerjali smo predvsem obseg spletne trgovine, za katerega seveda želimo, da je čim večji. Pozornost smo namenili tudi primerjavi cen in načinu, kako do trgovine dostopamo.

9. *Pomoč novim uporabnikom*
   Vsak začetek je težak, zato se nam je zdelo pomembno primerjati kateri izmed pogonov začetnikom nudi več pomoči v obliki vodenih videov, dokumentov in morebitnih seminarjev. Kategorija je pomembna le za nove uporabnike, zato smo v splošnem označili za manj pomembno.

10. *Razhroščevanje*
    Primerjali smo možnosti razhroščevanja, ki so vgrajene v pogon. Višjo oceno je dobil pogon, ki nam jih nudi več.

11. *Dostop do izvirne kode*
    Prednost je, če je dostop do izvirne kode omogočen.

5.1 Namestitev

Začnimo z nekaj popolnoma tehničnimi podatki. Unity zasede 5,9 GB prostora na našem disku, medtem ko Unreal Engine zasede kar 14,2 GB prostora. Pri igranju igre Busy Road Unity porablja 10% procesorja in približno 300 MB pomnilnika. Unreal Engine je bolj požrešen; pod istimi pogoji porablja 25% procesorja in kar 1 GB pomnilnika.

Igra Busy Road zgrajena v Unity je velika približno 800 MB, medtem ko je v pogonu Unreal Engine večja od 2 GB.

Sistemske zahteve pri Unity so odvisne od zahtevnosti našega projekta. Uporabljamo ga lahko v operacijskem sistemu Windows XP SP2+, 7 SP1+, 8 in 10. Za Mac OS X je potrebna vsaj različica 10.8. Nujno potrebujemo grafično kartico, ki podpira DirectX 9 [40].

Unreal Engine optimalno deluje na operacijskih sistemih Windows 7 64-bit in Mac OS X 10.9.2. ali novejših verzijah omenjenih. Grafična kartica naj podpira DirectX 11. Procesor naj ima vsaj 4 jedra in pomnilnik naj bo velik vsaj 8 GB. Program bo deloval tudi na manj zmogljivem računalniku, vendar bo njegovo delovanje omejeno [41].

Unity bo deloval na skoraj vsakem računalniku, še posebej če se lotimo ustvarjanja enostavnih iger. Unreal Engine ima veliko bolj stroge zahteve.

Ocena
Unity: 5
Unreal Engine: 3

5.2 Izgled in preglednost pogonov

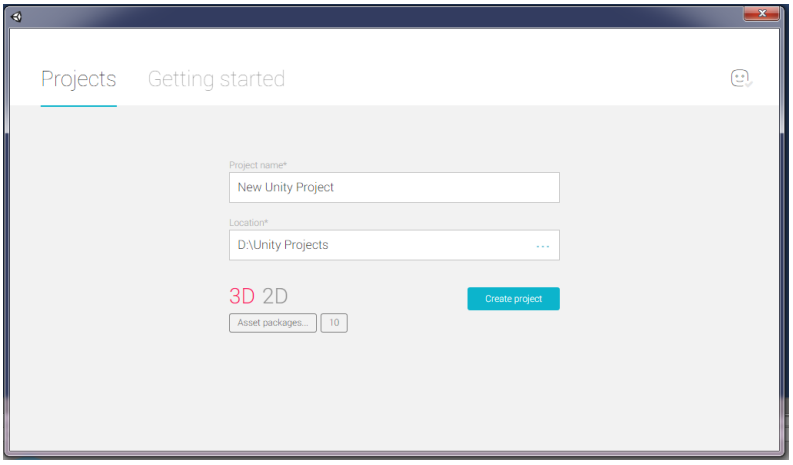
Oba programa sta v osnovi postavljena tako, da imamo na sredini pregled nad igralno površino, ki je obdana z okni, v katerih odpiramo mape, spreminjamo nastavitve in imamo seznam projektov in podobno. Oba nam omogočata, da si elemente začetnega zaslona prilagodimo po naših željah in potrebah. Ko je naša postavitev takšna, kakršno želimo, si jo shranimo in jo imamo pripravljeno za prihodnjo uporabo. Oba imata na vrhu opravilno vrstico. Unreal Engine je za svojo osnovno barvo izbral temno sivo, kar je še posebej prijazno očem, če delamo v večernih urah. Unity je obarvan v svetlo sivo. Prva pomembnejša razlika so ikone programov. Unreal Engine ima ikone večje, pogosto jih spremljajo tudi slike, medtem ko so se razvijalci pogona Unity odločili za manjše ikone.

Dejstvo je, da je izbira odvisna od posameznika, sami pa menimo, da je predvsem za začetnike lažje, če vidimo, kaj bomo dobili. V Unreal Engine se je po našem mnenju lažje znajti ravno zaradi večjih in preglednejših ikon.

Ocena
Unity: 4
Unreal Engine: 5

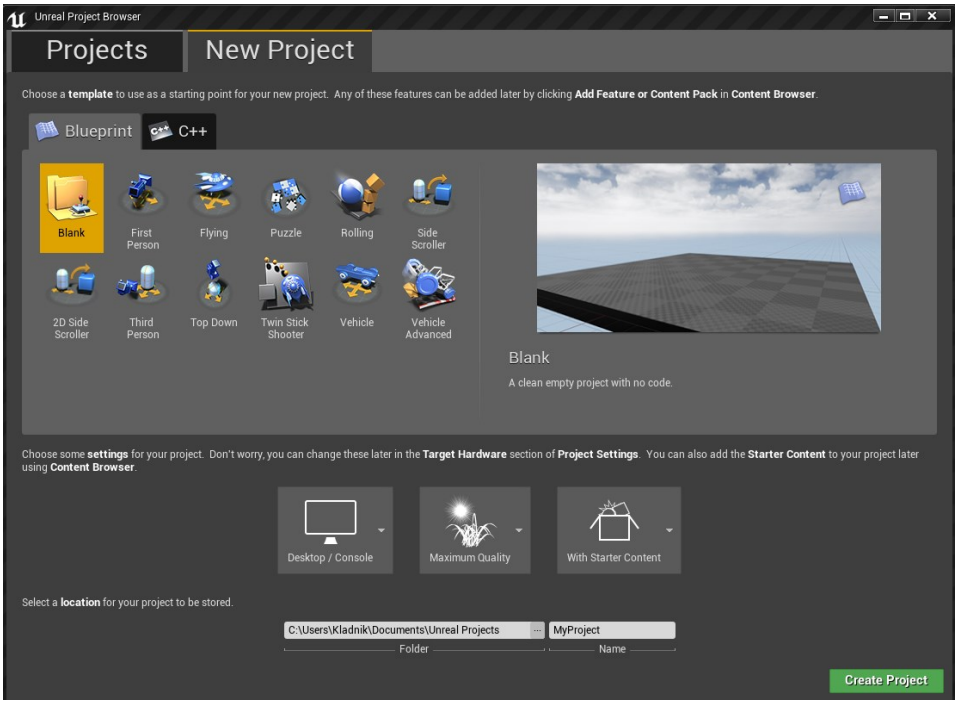
5.3 Ustvarjanje projekta in začetni paket

Pri obeh programih je ustvarjanje projekta enostavno, začetno okno v pogonu Unity vidimo na sliki 30. Ko je projekt ustvarjen, nas program postavi v začetni igralni prostor. Pri pogonu Unity je igralni prostor prazen, postavljeno je le sonce kot vir svetlobe. Ob začetku ustvarjanja programa se lahko odločimo, da želimo, da nam program dodeli tudi nekaj osnovnih elementov in nam na tak način olajša delo. V začetnem paketu nam Unity ponudi možnost uporabe le osnovnih tekstur, nekaj osnovnih objektov in tudi nekaj osnovnih efektov kot so na primer eksplozija, dim in ogenj. Teh efektov ne spremlja zvok, zato ga moramo sami kasneje dodati, v kolikor ga želimo imeti. Med osnovnimi objekti najdemo recimo avto in letalo. Možnost imamo uporabiti tudi kakšno izmed osnovnih skript.

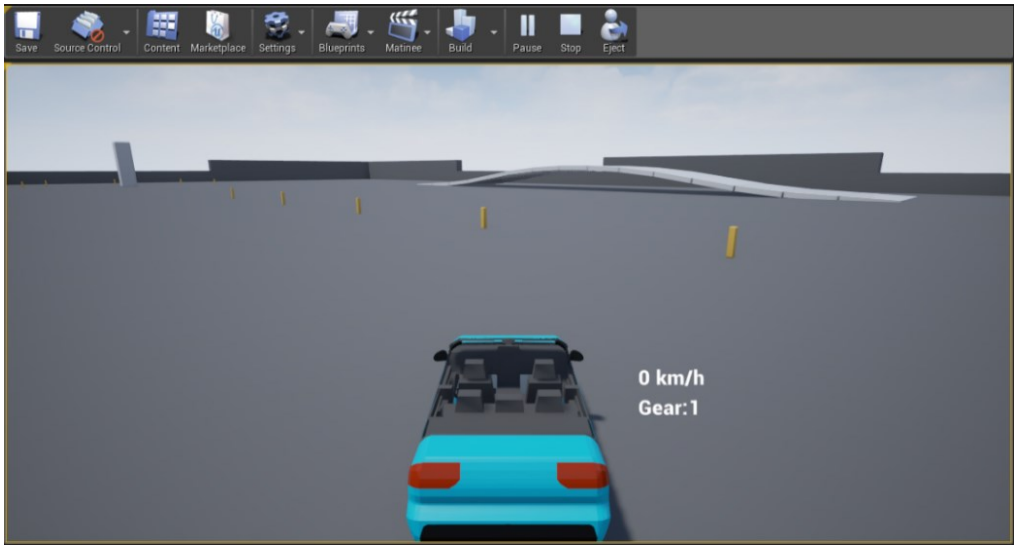


Slika 30: Ustvarjanje novega projekta v Unity.

Unreal Engine nas že ob ustvarjanju novega projekta vpraša, kakšen tip igre bomo gradili in od tega je odvisno, kakšen je začetni paket, ki nam ga pripravi. Možne izbire prikazuje slika 31. Če izberemo recimo igro, kjer vozimo avto, že v začetku dobimo zgrajen avto in pravi poligon. Hkrati pa imamo tudi dostop do posebnih efektov, ki jih v tem primeru spremlja tudi zvok. Na nek način že imamo podano celotno igro, avtu meri celo hitrost in prestave. Podana so tla, zidovi, osvetlitev in še določeni drugi elementi. Izgled omenjenega osnovnega prostora prikazuje slika 32. Omogočena je uporaba tekstur, katerih nabor je dosti širši kot v Unity, dosti širši tudi nabor osnovnih oblik. Določeni načrti so napisani že vnaprej in jih ni treba tvoriti.



Slika 31: Način ustvarjanja novega projekta v Unreal Engine.



Slika 32: Primer začetnega igralnega polja v avtomobilski igri v Unreal Engine.

V določenih primerih je prazno polje prednost, vendar menimo, da je lažje začeti s programiranjem, če osnovne elemente in funkcije že imamo. Tudi pogon Unreal Engine nam seveda omogoča, da vse že postavljene elemente izbrišemo in začnemo popolnoma od začetka. Unity ima to slabost, da je smo omejeni z malo teksturami in objekti, vse kar potrebujemo,

moramo narediti sami ali si prenesti s spletne trgovine. Velika pomanjkljivost se kaže tudi na področju zvokov, ki jih sploh ni na voljo. Unreal Engine skoraj celotno osnovno igro zgradi namesto nas in nam da na voljo ogromno tekstur, objektov in zvokov, s katerimi igro prilagodimo lastnim potrebam.

Ocena
Unity: 3
Unreal Engine: 5

5.4 Ustvarjanje kode

Unity dopušča ustvarjanje kode v več programskih jezikih. Izbiramo lahko med C#, JavaScript in Boo. Sami si izberemo tistega, ki nam najbolj leži, dejstvo pa je, da moramo enega izmed njih kar dobro poznati. Programiranje v Unreal Engine temelji na C++ ali na ustvarjanju diagramov. Prednost slednjega je, da je za delo z diagrami dovolj že poznavanje psevdokode. Možnost uporabe načrtov omogoča ustvarjanje iger tudi razvijalcem, ki so manj vešči programiranja in so na tem področju šele začetniki, saj je preprosto igro možno ustvariti brez ene vrstice kode [34]. Unity nam daje veliko možnosti pri izbiri programskega jezika, medtem je Unreal Engine omejen le na C++. Zavedati se moramo, da čeprav je delo z diagrami zelo priročno, ima pomanjkljivosti. Težava je predvsem v tem, da je odzivni čas tako ustvarjenih iger daljši in da diagrami niso primerni za zahtevne igre.

Ocena
Unity: 5
Unreal Engine: 3

5.5 Grafika in svetloba

Svetloba je tista, ki doda igri piko na i. Oba igralna pogona imata podoben način urejanja luči. Oba imata na voljo več različnih vrst luči npr. smerna luč, točkovna luč, glavna luč, ambientalna luč in še mnoge druge. Njihovo intenziteto prilagajamo s premikanjem kurzorja na svetlobni lestvici. Nastavimo lahko tudi barvo luči in njihov doseg. Možno je vplivati na obliko sence in na to, ali se naj svetloba te luči od česa odbija. Luči premikamo tako, da jih povlečemo na željeno lokacijo.

Delo z obema pogonoma je na tem področju precej enakovredno in nudita nam mnoge možnosti, vendar je sistem osvetlitve, ki ga uporablja Unreal Engine znan po tem, da lahko z

njim dosežemo zelo čudovito grafiko, ki izgleda kot resnični svet. To je možno zaradi tega, ker dopušča uvoz izredno kvalitetnih objektov, zelo dobro osvetljevanje in uravnavanje svetlobe, še posebej pomemben pa je način izgradnje igre. Unity aktivno dela na tem področju in Unity 5 se s svojimi grafičnimi sposobnostmi približuje pogonu Unreal Engine, še vseeno pa ne moremo ustvariti pokrajine, ki bi izgledala kot resnična. Razliko v grafiki lahko vidimo na slikah 33 in 34. Za primer, ki temelji na lastni igri lahko pogledamo sliki 25 in 29 v 4. poglavju.

Slika 33: Pokrajina ustvarjena v Unity [44].

Slika 34: Prostor ustvarjen v Unreal Engine [45].

Ocena
Unity: 3
Unreal Engine: 5

5.6 Trkalnik

O trkalniku smo že veliko govorili, a dejstvo je, da se mu pri ustvarjanju iger skoraj ni moč izogniti. Zato je pomembno, da igralni pogon naredi naše delo s trkalnikom čim lažje.

V pogonu Unity možnosti izbire zahtevnosti in natančnosti trkalnika nimamo. Če želimo poenostaviti trkalnik je potrebno ustvariti enostavnejšo repliko objekta, sicer pa je dodajanje trkalnika prav tako zelo enostavno.

Unreal Engine doda trkalnik na osnovni objekt (recimo na objekt strelca v tretjeosebni strelni igri) kar sam in nam s tem prihrani skoraj polovico dela v primerih, ko ugotavljamo trk osnovnega objekta. Naše delo je, da trkalnik dodamo le še na drug objekt, kar pa naredimo tako, da preprosto izberemo med različno zahtevnimi trkalniki in stisnemo tipko, s katero ga dodamo.

Ocena
Unity: 4
Unreal Engine: 5

5.7 Podprte platforme

Unity podpira 21 različnih platform in je tako primeren za razvijanje katerih koli iger. Podpira tako novejše kot starejše različice. Unreal Engine jih podpira le 11. Med njimi so le najnovejše različice posameznih, če želimo ustvarjati za starejšo različico potrebujemo starejšo različico pogona. Na to ne gledamo kot na slabost, saj običajno igre nastajajo ravno za novejše različice platform.

Ocena
Unity: 5
Unreal Engine: 4

5. 8 Spletna trgovina

Oba pogona imata spletno trgovino, ki nam omogoča nakup različnih stvari, ki jih potrebujemo za svoje delo. Spletno trgovino uporabljamo z računom, ki smo ga ob namestitvi pogona ustvarili. Kupimo lahko objekte, efekte, zvoke, predloge iger in tudi že dokončane igre. Obe trgovini imate možnost iskanja po različnih kategorijah in predogleda elementov. Določene

vsebine so tudi brezplačne. Z nakupom prihranimo čas izdelovanja teh objektov, eden izmed primerov je ogenj, katerega izgradnja je lahko zelo dolgotrajna in zamudna, trgovina pa nam omogoča, da ta element kupimo in ga uporabimo v lastni igri.

V Unity je trgovina dostopna preko spletne strani, medtem ko je v Unreal Engine trgovina del sprožilnika (angl. launcher). Trgovini imata sicer različen izgled, vendar pa imata enak namen in podobno funkcionalnost. Največja razlika med njima je obseg. Trgovina pogona Unity je veliko večja od trgovine pogona Unreal Engine. Trgovina slednjega se sicer, od kar je pogon brezplačen, širi, vendar je še vedno dosti manjša.

Unity kljub svoji velikosti določenih elementov ne zagotavlja, sicer pa so cene v povprečju gledane nižje kot v pogonu Unreal Engine.

Ocena
Unity: 4
Unreal Engine: 2

5.9 Pomoč novim uporabnikom

Unity ima na svoji spletni strani[4] zavihek za nove programerje, kjer jim ponuja različne načine učenja uporabe pogona Unity. Omogočena so predavanja o Unity, različna dokumentacija, forum in kar dosti vodenih videov, ki so uporabniku zelo prijazni. Razvrščeni so po težavnosti in z izbiro sklopa videov nas ti vodijo skozi ustvarjanje igre korak za korakom. Ob videih je na voljo tudi koda, ki bi jo sicer morali ustvariti sami in jo lahko uporabimo, kadar se nam kaj zalomi ali pa že samo zato, da preverimo pravilnost.. Že samo s pomočjo spletne strani lahko pridobimo osnovno znanje o delu v pogonu Unity, kar je zagotovo ogromna prednost.

Tudi Unreal Engine ima na svoji spletni strani[5] oddelek za učenje, ki vsebuje podkategorije z dokumentacijo, vodenimi videi in posebno Wikipedijo, namenjeno uporabnikom Unreal Engine. Najbolj uporabljeni so videi, ki so prav tako v več kategorijah in vodijo uporabnika po korakih. Ob njih sicer ni na voljo kode, kar je povezano s tem, da delo v pogonu Unreal Engine ne temelji na kodi. Morebitna pomanjkljivost je odsotnost predavanj.

Novi programerji pomoč pogosto iščejo tudi na forumih in na svetovnem spletu, kjer je Unity-jeva skupnost dosti večja od skupnosti pogona Unreal Engine, kar pomeni, da bodo na tak način lažje in hitreje našli rešitev za svoje težave.

[4] https://unity3d.com/

[5] https://www.unrealengine.com/

Ocena
Unity: 5
Unreal Engine: 4

5.10 Razhroščevanje

Razhroščevanje je proces iskanja in odpravljanja napak v našem programu. Proces razhroščevanja opravljamo tudi pri ustvarjanju iger in sicer tako, da igro na določeni točki ustavimo in preverimo, kako se obnaša v tistem trenutku, ali je prišlo do kakšne napake, ali vse funkcije potekajo kot morajo, koliko okvirjev na sekundo se predvaja in podobno.

Unity ima možnost razhroščevanja, vendar potrebujemo vtičnik `UnityVS`, ki je znan po tem, da je dokaj počasen. Nimamo možnosti hkrati preverjati večih različic igre ali urejati kodo med razhroščevanjem.

Unreal Engine nam razhroščevanje omogoča s pritiskom na tipko `F5`. Tudi med procesom razhroščevanja lahko kodo urejamo, edina omejitev je ta, da je ne moremo zgraditi. Postopek razhroščevanja lahko zaženemo tudi, če je igra že na omrežju in igrana s strani več igralcev. Unreal Engine bo preveril, kako se igra obnaša na vsakem izmed njih [33].

Ocena
Unity: 2
Unreal Engine: 5

5.11 Dostop do izvirne kode

Pogosto lažje rešimo kakšno izmed programerskih težav, če imamo dostop do izvirne kode in s tem možnost seznanitve z delovanjem samega igralnega pogona. Unreal Engine to možnost ima, saj je vsa koda prosto dostopna na GitHubu[6]. Unity te možnosti žal nima niti v brezplačni niti v plačljivi verziji. Če želimo vpogled v kodo je potrebno kontaktirati podjetje [31].

Ocena
Unity: 2
Unreal Engine: 5

[6] https://www.unrealengine.com/ue4-on-github

5.12 Rezultati primerjave

	POMEMBNOST KATEGORIJE	UNITY	UNREAL ENGINE
1	1	5	3
2	0,75	4	5
3	0,75	3	5
4	1,25	5	3
5	1	3	5
6	1	4	5
7	1,25	5	4
8	1	4	2
9	0,75	5	4
10	1	2	5
11	1	2	5
	Skupaj z upoštevanimi utežmi	= 41,5	= 44,25

Tabela 1: Prikaz rezultatov primerjave pogonov

Tabela 1 prikazuje rezultate naše primerjave. Glede na izbrane elemente vidimo, da je boljša izbira Unreal Engine, ki dosega na vseh področjih razen na področju spletne trgovine, visoke ocene. Kljub temu, da se je Unity bolje izkazal v obeh pomembnejših kategorijah, je zbral manj točk. To lahko pripišemo pomanjkljivostim na področju grafike, razhroščevanja in dostopa do izvirne kode.

6 Sklepne ugotovitve

Unity in Unreal Engine sta vodilna na področju igralnih pogonov, vsak izmed njiju ima svoje prednosti in svoje slabosti.

Unity razvijalcu iger ponuja ogromno možnosti pri ustvarjanju kode z različnimi programskimi jeziki in mnogo uporabnimi funkcijami. Pogosto je prva izbira razvijalcev tudi zato, ker podpira zelo širok spekter različnih platform. Njegova prednost je to, da je že dalj časa na voljo v brezplačni različici, kar zagotovo pripomore k temu, da je Unity danes najbolj uporabljan igralni pogon. Kljub ogromni skupnosti, ki pripomore k temu, da svoje težave pri programiranju lažje rešimo, se zelo trudijo na področju izobraževanja novih razvijalcev. Njegova slabost je predvsem to, da je bolj tog s področja ustvarjanja novih objektov, kar se delno odtehta s tem, da je njegova spletna trgovina ogromna in v večini primerov tam najdemo, kar potrebujemo. Žal nam v nobeni različici ne omogoča dostopa do izvirne kode.

Unreal Engine je postal popularnejši z letošnjim marcem, vendar je zagotovo njegova prednost to, da je tudi pred tem imel zelo zmerno in dosegljivo ceno. Čeprav se z vodenimi videi trudi poskrbeti za nove razvijalce, ne moremo trditi, da na tem področju tako uspešen kot Unity. Odličen je za tiste, ki niso tako vešči programiranja v katerem izmed popularnih programskih jezikov, saj omogoča ustvarjanje načrtov, za katere je dovolj že osnovno znanje. Za začetnike je primeren tudi zato, ker nam pripravi odličen začetni igralni prostor in nam s tem prihrani ogromno dela. Začetni paket je ogromen in lahko z njim zadovoljimo večino svojih potreb. Kljub temu, da za veliko stvari poskrbi namesto nas, nam omogoča, da sami ustvarimo teksture in objekte, ki jih potrebujemo in nam jih pomaga oživeti z naprednimi možnostmi osvetlitve.

Glede na izvedeno primerjavo lahko vidimo, da je Unreal Engine boljša izbira. Razlika v zbranih točkah ni velika in dejstvo je, da so vsakemu posamezniku pomembne različne kategorije. Unity je boljša izbira za tiste, ki dajejo manjši pomen na grafiko, razhroščevanje in dostop do odprte kode in ki so vešči programiranja v katerem izmed jezikov, ki jih ponuja. Ravno izbira mnogih programskih jezikov in podpor številnim platformam je to, zaradi česar je danes najbolj uporabljen pogon.

Naša primerjava bi bila še boljša, če bi primerjali še dodatne elemente in na drugačen način določili pomembnost kategorij. Zagotovo bi bila primerjava objektivnejša, če bi za izbor elementov in določitev pomembnosti izvedli anketo in tako pridobili mnenje večjega kroga ljudi. Primerjavo bi bilo smiselno razčleniti tudi glede na različne oblike iger (2D in 3D) in glede na različne platforme. Naši rezultati bi bili tako bolj specifični in bi omogočali izbor glede na naše lastne potrebe. Kljub omenjenemu izvedena primerjava omogoča, da uporabnik

primerja le posamezne kategorije, ki jih on smatra za pomembne in na tak način ugotovi, kateri pogon je primernejši zanj.

Dejstvo je, da končna odločitev o tem, kateri izmed igralnih pogonov je boljši, leži na vsakem posamezniku. Pri odločitvi je potrebno imeti v mislih znanje, ki ga imamo, predvsem pa igro, ki jo želimo ustvariti. V pomoč nam naj bo dejstvo, da sta oba programa brezplačna in da ju imamo možnost preizkusiti, preden se dokončno odločimo. Zavedati se moramo, da vsaka še tako nepomembna igra prinaša zadovoljstvo in nasmeh na usta igralcev in to naj bo naša motivacija za delo, ne glede na to za kateri igralni pogon se odločimo.

Literatura

[1] (14.9.2015) Age of Empires. Dostopno na: http://www.ageofempires.com/

[2] (14.9.2015) Bejeweled. Dostopno na: http://bejeweled.popcap.com/html5/0.9.12.9490/html5/Bejeweled.html

[3] (14.9.2015) Boo. Dostopno na: https://en.wikipedia.org/wiki/Boo_(programming_language)

[4] (14.9.2015) Command & Conquer. Dostopno na: http://www.commandandconquer.com/

[5] (14.9.2015) CryEngine. Dostopno na: http://www.crytek.com/cryengine

[6] (14.9.2015) Crysis. Dostopno na: http://crysis.com/,

[7] (14.9.2015) Časovni pregled arkadnih iger: Dostopno na: https://en.wikipedia.org/wiki/Timeline_of_arcade_video_game_history

[8] (14.9.2015) Doom sporočilo javnosti ob izidu. Dostopno na: http://web.archive.org/web/20120825053443/http://www.rome.ro/lee_killough/history/doompr3.shtml

[9] (14.9.2015) Farmwille. Dostopno na: https://www.farmville.com/

[10] (14.9.2015) Gameboy. Dostopno na: http://arstechnica.com/gaming/2009/04/game-boy-20th-anniversary/

[11] (14.9.2015) Gear VR. Dostopno na: http://www.samsung.com/global/microsite/gearvr/gearvr_features.html

[12] (14.9.2015) Graf porabe sredstev v industriji računalniških iger. Dostopno na: www.newzoo.com

[13] (14.9.2015) HeroEngine. Dostopno na: http://www.heroengine.com/

[14] (14.9.2015) HTC Vive. Dostopno na: http://www.htcvr.com/

[15] (14.9.2015) Igralni pogon. Dostopno na: https://en.wikipedia.org/wiki/Game_engine

[16] (14.9.2015) Informacije o OXO igri. Dostopno na: https://en.wikipedia.org/wiki/OXO

[17] (14.9.2015) Izbira med različnimi igralnimi pogoni. Dostopno na: http://publicvr.info/publications/Lewis2002.pdf

[18] (14.9.2015) Likertova lestvica. Dostopno na: http://www.mizs.gov.si/fileadmin/mizs.gov.si/pageuploads/podrocje/odrasli/Gradiva_ESS/ACS_Izobrazevanje/ACSIzobrazevanje_33Metodologija.pdf (str.15)

[19] (14.9.2015) Microsoft Hololens. Dostopno na: https://www.microsoft.com/microsoft-hololens/en-us

[20] (14.9.2015) O NES. Dostopno na: http://nintendo.wikia.com/wiki/Nintendo_Entertainment_System

[21] (14.9.2015) Oculus Rift. Dostopno na: https://www.oculus.com/en-us/

[22] (14.9.2015) Osnovne informacije o Unityju – podpora platform. Dostopno na: http://unity3d.com/unity/multiplatform/

[23] (14.9.2015) Osnovne informacije o Unityju in slika logotipa. Dostopno na: https://en.wikipedia.org/wiki/Unity_%28game_engine%29

[24] (14.9.2015) Osnovne informacije o Unityju. Dostopno na: http://www.informit.com/articles/article.aspx?p=2031153

[25] (14.9.2015) Osnovne informacije o Unreal Enginu in logotip. Dostopno na: https://en.wikipedia.org/wiki/Unreal_Engine

[26] (14.9.2015) Osnovne informacije o Unreal Enginu. Dostopno na: http://www.popularmechanics.com/culture/gaming/a9178/how-the-unreal-engine-became-a-gaming-powerhouse-15625586/

[27] (14.9.2015) Osnovne informacije o Unreal Enginu. Dostopno na: https://www.unrealengine.com/what-is-unreal-engine-4

[28] (14.9.2015) Ouya. Dostopno na: https://www.ouya.tv/

[29] (14.9.2015) PlayStation. Dostopno na: http://si.playstation.com/ps4/index.html

[30] (14.9.2015) Pomen Spacewars!. Dostopno na: http://www.nytimes.com/1990/12/16/business/digital-fetes-the-germ-that-began-a-revolution.html

[31] (14.9.2015) Prednosti in slabosti Unityja in Unreal Engina. Dostopno na: https://www.reddit.com/r/gamedev/comments/2xtcmr/the_inevitable_ue4_vs_unity_5_faceoff_megathread/cp36s7c

[32] (14.9.2015) Pregled zgodovine računalniških iger. Dostopno na: http://www.emunix.emich.edu/~evett/GameProgramming/History.pdf

[33] (14.9.2015) Primerjava razhroščevanja. Dostopno na: http://maikklein.github.io/2015/07/20/Unreal-vs-Unity/

[34] (14.9.2015) Primerjava ustvarjanja kode v Unityju in Unreal Enginu. Dostopno na: http://stfalcon.com/en/blog/post/unity3d-vs-unreal-engine-4

[35] (14.9.2015) Razlike med Unity Personal in Unity Professional. Dostopno na: https://unity3d.com/get-unity

[36] (14.9.2015) Razširjenost Unity. Dostopno na: https://unity3d.com/public-relations

[37] (14.9.2015) Razvoj PlayStationa. Dostopno na: http://www.pocket-lint.com/news/97585-evolution-of-sony-playstation-consoles

[38] (14.9.2015) Saga Genesis. Dostopno na: https://en.wikipedia.org/wiki/Sega_Genesis

[39] (14.9.2015) Seznam igralnih pogonov. Dostopno na: http://venturebeat.com/2014/08/20/the-top-10-engines-that-can-help-you-make-your-game/

[40] (14.9.2015) Sistemske zahteve za Unity. Dostopno na: https://unity3d.com/unity/system-requirements

[41] (14.9.2015) Sistemske zahteve za Unreal Engine. Dostopno na: https://www.unrealengine.com/faq

[42] (14.9.2015) Slika Atari 2600 konzole. Dostopno na: https://en.wikipedia.org/wiki/Atari_2600#/media/File:Atari-2600-Wood-4Sw-Set.jpg

[43] (14.9.2015) Slika igre Tennis for two. Dostopno na: http://computingforever.com/2014/09/12/first-video-game-grand-theft-auto-online/

[44] (14.9.2015) Slika pokrajine v Unityju. Dostopno na: https://www.youtube.com/watch?v=dyb79pnaWek

[45] (14.9.2015) Slika pokrajine v Unreal Enginu. Dostopno na: http://www.dsogaming.com/screenshot-news/here-is-what-mirrors-edge-could-look-like-in-unreal-engine-4/

[46] (14.9.2015) Slika prvega PlayStationa. Dostopno na: http://za.playstation.com/media/kEBdlHhR/161/PS1_Image.jpg

[47] (14.9.2015) Slika Unreal Paris. Dostopno na: https://www.unrealengine.com/blog/building-unreal-paris

[48] (14.9.2015) Source 2. Dostopno na: http://www.valvesoftware.com/

[49] (14.9.2015) Število igralcev Farmwille. Dostopno na: http://mashable.com/2010/02/20/farmville-80-million-users/#kL7bKtlQCEk0

[50] (14.9.2015) Število igralcev World of Warcraft. Dostopno na: http://www.statista.com/statistics/276601/number-of-world-of-warcraft-subscribers-by-quarter/

[51] (14.9.2015) Tizen. Dostopno na: https://www.tizen.org/

[52] (14.9.2015) Unreal Engine Photorealism. Dostopno na: http://www.gamespot.com/articles/these-unreal-engine-4-scenes-are-unbelievably-real/1100-6428589/

[53] (14.9.2015) Uspehi Unreal Engine. Dostopno na: https://www.unrealengine.com/awards-accolades

[54] (14.9.2015) Vodeni videi za Endless Running. Dostopno na: https://wiki.unrealengine.com/Videos/Player?series=PLZlv_N0_O1gbY4FN8pZuEPVC9PzQThNn1

[55] (14.9.2015) Vodeni videi za igro Space Shooter. Dostopno na: https://unity3d.com/learn/tutorials/projects/space-shooter-tutorial

[56] (14.9.2015) Wii. Dostopno na: http://wii.com/

[57] (14.9.2015) World of Warcraft. Dostopno na: http://eu.battle.net/wow/en/

[58] (14.9.2015) Xbox. Dostono na: http://www.xbox.com/sl-SI/

[59] (14.9.2015) Zgodovina računalniških iger in slika OXO. Dostopno na: http://www.cs.uu.nl/docs/vakken/b2go/literature/history_of_games.pdf