Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aljoša Omejc

Igra na platformi Daydream

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Peter Peer

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Igra na platformi daydream

Daydream platform based game

Navidezna resničnost je že dodobra zasidrana v svetu iger. Tudi podpora razvoju je že zelo pestra. Predstavite platformo Daydream ter na njeni osnovi začrtajte igro ter jo implementirajte. Na koncu zasnujte kvalitativno evalvacijo igre ter predstavite izsledke.

Zahvaljujem se mentorju izr. prof. dr. Petru Peeru za nasvete in pomoč pri pisanju diplomske naloge. Zahvala gre tudi družini za vso pomoč skozi leta šolanja.

Posebna zahvala pa gre prijateljem Davidu Mavcu, Aljažu Koširju in Janu Živkoviču za vso pomoč in vse zabavne trenutke v času študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
VR	virtual reality	navidezna resničnost
SDK	software development kit	komplet programskih orodij
UI	user interface	uporabniški vmesnik
NPC	non-player character	umetno-inteligenčni lik
IDE	integrated development environment	integrirano razvojno okolje
DOF	degrees of freedom	prostostne stopnje

Povzetek

Naslov: Igra na platformi Daydream

Avtor: Aljoša Omejc

Zmeraj večja priljubljenost in čedalje boljša izkušnja uporabe navidezne resničnosti (angl. virtual reality; VR) privablja k razvoju VR aplikacij. Cilj diplomske naloge je bil razviti igro Pot vikingov s pomočjo pogona Unity. Ker pa je to naša prva izkušnja z razvijanjem v Unity, se sproti z razvojem igre učimo.

Za ciljno platformo smo si izbrali Daydream, ki deluje na Android napravah. V diplomski nalogi si najprej pobližje pogledamo delovanje Daydream očal in kontrolerja. Nadalje se spoznamo z orodji, ki so bila uporabljena v razvoju, izdelamo načrt igre in se sprehodimo skozi njen razvoj. Na koncu naredimo še kvalitativno evalvacijo igre, kjer si s pomočjo ankete pogledamo uporabniško izkušnjo.

Ključne besede: navidezna resničnost, igre, Unity, Daydream.

Abstract

Title: Daydream platform based game

Author: Aljoša Omejc

The goal of this bachelor thesis was to develop a game Vikings Path with the help of Unity engine. Because this is our first experience with Unity development, we are still learning in the process. We have chosen Daydream for the target platform, which works on Android devices. In the thesis we take a closer look at how the Daydream headset and controller work. We get familiar with the tools we used in the development. After that we design a game plan and explain its development. At the end, we make a qualitative evaluation of the game, where we look at the user experience with the help of a survey.

Keywords: virtual reality, games, Unity, Daydream.

Poglavje 1

Uvod

Da se lahko začnemo pogovarjati o izdelovanju VR igre, moramo najprej spoznati, kaj pomeni VR, kako se to prenese na igre in kako sploh delujejo VR očala, ki nam prikazujejo navidezna okolja.

Navidezna resničnost je interaktivna računalniško generirana simulacija tridimenzionalne slike ali okolja. Z njo lahko z uporabo elektronske opreme interaktiramo, kot so očala z vgrajenimi zasloni ali rokavice s senzorji [1].

Uporabnik je potopljen v to virtualnost in potuje po teh svetovih, uporabljajoč grafično reprezentacijo lastnih rok in telesa, izvajajoč virtualne akcije – pobira predmete, jih premika, celo konstruira nove objekte in raziskuje nova področja [2]. Ključen del potopljenosti v navideznem svetu dosežemo s tem, da se svet premika z našim obračanjem glave, saj uporabnik s tem res dobi občutek, da je v novem svetu. Realizem še povečamo s simulacijo telesa, na primer dodamo uporabo rok. Za zaznavanje premikanja glave in rok uporabljamo senzorje za gibanje in žiroskop, ki jih v našem primeru najdemo v pametnem telefonu in kontrolerju.

Zaradi vse večjega trga navidezne resničnosti in zanimanja za izdelovanje iger, se nam to zdi primerna smer za nadaljevanje izobraževanja. Namen diplomske naloge je pridobiti nova znanja s področja VR, Unity in Daydream skozi razvoj igre. V igro smo poskušali vpeljati čim večje število različnih elementov igre, da dobimo splošni pregled pogona Unity. Za uvodom, v

drugem poglavju, si bomo ogledali uporabljena orodja. Posebno pozornost smo namenili Daydream platformi, ki zajema Daydream komplet programskih orodij (angl. software development kit; SDK), Daydream kontroler in Daydream očala. Nadalje je opisan igralni pogon Unity, predvsem njegove glavne komponente. V tretjem poglavju smo izdelali načrt igre, kjer smo podrobneje opisali ključne mehanike igre, kot so premikanje, uporaba orodij in obnašanje okostnjaka. Četrto poglavje zajema več delov implementacije načrta znotraj igralnega pogona Unity. Tukaj smo si pogledali vse od dodajanja modelov do shranjevanja igre. V zadnjem, petem poglavju je prikazana kvalitativna evalvacija igre skozi oči igralcev, ki so se preizkusili v igri Pot vikingov.

Poglavje 2

Uporabljena orodja

2.1 Daydream

Daydream je mobilna VR platforma [3]. Naznanjena je bila leta 2016 s strani Googla in deluje samo na podprtih napravah z operacijskim sistemom Android 7.1 in naprej. Podpira tri razvojne platforme:

- Android
- Unity
- Unreal.

Za delovanje potrebujemo Daydream očala ter kontroler in mobilno napravo, ki jo vstavimo v očala, kot vidimo na sliki 2.1. Od svoje glavne konkurence, ki je Samsung Gear VR [4], se razlikuje predvsem po tem, da podpira večji nabor naprav. Hkrati pa, če aplikacija ne potrebuje Daydream kontrolerja, je enostavno prenosljiva na Google Cardboard, s čimer lahko dosežemo veliko večji trg. Google je oznanil, da je prodal že več kot 10 milijonov Cardboard očal [5]. Google Cardboard je poceni komplet narejen iz kartona. Izdan je bil za spodbujanje zanimanja širše javnosti. Za obe platformi je izdan Google SDK za razvijanje v navidezni resničnosti. Paket nam omogoča komunikacijo s kontrolerjem, preizkušanje VR v urejevalniku Unity in interakcijo

kontrolerja z uporabniškim vmesnikom (angl. user interface; UI), ki ga implemenitramo znotraj Unity.

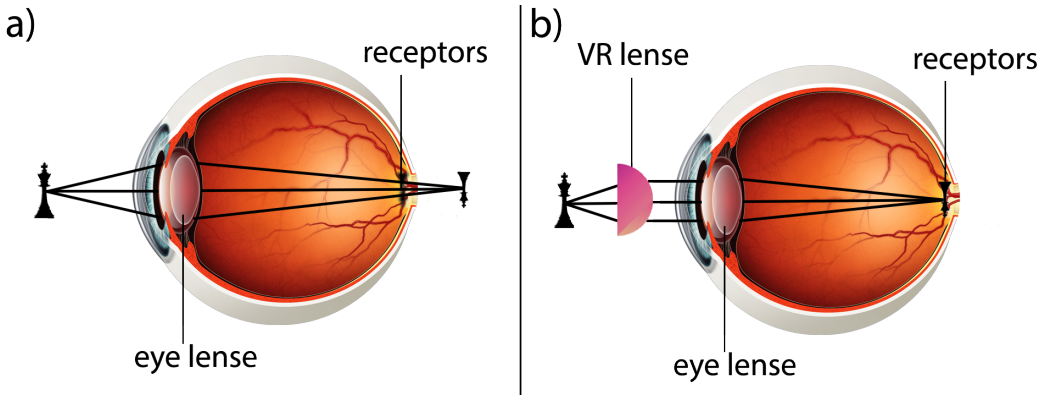
Slika 2.1: Daydream očala in kontroler

VR očala so novodobna različica že dolgo poznanega steroskopa (slika 2.2). Naredijo pregrado in vsakemu očesu kažejo samo eno malce zamaknjeno sliko. S tem posnemajo delovanje naših oči, saj v realnosti dosežemo zaznavanje globine in 3D efekta zato, ker imamo očesi na različnih mestih in vsako oko dobi malce drugačno sliko. Možgani pa znajo obdelati razlike med slikama in jih združiti v eno in s tem dobimo percepcijo globine [6].

Ker ljudje ne moremo gledati objektov, ki so tik pred našim obrazom, imajo očala vgrajene leče. Za boljše razumevanje funkcionalnosti leč, si najprej poglejmo delovanje leče v očesu. Ta z upogibanjem spreminja smer prihajajoče svetlobe, tako da jo osredotoča na receptorje, ki se nahajajo na zadnjem delu očesa. Te skrbijo za pretvarjanje svetolobe v koristne informacije. Naredimo enostaven poskus, pri katerem se osredotočimo na prst in ga potem počasi približujemo očesu. Pri tem na neki točki opazimo, da začnemo izgubljati osredotočenost. To se nam zgodi, ker se leče v našem

Slika 2.2: Stereoskop Holmes

očesu ne morejo več skrčit, kar prikazuje slika 2.3 a. Če pa uporabimo leče v očalih pa lahko vidimo na sliki 2.3 b, da leče spremenijo smer svetlobe, kar nam daje občutek, da so objekti oddaljeni in povečani.

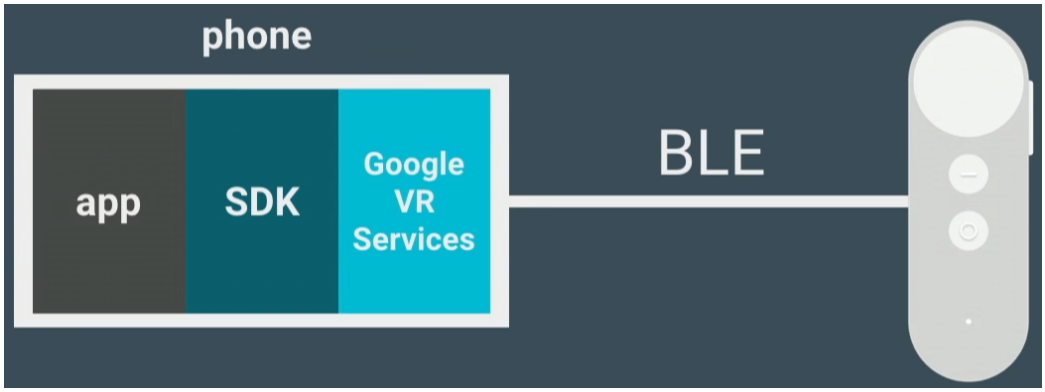


Slika 2.3: Levo preblizu in izgubimo fokus, desno VR leče usmerijo svetlobo

Komunikacija s kontrolerjem poteka preko brezžičnega protokola bluetooth. Kot vidimo na sliki 2.4, nam ni treba pisati kode za povezovanje in prenašanje podatkov, saj je za nas to vgrajeno že v Googlov VR SDK. Kontroler ima pet glavnih funkcij:

- Orientacija (angl. orientation) – pove v katero smer kaže kontroler v 3D prostoru in v igri jo uporabimo za usmerjanje žarka orodja laser.
- Žiroskop (angl. gyroscope) – pove, kako se kontroler obrača, specifično kotno hitrost (angl. angular speed) za vsako lokalno os (x, y, z) v radijanih na sekundo.
- Merilnik pospeškov (angl. accelerometer) – vrača sile pospeševanja za vsako lokalno os (x, y, z) v m/s^2. Nanj vpliva tudi gravitacija, zato če ga postavimo na mizo, bo imela os y še zmeraj približno vrednost 9,8.
- Sledilna ploščica (angl. trackpad) – spodaj ima gumb, zato zaznava dotik ali pritisk. Vrne nam (x,y) kordinate in nam s tem pove lokacijo prsta na ploščici.
- Gumba – na voljo imamo dva gumba:
  - aplikacijski
  - domov.

  Domov je rezerviran za izhod iz aplikacije. Aplikacijskega pa v našem primeru uporabljamo za menjavo med orodji.



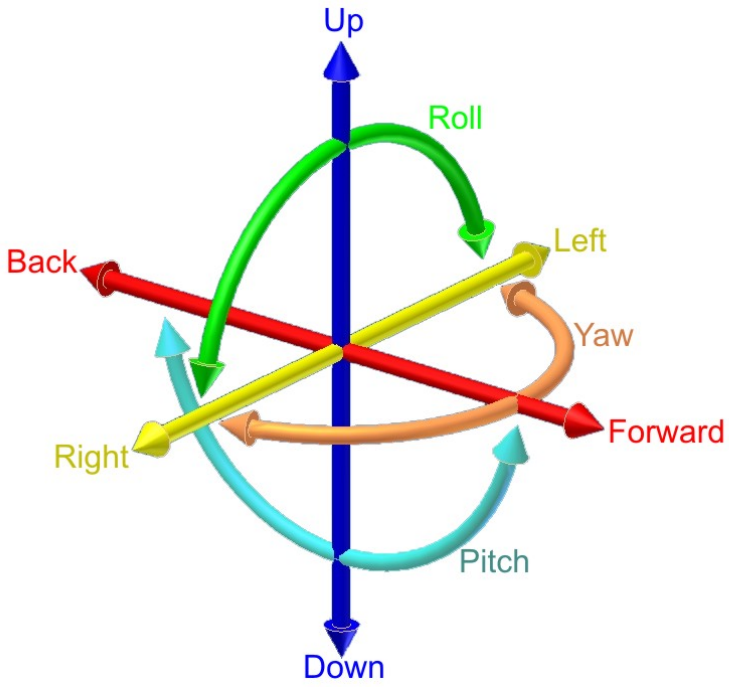
Slika 2.4: Kako deluje povezava s kontrolerjem

Za lažjo uporabo orodja imamo na voljo zbirko predstavitvenih programov imenovanih Daydream elementi [7]. V njih imamo prikazane koncepte in

dobre prakse za razvoj kvalitetnih VR izkušenj. Glavne mehanike v vsakem elementu so enostavno nastavljive in preproste za prenos v svojo aplikacijo. Elementi pokrijejo nasledje VR koncepte:

- Premikanje (angl. locomotion). Izjemno pomebna točka razvoja VR iger je obvladovanje slabosti uporabnika. Posebno moramo paziti pri razvoju mobilnih VR iger, saj ima večina mobilnih naprav na voljo zaslone, ki osvežujejo samo 60 sličic na sekundo (angl. frames per second), kar je prepočasno, da bi možgane prepričali o resničnosti novega sveta. Največkrat se pomanjkanje števila sličic in posledično slabost uporabnika opazi pri premikanju kamere. Na voljo imamo tri različne prakse, ki zmanjšujejo slabost uporabnika: preslikava, prodiranje in sledilna kamera.
  - Preslikava (angl. teleportation) se uporablja v igrah s prvoosebno perspektivo, dovoli nam skoraj takojšni premik na ciljno lokacijo. S tem zmanjšamo slabost, ker premikamo kamero samo, ko se preslikamo.
  - Prodiranje (angl. tunneling) se uporablja v igrah s prvoosebno ali tretjoosebno perspektivo. Medtem ko se premikamo, pokažemo samo to, kar je pred nami, ostalo pa nadomestimo s stabilnim ozadjem. S tem dobimo občutek gledanja televizije ali računalniškega zaslona, saj se spreminja samo vsebina na zaslonu in ne okolica.
  - Sledilna kamera (angl. chase camera) prikazuje uporabo premikanja kamere v tretji osebi. Na daljše razdalje kamera sledi igralcu, na krajše pa ostane na istem mestu in s tem zmanjšuje slabost.
- Interakcija z objekti (angl. object interaction). Premikanje orodij znotraj igre s premikanjem kontrolerja, ki ga držimo v roki je zahtevna naloga. Zato imamo že vnaprej pripravljene modele. Na voljo imamo modele za mahanje, streljanje, metanje in manipuliranje predmetov. V modelih uporabljamo vseh 6 prostostnih stopenj (angl. degrees of

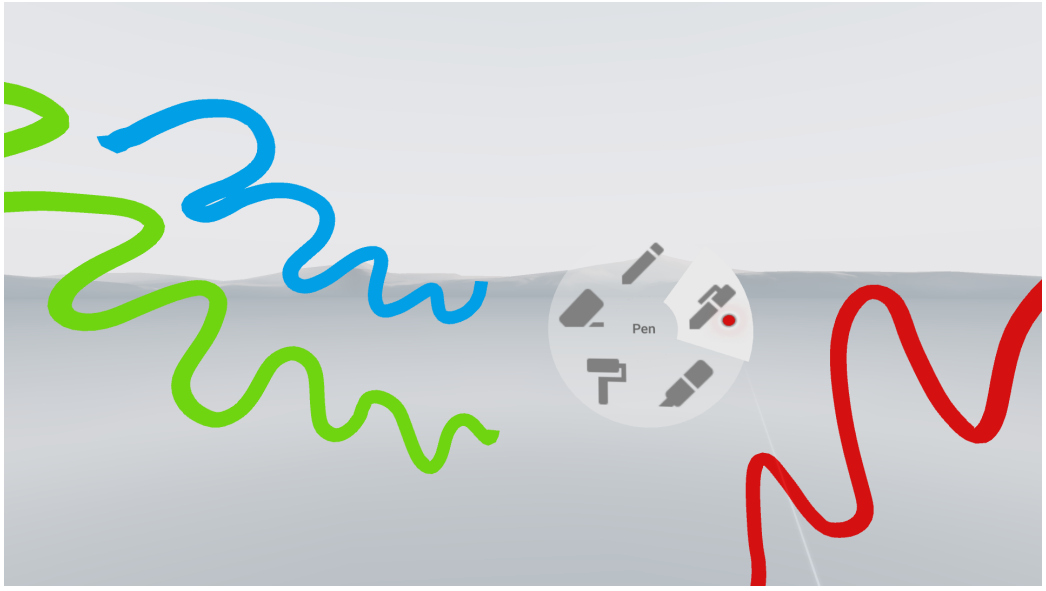
freedom; DOF) (slika 2.5). Kontroler nam vrača točne podatke samo za orientacijo, da dosežemo 6 DOF, model izračuna približek premika (angl. translation) s pomočjo žiroskopa, merilnika pospeškov in magnetometra. Seveda je to le približek, da bi dosegli boljše rezultate, bi morali uporabljati dodatne senzorje, ki pa jih Daydream platforma zaenkrat ne omogoča.



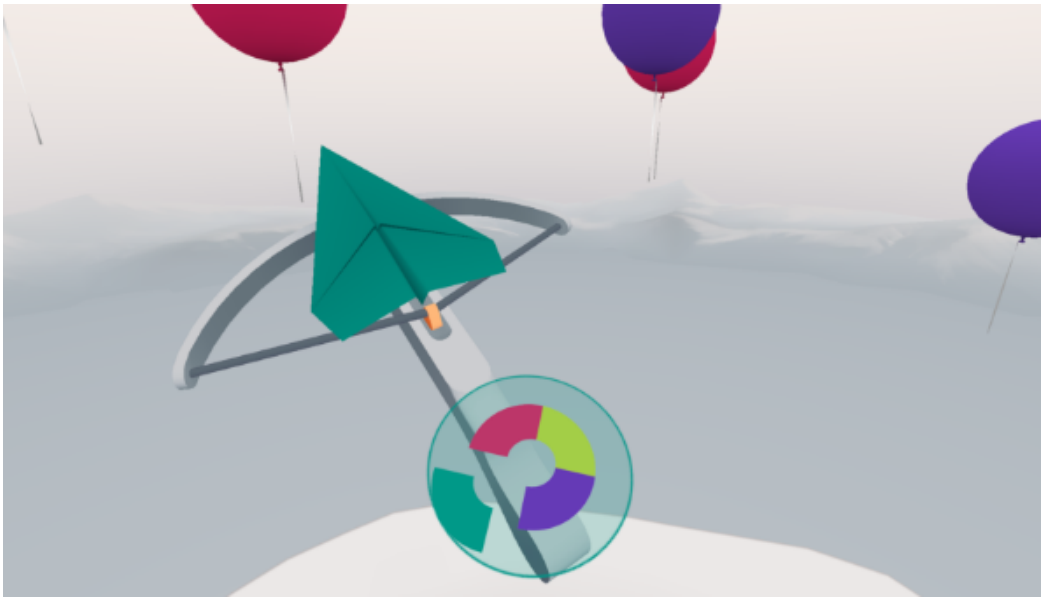
Slika 2.5: Šest prostostnih stopenj v 3D prostoru

- Upodabljalnik (angl. renderer) in luči (angl. lights). Pri razvijanju VR aplikacij za pametne telefone se moramo posvetiti učikoviti porabi njihovih virov. Posebaj pazimo na grafično kartico, saj moramo prikazovati v visoki ločljivosti in dveh pogledih. Zato je Google izdal Daydream Renderer, ki omogoča mobilno VR optimizacijo osvetlitve in senčilnikov. Program prikazuje uporabo funkcij, kot so osvetlitev vsakega slikovnega elementa, dinamične sence, odsev in zrcalne poudarke v realnem času.

- Meniji. Na voljo imamo samo dva gumba in sledilno ploščico s klikom. Vendar je en rezerviran za izhod iz aplikacije. Ostaneta samo še en gumb in sledilna ploščica s klikom. Ker je to premalo, za večino aplikacij uporabljamo navidezne menije. Pripravljene imamo klik (angl. click) (slika 2.6), drsni (angl. swipe) (slika 2.7) in zvezdni (angl. constellation) (slika 2.8) meni.



Slika 2.6: Primer klik menija



Slika 2.7: Primer drsnega menija

2.2 Unity

Igralni pogon Unity [8] je primarno namenjen izdelovanju 3D iger. Lahko pa ga uporabljamo tudi za izdelovanje 2D iger ali navadnih aplikacij za telefon in računalnik. V preteklosti je bil na voljo v treh programskih jezikih, vendar so z Unity 5 in Unity 2017 odstranili podporo za Boo in UnityScript. Zdaj je primarni jezik C#. Za uporabo pogona je narejen grafični vmesnik (urejevalnik Unity), v katerem lahko uporabljamo funkcijo primi in spusti za dodajanje objektov ali skript. Vgrajenih ima še kar nekaj uporabnih orodij:

- Igralni način (angl. play mode) – nam omogoča igranje igre znotraj urejevalnika. Vse spremembe, ki smo jih naredili v igralnem načinu se izgubijo. Zato je odličen za testiranje različnih nastavitev in scenarijev, saj nas ni strah, da bi kaj pokvarili.
- Ustvarjalec terena (angl. terrain builder). Z njim lahko enostavno dodajamo teren, drevesa in kamne, kar znotraj urejevalnika, in ne potrebujemo ostalih programov za modeliranje.

Slika 2.8: Primer zvezdnega menija

- Vgrajena trgovina sredstev (angl. asset store). Na njej imamo ogromno že pripravljenih modelov, tekstur, animacij in različnih dodatkov za urejevalnik, s katerimi si lahko olajšamo izdelovanje igre.
- Sistem delcev (angl. particle system) nam omogoča prikaz tekočih in neoprijemljivih stvari, kot so oblaki, voda, ogenj in ipd.
- ipd.

Vsaka igra ima vsaj eno sceno, na katero dodajamo igralne objekte. Vsak objekt je sestavljen iz komponent, ki jih imamo že pripravljene v pogonu. Podrobno si poglejmo nekaj komponenet, da bomo lažje razumeli delovanje igre.

Skripta (angl. script)

Skripte so osnoven del vsake igre, z njimi vodimo igro in dodajmo svojo funkcionalnost. To delamo tako, da beremo in spreminjamo ostale komponente, kar naredimo s funkcijo GetComponent, s katero dostopamo do komponent,

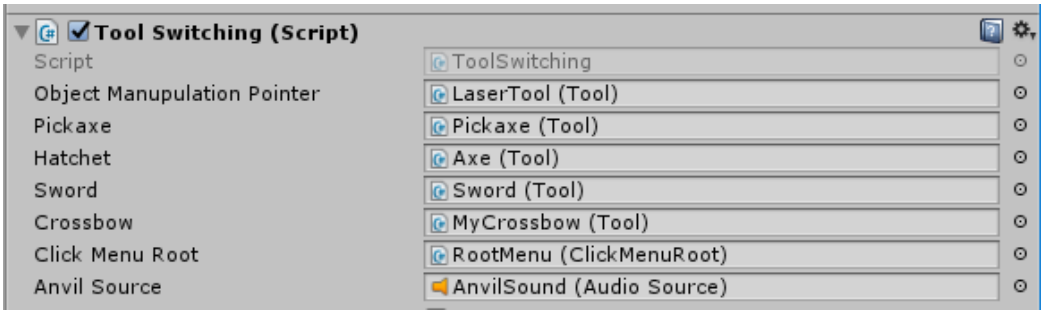
ki so na istem objektu kot naša skripta (koda 2.1).

Koda 2.1: Dostop do komponente kamera

```
Camera camera = gameObject.GetComponent<Camera>();
```

Seveda si velikokrat želimo spreminjati komponente in s tem tudi skripte na drugih igralnih objektih. Za to imamo na voljo dve različni metodi dostopanja. Prva je referenca v urejevalniku (slika 2.9), pri kateri v naši skripti naredimo javno spremenljivko GameObject. Nato se nam v urejevalniku pokaže prostorček, kamor lahko povlečemo in izpustimo objekt, do katerega želimo dostopati. Prednost je, da imamo že shranjeno referenco do željenega objekta, kar nam omogoča takojšni dostop.

V drugem načinu pa uporabimo funkciji GameObject.Find(name) ali GameObject.FindWithTag(tag). Obe funkciji delujeta na isti način, kjer gresta čez vse objekte v sceni in najdeta igralni objekt, ki ima enako vrednost kot poslani parameter. S tem da ena primerja oznako objekta, druga pa ime. Ta način uporabimo, če objekt naredimo dinamično, saj ga potem nimamo na voljo v urejevalniku.



Slika 2.9: Primer dodanih referenc v urejevalniku

Transformacija (angl. transform)

Transform uporabljamo za shranjevanje in manipuliranje podatkov o poziciji, rotaciji in skaliranju objekta.

Kamera (angl. camera)

Kamera zajame in prikaže svet igralcu. Poznamo dve vrsti projekcij kamere. Prva je perspektivna in nam prikazuje globino in simulira pogled sveta iz naše perspektive. Druga je ortografska, pri njej izgubimo občutek globine, zato se uporablja za 2D igre in menije v 3D.

Trkalnik (angl. collider)

Trkalnik definira obliko objekta za fizične trke. Lahko pa ga uporabljamo tudi kot sprožilec. Takrat ne simulira fizike, ampak nam pove, kdaj je drug objekt v njegovem prostoru.

Kontroler igralniškega lika (angl. character controller)

Igralčev kontroler nam dovoli premikanje omejeno s trki. Zunanje sile ne vplivajo na komponento in sama poskrbi za fiziko premikanja in trkov. Za premik ji podamo smer in hitrost.

Sistem dogodkov (angl. event system)

Sistem dogodkov je način pošiljanja dogodkov vhodnih naprav objektom v aplikaciji. Lahko je to miška, tipkovnica, igralna palica ali pa v našem primeru Daydream kontroler.

Luč (angl. light)

Komponenta luč se uporablja za osvetljevanje prostora. Lahko nastavimo smer, intenzivnost in barvo. Lahko izbiramo med štirimi vrstami luči:

- Točkasta luč (angl. point light) je točka v 3D prostoru, od koder pošilja svetlobo v vse smeri enakomerno. Moč svetlobe pa upada z razdaljo, dokler ne pridemo na doseg, ki smo ga nastavili na komponenti.
- Omejena točkasta luč (angl. spot light) nam že iz imena pove, da deluje podobno kot točkasta luč. Vendar ker je omejena, sveti samo pod določenim kotom in ne v vse smeri. Uporablja se za umetne luči, kot so svetilka, avtomobilske luči ipd.
- Območna luč (angl. area light) je določena s pravokotnikom v 3D prostoru. Svetlobo pošilja enakomerno po površini, ampak samo iz ene strani pravokotnika. Ker zahteva ogromno procesorskega časa, jo moramo izračunati vnaprej.
- Usmerjena luč (angl. directional light) se uporablja primarno za simuliranje sonca in lune. Predstavljamo si jo lahko kot neskončno oddaljeno svetlobo, ki nima točno določenega izvora. Vsi predmeti v sceni so osvetljeni iz iste smeri.

Navigacijska mreža (angl. navigation mesh; NavMesh)

Komponente za navigacijsko mrežo niso na voljo v standardnem Unity in jih moramo posebej naložiti. Uporabljajo se za premikanje umetno — inteligenčnega lika (angl. non-player character, NPC) po okolju. NavMesh Agent komponento dodamo na NPC in ji nastavimo parametre, kot so hitrost premikanja in obračanja. Agentu določimo novo lokacijo, vendar si sam izbere pot in pri tem poskrbi za izogibanje oviram in ostalim agentom.

Avdio vir (angl. audio source)

Avdio vir se uporablja za predvajanje posnetkov glasbe. Na komponenti je veliko uporabnih nastavitev, ki nam prihranijo čas. Med drugim lahko nastavimo:

- glasnost

- višino
- predvajanje na levo, desno ali obe slušalki
- igraj na začetku
- predvajaj v neskončnost
- 2D ali 3D zvok
- ipd.

Na sliki 2.10 vidmo izgled programa Unity.

Slika 2.10: Program Unity

2.3 Git

Git [9] je sistem za nadzorovanje verzij kode. V procesu razvoja nam omogoča združevanje in primerjanje kode večih ljudi. V primeru napake pa tudi povrnitev nazaj na prejšnje verzije. V našem projektu je bil predvsem uporabljen

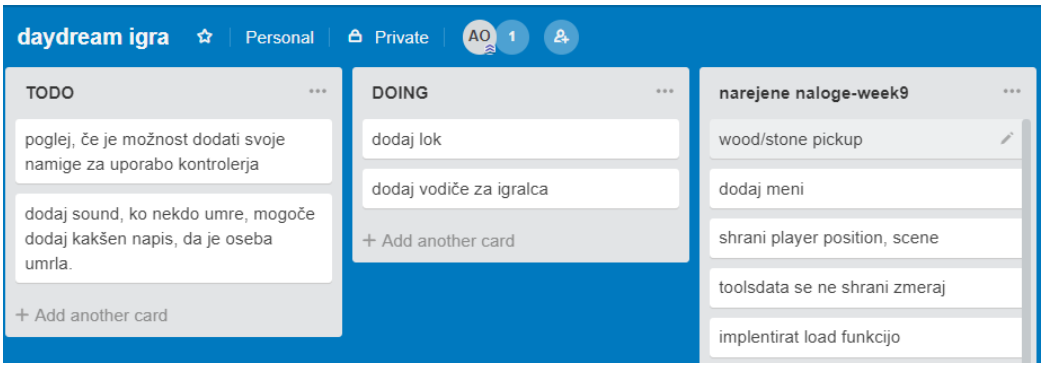
za shranjevanje verzij na splet, kot varnostna kopija, in za povrnitev na prejšnjo verzijo v primeru težav.

2.4 Trello

Trello [10] je spletna aplikacija za upravljanje projektov. Pomaga nam pri planiranju in organizaciji dela. Ker je razvoj igre trajal več mesecev in smo bili dogovorjeni za redna poročila, smo si po vsakem poročilu zadali naloge in jih sproti uvrščali v tri kategorije:

- odprte naloge
- naloge v implementaciji
- že narejene.

To nam je pomagalo pri hitrejšem razvoju igre in pisanju sprotnih poročil. Na sliki 2.11 vidimo primer dodanih nalog, ki jih lahko preprosto povlečemo in spustimo, ko jih želimo prestaviti v drugo kategorijo.



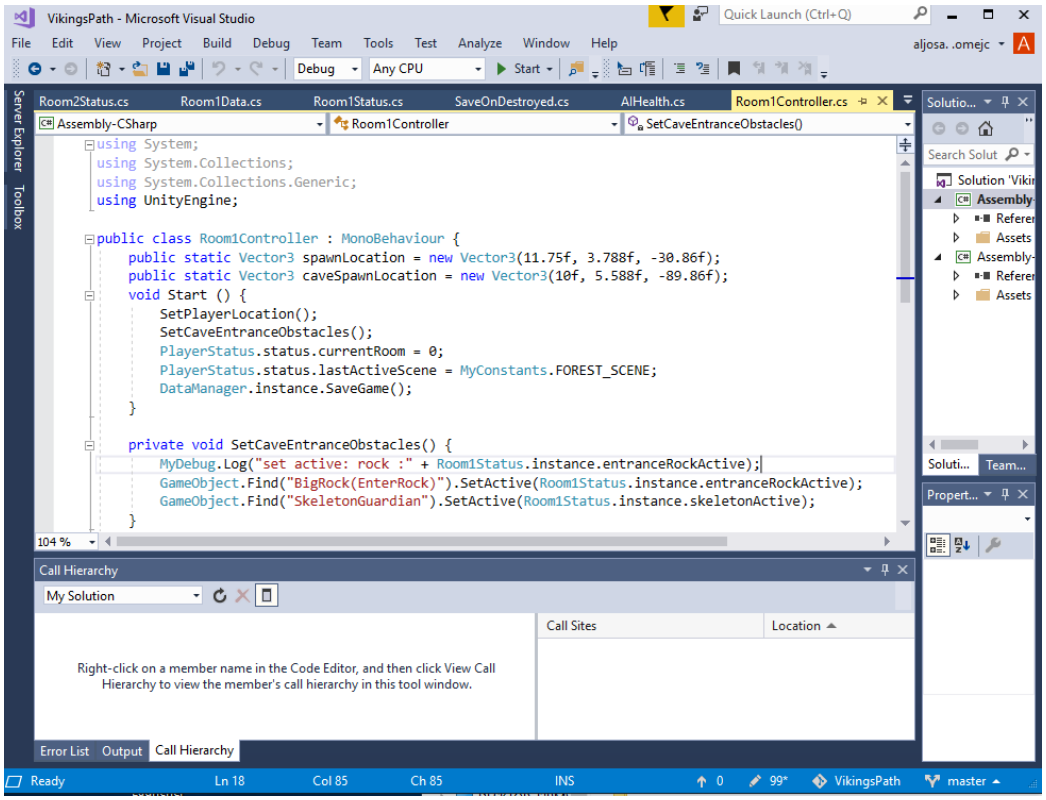
Slika 2.11: Primer upravljanja z nalogami v aplikaciji Trello

2.5 Microsoft Visual Studio

Microsoft Visual Studio [11] je integrirano razvojno okolje (angl. integrated development environment; IDE), ki se uporablja za pisanje kode in je privzet

IDE za Unity 2017. V našem primeru smo ga uporabljali za pisanje skript (angl. script), primer si lahko pogledamo na sliki 2.12. Skripte se uporabljajo za premikanje objektov, shranjevanje podatkov, itd. Visual Studio ima vgrajenih kar nekaj funkcionalnosti, ki pomagajo pisati in urejati kodo, kot so:

- poudarjanje sintakse
- dokončevanje kode
- povratne informacije o napakah
- spreminjanje spremenljivk na celotnem projektu
- ipd.



Slika 2.12: Primer pisanja skripte v programu Microsoft Visual Studio

Poglavje 3

Načrtovanje igre

3.1 Koncept igre

Igro postavimo v čas vikingov. Igralec bo igro začel na površju. Nabrati mora surovine in si na obrtni postaji izdelati orožja. Nato mora najti pot do kralja, ki se skriva v podzemlju, saj mu je ta ukradel vredno lastnino. Na poti mora premagati uganke in pošasti, ki so mu v napoto. Igralec ima na voljo pet orodij:

- sekiro
- kramp
- laser
- meč
- samostrel in puščice.

Sekiro in kramp ima igralec na voljo že na začetku in z njima lahko nabira kamen, les in železo. Z laserjem igralec odpre meni za izdelovanje orožja in v naslednjih sobah lahko z njim prenaša ključe. Če nam življenjske točke padejo na nič, se pojavimo, kjer smo igro začeli. Igre je konec, ko ubijemo kralja in s tem dobimo nazaj svojo lastnino.

3.2 Načrtovanje sob

Igra ima štiri glavne sobe:

- zunanji svet
- vhod v notranji prostor
- okostnjakova dvorana
- kraljeva sobana.

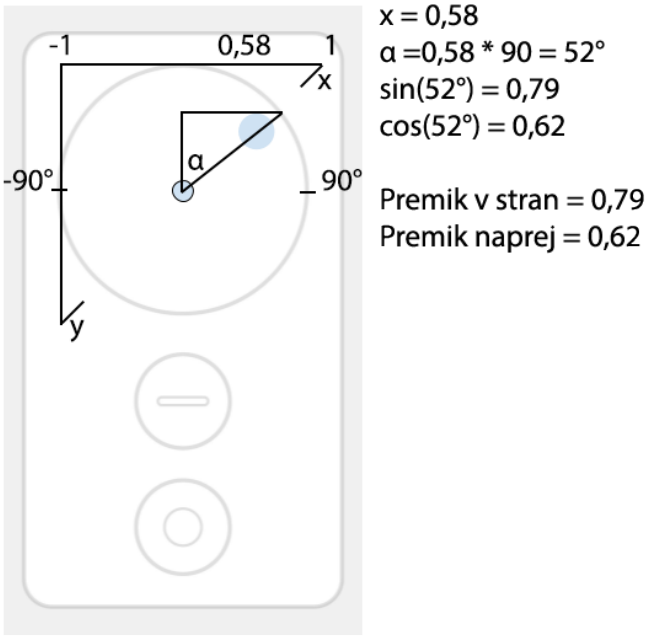
Zunanji svet in vhod v notranji prostor sta namenjena predvsem temu, da se igralec navadi na premikanje in uporabo orodij. V tem času mu dajemo čim več informacij, da bo potem čim bolj samostojno reševal naslednje sobe. Ko ubijemo okostnjaka, ki varuje vhod v notranji prostor, lahko vstopimo v Okostnjakovo dvorano. Cilj dvorane je, da najdeš tri ključe in jih postaviš na pravi stebriček pred vrati. Ključi se skrivjajo pod stebri, kjer ima vsak steber narisan znak. Za pomoč so nad vrati, ki vodijo naprej, narisani znaki, ki se ujemajo s stebri, ki skrivajo ključe. Težavnost pa je dodana s tem, da si je veliko znakov zelo podobnih in če podiraš napačne, se pojavi zmeraj več sovražnikov. Zadnja soba je kraljeva sobana, v kateri moraš moraš priti čez ognjeni zid in nato premagati kralja. Da prideš čez ognjeni zid, moraš postaviti kamne, ki ležijo po tleh, v obliko vrat. Ko premagaš kralja, dobiš nazaj svojo lastnino in igre je konec.

3.3 Mehanika igre

3.3.1 Premikanje

Implementacijo premikanja smo naredili s klikanjem po sledilni ploščici. Kontroler nam ob pritisku po sledilni ploščici vrne točko (x, y) na razponu od 0 do 1. Točko pretvorimo na razpon -1 do 1, kjer je 0 sredina. Koordinato x pretvorimo na razpon kota [-90,90] stopinj, iz katerega potem dobimo smer premikanja. Da se igralec zmeraj premakne za isto razdaljo, si pomagamo

s kotnima funkcijama *sin* in *cos*. Koordinata y pa nam pove, če se igralec premika naprej ali nazaj. Za lažje razumevanje si poglejmo sliko 3.1. Pri računanju upoštevamo tudi, če je pogon spustil kakšen okvir (angl. frame), da se igralec lepše premika. V Unity smo naredili 3D igralni objekt kapsula, kateremu smo dodali komponento CharacterController.



Slika 3.1: Primer klika na sledilno ploščico

3.3.2 Uporaba orodja

Za uporabo orodja smo se odločali med dvema načinoma. Prvi način je, da se ob pritisku gumba zgodi animacija, na primer naredimo zamah. Prednost tega je, da točno vemo, kdaj smo v zamahu in s tem lažje vemo, kdaj upoštevamo trk z drugim objektom. Drugi način je, da premikamo orodje tako, kot premikamo kontroler v roki, kjer je glavna prednost, da dobimo občutek, da orodje držimo v svojih rokah. Se pa poveča težavnost zaznavanja trkov, saj se nam na primer ne sme zgoditi, da zamahnemo od spodaj

navzgor in s tem posekamo drevo. Izbrali smo drug način, saj daje boljši občutek z vidika uporabniške izkušnje. Za implementacijo premikanja roke smo si pomagali z že narejenim primerom meča, ki je na voljo v Google VR SDK.

3.3.3 Obnašanje okostnjaka

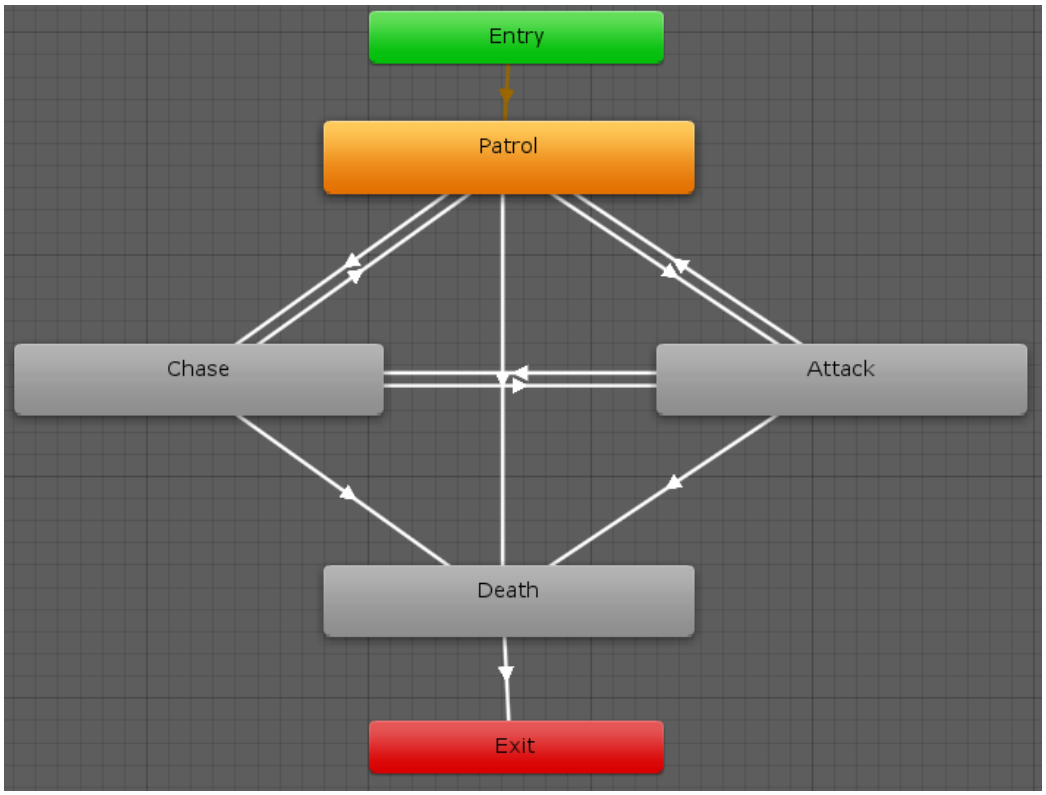
Obnašanje okostnjaka razdelimo na štiri stanja:

- napadanje
- lovljenje
- patroliranje
- smrt.

Na sliki 3.2 vidimo, kako lahko prehajamo med stanji. Med stanji menjamo s simuliranjem sluha in vida okostnjaka, glede na položaj igralca. V primeru, da je v dosegu meča, gremo v stanje napadanje. Sicer pogleda, če je zadosti blizu, da ga sliši ali vidi in nato preide v stanje lovljenja. Drugače pa patrolira med prej pripravljenimi točkami, ki smo jih realizirali z nevidnimi igralnimi objekti. Za lažje razumevanje si lahko pogledamo kodo 3.1.

Koda 3.1: Preklopi med stanji okostnjaka

```
private void ChangeAnimatorState() {
        if (IsPlayerInSight(Const.SWORD_RANGE)) {
                StartAttacking();
        } else if (DistanceToPlayer() < Const.
           ↪ HEARING_RANGE * IncreaseSenses()) {
                StartChasing();
        } else if (IsPlayerInSight(Const.EYE_RANGE *
           ↪  IncreaseSenses())) {
                StartChasing();
        }else {
                StartPatrolling();
        }
}
```



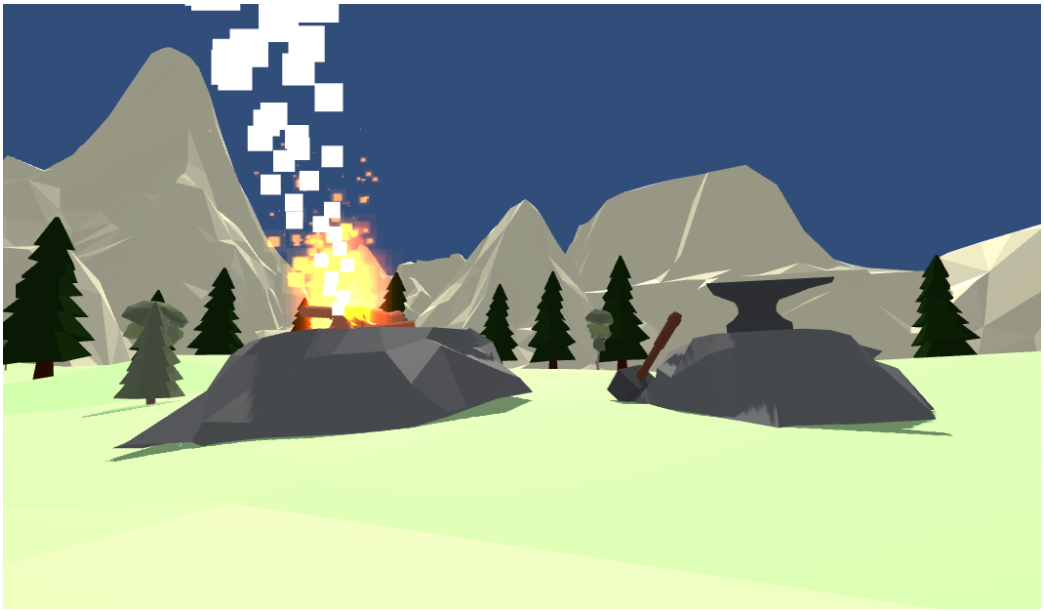
Slika 3.2: Stanja okostnjaka v Unity

Poglavje 4

Implementacija igre

4.1 Zunanji svet

Ko smo naredil načrt igre, smo se lotili implementacije v Unity. V prvem koraku je bil cilj, da se lahko igralec premika po terenu in uporablja orodje s pomočjo kontrolerja. Iz Unity trgovine s sredstvi smo dodali testno okolico in v projekt vpeljali Google VR SDK, ki nam omogoča komunikacijo s kontrolerjem. Kako smo implementirali premikanje si lahko pogledamo v poglavju 3.3.1, kako pa uporabo orodij v poglavju 3.3.2. Za premikanje smo vpeljali še funkcijo prodiranje (angl. tunneling) v poglavju 2.1, ki jo lahko vklopimo v meniju opcij. Ob začetku igre nimamo na voljo vseh orodij, zato smo dodali obrtno postajo (slika 4.1). Izbiranje orodja, ki ga želimo narediti smo implementirali s klik menijem (poglavje 2.1). Spremeniti je bilo potrebno način odpiranja menija, saj smo imeli aplikacijsko tipko že zasedeno. To smo rešili tako, da smo ob kliku na sredino ploščice pogledali, če se laser dotika obrtne postaje in takrat odprli meni. Igralec izbere orožje, ki ga želi narediti. V primeru, da ima dovolj surovin, mu orožje dodamo v roke. Drugače pa mu z rdečo barvo obarvamo surovine, ki mu primankujejo.



Slika 4.1: Obrtna postaja v zunanjem svetu

4.2 Uporaba več scen

Razdelitev na več scen ni nujna, vendar ker imajo pametni telefoni slabše specifikacije, pomaga pri manjši porabi virov naprave in za boljšo organiziranost hierarhije scene. Sami smo igro razdelili na meni, zunanji in notranji svet. Objekte, ki jih želimo obdržati med menjavo scene, moramo označiti s funkcijo DontDestroyOnLoad, saj se vsi ostali avtomatsko uničijo. V naši igri želimo obdržati objekte, ki nam držijo podatke, kot so življenjske točke in druge objekte, ki se podvajajo, na primer uporabniški vmesnik. Prehod med zunanjim in notranjim svetom se zgodi, ko stopimo v bližino odprtine, kjer imamo postavljen preprost sprožilec (poglavje 2.2). Med čakanjem, da igra naloži uporabniku sceno, pokažemo nalagalni zaslon (slika 4.2), na katerem lahko vidi napredek nalaganja.

4.3 Uporabniški vmesnik

Implementacijo uporabniškega vmesnika smo naredili z že pripravljenimi Unity UI objekti. Pri vseh smo uporabili UI platno, na katerega postavimo ostale elemente, na primer gumbe. Dodali smo novo kamero z ortografsko projekcijo in jo uporabljamo samo za izrisovanje UI elementov. Na kameri nastavimo globino na največjo vrednost, kar nam omogoča, da se UI vedno nariše pred ostalimi objekti.

Uporabniški vmesnik smo uporabili na tri različne načine. Najprej za prikazovanje stanja igralca. Nato kot nalagani zaslon, kjer smo prekrili celotni pogled. In nato še pri začetnem meniju (slika 4.3). Tukaj je glavna razlika, da se morajo gumbi odzivati na naše akcije. K sreči je za nas že pripravljen igralni objekt GVREventSystem, ki je deluje na vrhu Unity sistema dogodkov (angl. event system) (poglavje 2.2).

Slika 4.2: Nalagalni zaslon igre

Slika 4.3: Začetni meni

4.4 Notranji svet

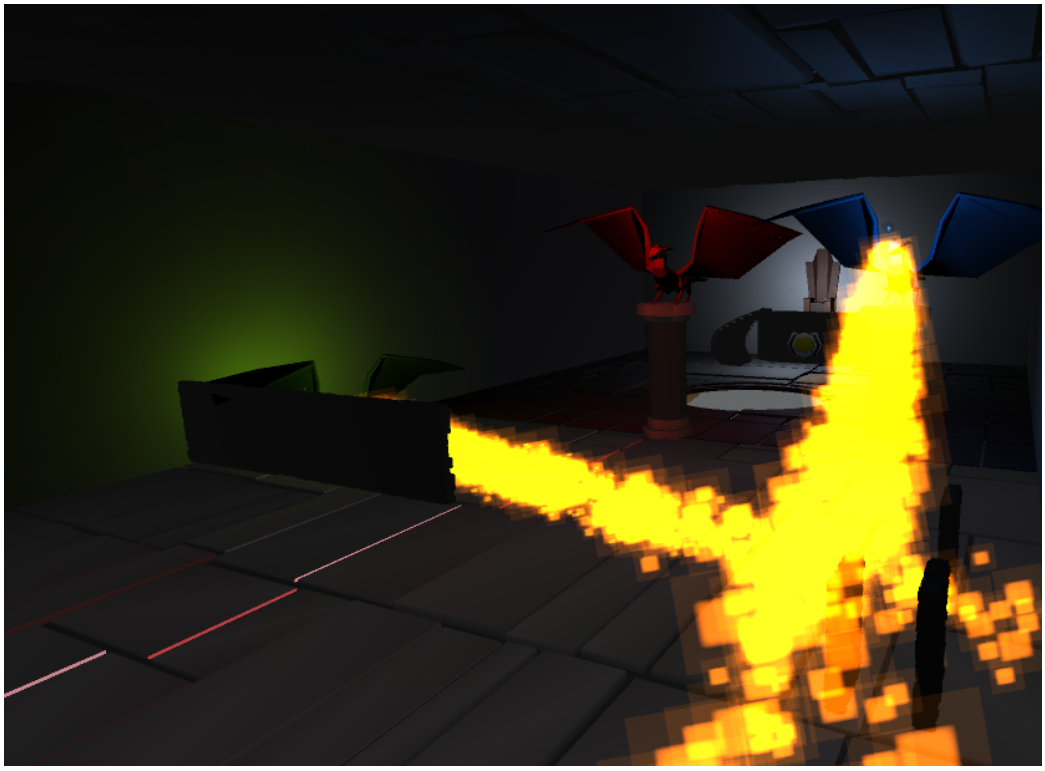
Po uspešni razdelitvi na več scen, smo testne grafike zamenjali z novimi, ki jih je naredila Sara Domnik za svojo diplomsko nalogo na Inštitutu in akademiji za multimedije, Ljubljana. Na nove objekte smo pripeli komponente, kot so trkalnik in rudarska skripta. Največ časa smo v notranjem svetu porabili za vpeljevanje okostnjaka (slika 4.4). Model in animacije smo našli v Unity trgovini [12]. Več o obnašanju okostnjaka si lahko preberete v poglavju 3.3.3. Naslednji korak so bili ključi za odpiranje vrat kraljeve sobane. Dodali smo jih pod stebre in jih lahko prenesemo z laserjem na stebričke pred vrati. S sprožilci zaznamo (poglavje 2.2), če ključ ustreza in je na pravem mestu. Ko postavimo vse tri ključe na pravo mesto pridemo v kraljevo sobano. Tukaj moramo priti mimo zmajev, ki bruhajo ogenj (slika 4.5). Prva dva ustrelimo s samostrelom, da pridemo mimo zadjega pa moramo premakniti majhne plošče, da zakrijemo ogenj. Tega smo naredili s pomočjo komponente sistema delcev (angl. particle system), kjer ti vsak delec, ki te zadane odbije

življenjske točke.

Slika 4.4: Okostnjakova dvorana

4.5 Shranjevanje igre

Shranjevanje je posnetek trenutnega stanja igre [13]. Glavna razloga za uporabo v naši igri sta možnost kasnejšega nadaljevanja in zavarovanja napredka igralca pred morebitnim sesutjem igre. Igro shranjujemo avtomatsko ob ključnih dogodkih, na primer dodajanje novega orodja, zaprtje aplikacije itd. Najlažji način za shranjevanje je, da uporabimo serializacijo, kar pomeni, da shranimo trenutno stanje objektov v primeren format za shranjevanje na disk. Sami smo izbrali format JSON, saj je prilagodljiv in ima relativno majhno velikost datotek. Primer shranjene datoteke je na voljo v kodi 4.1. V njej vidimo vrstico z nizom "hashOfContent", katerega uporabljamo kot prvo linijo obrambe pred spreminjanjem datoteke. Deluje tako, da naredimo razpršilni algoritem (angl. hash) celotne vsebine podatkov in to shranimo v vrstico "hashOfContents". Potem, ko podatke nalagamo, naredimo ponoven razpršilni algoritem podatkov in jih med seboj primerjamo. Če sta enaka,



Slika 4.5: Zmaji v kraljevi sobani

potem datoteka ni bila spremenjena. Seveda, kot smo že zgoraj omenili, je to le prva linija obrambe; če želimo podatke res zaščiti, jih moramo hraniti na strežniku in ne lokalno.

Koda 4.1: Primer shranjenih JSON podatkov o stanju igre

```
{
"room1Data": {
"entranceRockActive": true,
"skeletonActive": true
},
"room2Data": {
"greenKey": false,
"blueKey": false,
"redKey": false
```

```
},
"playerData": {
"health": 100,
"wood": 80,
"stone": 80,
"currentTool": 0,
"numberOfArrows": 0,
"iron": 0
"lastActiveScene": "ForestScene",
"playerTransform": {
"positionX": 20.0951,
"positionY": 3.95898,
"positionZ": -29.4729
}
},
"hashOfContents": "348771af5d2a6360f8bde9d973ef
19c7af799c3b562d4eff453d15fb139aa6d6"
}
```


4.6 Luči in zvok

Luči in zvok sta osnovna gradnika vsake igre, saj ogromno prispevata k vzdušju. Prikazovanje realističnih luči je računsko zelo zahtevna operacija, saj vir svetlobe ne sveti samo neposredno na okolico, ampak se od nje tudi odbija. Pri odboju moramo upoštevati tudi, da vsi materijali ne odbijajo svetlobe na enak način, zato v igrah poskušamo narediti čim boljši približek. Mobilne naprave imajo splošno slabše grafične kartice in imajo težavo z računanjem velikega števila luči v realnem času. Zato smo v naši igri večinoma izkoristili že vnaprej izračunane luči.

Za osvetljevanje celotnega prostora smo uporabili ambientno luč. V zunanjem svetu smo izbrali svetlo barvo, ki nam daje občutek dneva in v notranjih prostorih temno, za posnemanje vzdušja jame. Za dodatno osvetlitev smo v zunanjem svetu dodali usmerjeno lučjo (angl. directional light), ki posnema obnašanje sonca. V notranjem svetu pa različne barve točkaste luči (angl. point light) za usmerjanje uporabnikove pozornosti.

Zvok smo vpeljali s komponento avdio vir (angl. audio source) (poglavje 2.2). Uporabili smo jo za predvajanje glasbe v ozadju in zvočne efekte za stvari, kot so uporabljanje orodij, oglašanje okostnjaka in prasketanja ognja. Glasbo v ozadju smo uporabljali glede na to, v katerem prostoru se nahajamo. Na primer v zunanjem svetu predvajamo skladbo posneto v gozdu, na kateri posnemamo zvoke narave, kot so veter in ptičje petje. Ker si želimo, da se uporabnik res vživi v igro, smo komponento nastavili na 3D zvok in jo dodali na objekt, ki predvaja določen zvok. V praksi to pomeni, da se s premikanjem ali oddaljevanjem od objekta glasnost za vsako uho spreminja. S tem imitiramo zvok v realnosti. Vse skladbe in zvočne efekte smo prenesli iz spletne strani Freesound [14].

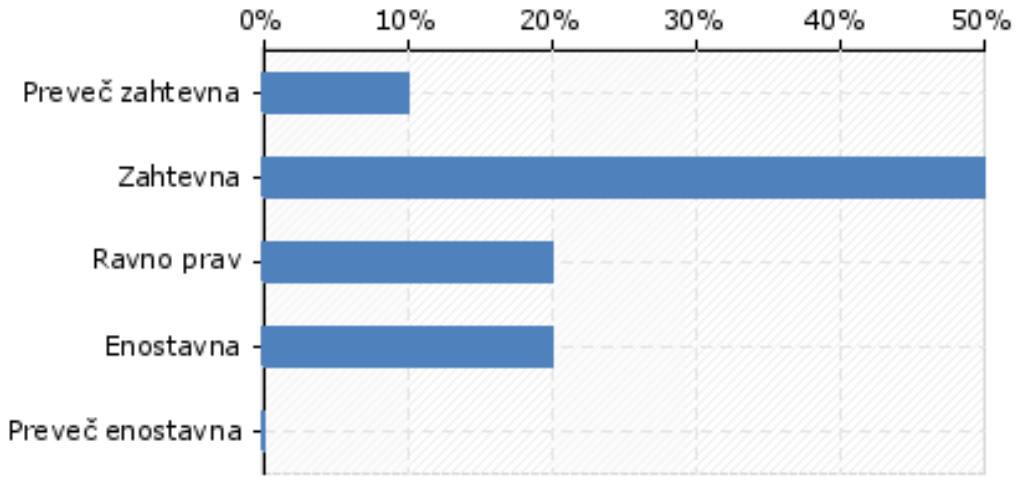
Poglavje 5

Evalvacija uporabniške izkušnje

Za ocenitev uporabniške izkušnje smo našli deset igralcev, ki so igro najprej preizkusili, nato pa so izpolnili anketo (priloga A). Hkrati smo opazovali igralce pri igranju in s tem poskušali pridobiti bolj objektivne podatke o njihovi izkušnji (priloga B).

Igro je igralo 9 moških in 1 ženska, povprečna starost je bila dvaintrideset let. Med njimi jih je osem že prej poskusilo VR očala. Igralci so, razvidno iz slike 5.1, v povprečju igro ocenili za zahtevno (rezultat 3,5/5). Prav tako je večina (70 %) potrebovala dodatno pomoč med igranjem igre. Pri tem vidimo, da naši igri manjkajo stopnje težavnosti, saj si želimo, da igralec sam premaga igro. Sklepamo lahko, da razlog za težavnost igre ni bila uporaba kontrolerja, saj so to igralci ocenili med zmerno in enostavno (rezultat 3,6/5). Za lažjo stopnjo bi lahko dodali več navodil, več življenjskih točk in manjše število okostnjakov.

Ob opazovanju smo ugotovili, da so povprečno igro igrali šestindvajset minut. Od tega so v povprečju porabili tri minute in pol, da so si nastavili očala, dobili občutek za premikanje in pobrali svojo prvo surovino. Od desetih jih je sedem dokončalo igro, ti so v povprečju potrebovali trideset minut za zmago. Od treh, ki so odnehali, sta dva zaradi pomanjkanja časa, tretjemu pa je bila igra pretežka. Najboljši čas za dokončanje igre je bil šestnajst minut in je bil dosežen brez kakršne koli dodatne pomoči. Sami

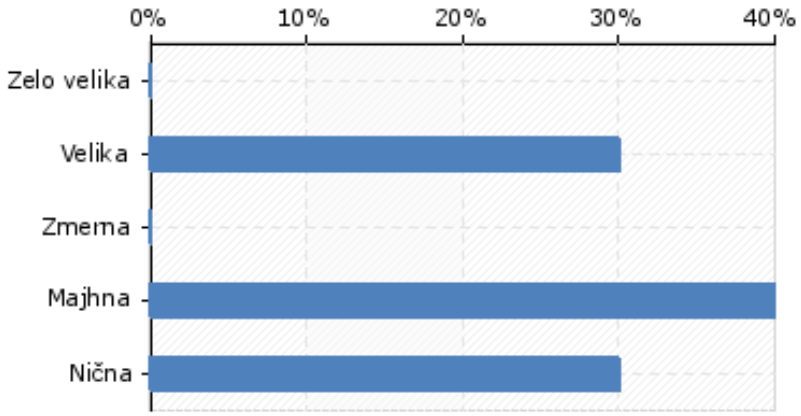


Slika 5.1: Ocena težavnosti igre

porabimo pet minut, ampak vemo vse rešitve. Slabost je izjemno pomembna tema razvoja mobilnih VR iger, zato smo igralce vprašali, kakšno stopnjo slabosti so občutili. Povprečno so slabost ocenili z malo (rezultat 3,7/5), vendar smo imeli še zmeraj tri, ki so občutili veliko slabosti (slika 5.2). To pomeni, da moramo igro testirati še na večih ljudeh in ugotoviti, kaj povzroča največ slabosti in to odpraviti. Dodatno smo opazovali udobje igralcev. Večina je raje sedela, kot pa stala in med igranjem so si samo štirje igralci vsaj enkrat popravili očala.

Igralci so na vprašanje *"Kaj se vam je zdelo v igri najboljše?"* večinoma navedli uporabo orodij. Najboljše jim je bilo nabiranje surovin, kot sta sekanje dreves in kopanje skal. Kot zabavna je bila omenjena tudi interakcija z okostnjaki, prav tako je bila pohvaljena okolica igre.

Na vprašanje *"Kaj se vam je zdelo v igri najslabše?"* smo prejeli raznolike odgovore. Prvi igralci so navedli nejasnost navodil in pomanjkanje njihove stalne prisotnosti. Zato smo jih nadgradili za ostale igralce. V splošnem so povedali, da jim nalagalni zaslon ne odgovarja in jim pri njem postaja slabo. Po premisleku smo ugotovili, da nalagalnega zaslona ne smemo premikati s



Slika 5.2: Stopnja slabosti igralcev

premikanjem glave, ampak mora ostati statičen. V času nalaganja igralcu omogočimo, da si ogleduje okolico.

Igralci so našli tudi tri pomanjkljivosti v igri. Prvič, najdeni sta bili dve točki, kjer se je igralec lahko zagozdil v teren. To nam je pomagalo, da smo v novi verziji popravili napako. Drugič, igralec je lahko prišel mimo zelenega zmaja ne da je pravilno postavil plošče. Pri reševanju te napake smo ugotovili, da pri steni ogenj ni dovolj močan, kar je igralcu omogočilo prost prehod. Težavo smo odpravili s povečanjem moči ognja in količine ognjenih delcev. Tretjič, udarec ob zamahu z mečem, ki se je igralcem sicer zdel pravilen, ni bil zaznan v igri. Za rešitev tega problema bi morali na novo opredeliti, kaj se šteje pod udarec in kaj ne. To bi nam vzelo preveč časa, zato to v zadnji verziji ni bilo odpravljeno.

Pri zadjem vprašanju smo pozvali igralce, če želijo še kaj dodati. Dobili smo kar nekaj dobrih nasvetov za dopolnitev igre. Prvi, da naredimo kakšen drug zvočni in vizualni efekt, ko kopljemo stvari, ki jih ne moremo uničiti. Drugi, da zmeraj prikazujemo UI s surovinami in naredimo bolj preglednega. Tretji, da dodamo kakšen namig, če igralec po določenem času ne napreduje.

Poglavje 6

Zaključek

Največje presenečenje je bilo število različnih nalog, ki jih moramo opraviti, da projekt začne izgledati kot igra. Najprej smo naredili koncept in načrt igre. Nato smo implementirali okolico, premikanje igralca, modele, shranjevanje, zvok, orodja, menije, uporabniški vmesnik, obnašanje in premikanje NPC ter luči. Spoznali smo tudi nekaj dobrih in slabih praks za obvladovanje slabosti uporabnikov, kar je izjemno pomembna tema pri razvoju mobilnih VR aplikacij.

V igri je še veliko prostora za izboljšave skoraj pri vsakem elementu igre. Za resno izdelovanje VR igre bi potrebovali večjo ekipo, saj je preveč različnih nalog, ki zahtevajo svoj čas, in različnih področij, ki zahtevajo specifična znanja. Za prvo pravo verzijo igre nam manjkata predvsem število dejavnosti in količina pomoči v zunanjem svetu, s čimer bi se igralec lahko še bolje spoznal z uporabo vseh orodij. Prav tako bi bilo potrebno dodelati osvetlitev ter izdelati boljši zaključek igre.

Navkljub nekaterim pomanjkljivostim se nam zdi, da je bil cilj diplomske naloge uresničen, saj smo pridobili ogromno novega znanja o pogonu Unity in platformi Daydream. Prav tako so uporabniki radi igrali igro in se pri tem zabavali z uporabo različnih orodij.

Literatura

[1] *Virtual reality.* Oxford University Press. Dosegljivo: `https://en.oxforddictionaries.com/definition/virtual_reality`. [Dostopano: 14. 8. 2018].

[2] M. Gržinič. *V vrsti za virtualni kruh. Čas, prostor, subjekt in novi mediji v letu 2000.* Znanstveno in publicistično središče Ljubljana, 1996.

[3] Daydream. Dosegljivo: `https://vr.google.com/daydream/`. [Dostopano: 14. 8. 2018].

[4] Vr headset sales comparison. Dosegljivo: `https://www.statista.com/statistics/752110/global-vr-headset-sales-by-brand/`. [Dostopano: 14. 8. 2018].

[5] Cardboard. Dosegljivo: `http://fortune.com/2017/03/01/google-cardboard-virtual-reality-shipments/`. [Dostopano: 14. 8. 2018].

[6] D. Brewster. *The Stereoscope; Its History, Theory and Construction, with Its Application to the Fine and Useful Arts and to Education, Etc.* John Murray, 1856.

[7] Daydream elements overview. Dosegljivo: `https://developers.google.com/vr/elements/overview`. [Dostopano: 14. 8. 2018].

[8] Unity. Dosegljivo: `https://unity3d.com/`. [Dostopano: 14. 8. 2018].

[9] Git. Dosegljivo: `https://git-scm.com/`. [Dostopano: 14. 8. 2018].

[10] Trello. Dosegljivo: https://trello.com/. [Dostopano: 14. 8. 2018].

[11] Visual studio. Dosegljivo: https://visualstudio.microsoft.com/. [Dostopano: 14. 8. 2018].

[12] Fantasy monster skeleton. Dosegljivo: https://assetstore.unity.com/packages/3d/characters/humanoids/fantasy-monster-skeleton-35635. [Dostopano: 14. 8. 2018].

[13] P. Peer, B. Klemenc, M. Jan. *Tehnologija iger*. Fakulteta za računalništvo in informatiko, 2011.

[14] Freesound. Dosegljivo: https://freesound.org. [Dostopano: 14. 8. 2018].

Priloga A: Anketa

1. Spol: A. Moški B. Ženski
2. Koliko ste stari? _____
3. Ste že kdaj uporabljali VR očala? A. Da B. Ne
4. Če ste, kolikokrat? _____
5. Ste kdaj izkusili slabost pri uporabi VR očal? A. Da B. Ne
6. Ste izkusili slabost pri igranju te igre? Ocenite stopnjo slabosti.

   A. Zelo velika B. Velika C. Zmerna D. Majhna E. Nična
7. Kakšno se vam je zdelo premikanje s kontrolerjem?

   A. Zelo zahtevno B. Zahtevno C. Zmerno D. Enostavno E. Zelo enostavno
8. Kakšna se vam je zdela uporaba orodij v igri?

   A. Zelo zabavna B. Zabavna C. Nič posebnega D. Dolgočasna E. Zelo dolgočasna
9. So bila navodila igre dovolj jasna?

   A. Zelo dobra B. Dobra C. V redu D. Slaba E. Zelo slaba
10. Kolikokrat na mesec povprečno igrate računalniške igre? _____

11. Kako bi ocenili težavnost igre?

    A. Preveč zahtevna B. Zahtevna C. Ravno prav D. Enostavna E. Preveč enostavna

12. Ste lahko (sami) dokončali igro?

    A. Da B. Ne

13. Kaj se vam je zdelo v igri najboljše?

    _____

14. Kaj se vam je zdelo v igri najslabše?

    _____

15. Bi še kaj dodali? (kakršenkoli komentar bo v veliko pomoč)

    _____

Priloga B: Dodatna vprašanja

1. Koliko časa je potreboval za dokončanje igre? _____
2. Koliko časa je igral, preden je rekel dovolj (in ni prišel do konca)? _____
3. Je moral sedeti ob igranju? A. Da B. Ne
4. Je stal na začetku, a se je moral potem vsesti? A. Da B. Ne
5. Koliko časa je trajala kalibracija (da se je navadil na očala in kontroler)? (Pobrana prva surovina.) _____
6. Si je očala med igranjem moral kaj popravljati? A. Da B. Ne