# Šolski center Celje Gimnazija Lava

IZDELAVA IGER  
PREŽIVETJE

Maturitetna seminarska naloga pri računalništvu

Mentorica Avtor

Celje, april 2016

**Povzetek**

Tema seminarske naloge je izdelava igre. To temo sem si izbral, ker me oblikovanje in ustvarjanje videoiger zelo zanima. V razvojnem okolju Unity sem ustvaril preprosto 2D arkadno igro, ki jo je mogoče igrati na operacijskem sistemu Windows. Vso kodo sem napisal v jeziku C# s programom Notepad++, vse potrebne sličice pa sem narisal s programom Aseprite. Med ustvarjanjem igre sem se naučil veliko osnov izdelave iger in risanja, prav tako pa sem ugotovil da je izdelava igre zelo kompleksen in zahteven proces. S končno igro sem zadovoljen, a je zmeraj prostor za izboljšavo.

**Summary**

The theme of the seminar work is making a videogame. I chose this topic because I'm very interested in videogame design and creation. I created a simple 2D arcade game using the Unity engine that can be played on the Windows operating system. I've written all of the code in C # with Notepad ++, and I made all of the sprites with a drawing program called Aseprite. While creating the game, I learned a lot of basics of making games and drawing, though I also realized, that making games is a very complex and difficult process. Im satisfied with the finished game, but there's always room for improvement.

**Ključne besede**

Unity, videoigre, razvijanje, programiranje, risanje.

Kazalo vsebine

[1 UVOD 5](#_Toc448690298)

[2 Načrt 5](#_Toc448690299)

[3 Razvojno okolje Unity 5](#_Toc448690300)

[4 Izdelava likov 6](#_Toc448690301)

[4.1 Risanje 6](#_Toc448690302)

[5 Animacije 7](#_Toc448690303)

[6 Kontroliranje igralca in animacije 8](#_Toc448690304)

[6.1 Premikanje igralca 8](#_Toc448690305)

[6.2 Premikanje sovražnika 9](#_Toc448690306)

[6.3 Napad, smrt 9](#_Toc448690307)

[5.4 Omejitev igralnega prostora 10](#_Toc448690308)

[7 Programiranje igre 11](#_Toc448690309)

[7.1 Pojavitev sovražnikov 11](#_Toc448690310)

[7.2 Valovi igre 12](#_Toc448690311)

[7.3 Konec igre 12](#_Toc448690312)

[7.4 Sledenje rezultatu 12](#_Toc448690313)

[7.5 Tekstovni objekti 12](#_Toc448690314)

[8 Izdelava šefov 13](#_Toc448690315)

[8.1 Prvi šef 13](#_Toc448690316)

[8.2 Drugi šef 14](#_Toc448690317)

[8.3 Tretji šef 14](#_Toc448690318)

[8.4 Četrti šef 15](#_Toc448690319)

[8.5 Pojavljanje šefov 15](#_Toc448690320)

[8.6 Popravljanje trkalnikov 15](#_Toc448690321)

[9 Prikaz dobljenih točk 16](#_Toc448690322)

[9.1 Prikaz točk ob smrti sovražnika 16](#_Toc448690323)

[9.2 Prikaz življenjskih točk šefov 16](#_Toc448690324)

[10 Zvok 17](#_Toc448690325)

[10.1 Zvočni efekti 17](#_Toc448690326)

[10.2 Glasba 17](#_Toc448690327)

[11 Izdelava glavnega menija 17](#_Toc448690328)

[12 Testiranje 18](#_Toc448690329)

[13 Navodila za uporabo 18](#_Toc448690330)

[14 Zaključek 19](#_Toc448690331)

[15 Viri in literatura 20](#_Toc448690332)

[16 Priloge 20](#_Toc448690333)

**Kazalo slik**

[Slika 1: Izgled razvojnega okolja 6](#_Toc448689029)

[Slika 2: Sličice animacije napada 7](#_Toc448689030)

[Slika 3: Delo na sličici sovražnika 7](#_Toc448689031)

[Slika 4: Rezanje sličic 8](#_Toc448689032)

[Slika 5: Metoda za obračanje lika 9](#_Toc448689033)

[Slika 6: Trkalnik meča 10](file:///C:\Users\Antllej\Desktop\maturitetna\seminarska.docx#_Toc448689034)

[Slika 7: Korutina Sword() 10](#_Toc448689035)

[Slika 8: Meje prostora 11](#_Toc448689036)

[Slika 9: Primer pojavitve sovražnika 11](#_Toc448689037)

[Slika 10: Ekran ob izgubi igre 13](#_Toc448689038)

[Slika 11: Prvi šef 14](#_Toc448689039)

[Slika 12: Koda ki omogoča in onemogoča premikanje 14](#_Toc448689040)

[Slika 13: Celoten pogoj za konec vala 15](#_Toc448689041)

[Slika 14: Nove sličice šefov 15](#_Toc448689042)

[Slika 15: Prikaz dobljenih točk 16](#_Toc448689043)

[Slika 16: Izgled življenjskih točk 16](#_Toc448689044)

[Slika 17: Koda za spremembo besedila 17](#_Toc448689045)

[Slika 18: Glavni meni igre 18](#_Toc448689046)

[Slika 19: Izbira resolucije 19](#_Toc448689047)

# Uvod

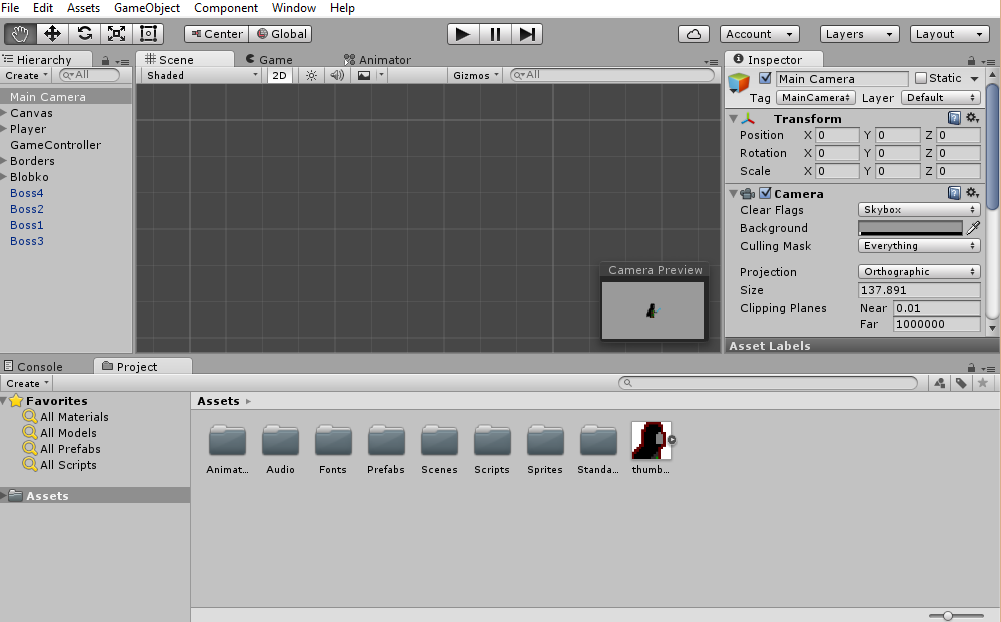
Ker je igranje videoiger eden izmed mojih najljubših hobijev, sem si za temo seminarske naloge izbral izdelavo igre. Ker z izdelavo iger še nimam izkušenj, bi predvsem rad, da se naučim osnove ustvarjanja igre, ki jih lahko kasneje uporabim za večje, bolj ambiciozne projekte.   
 Želim ustvariti arkadno dvodimenzionalno igro, ki se dogaja le na enem ekranu, in katere cilj je zbrati čim več točk z ubijanjem sovražnikov. Zgledoval se nisem po nobeni specifični igri. Igre, ki temeljijo le na zbiranju čim večjega števila točk so dandanes zelo pogoste, predvsem na mobilnih napravah. Zaradi predlogov sošolcev bom igro ustvaril z razvojnim okoljem Unity. Uporablja jezik C#, ki je podoben jeziku Java, s katerim imam že precej izkušenj. Vso kodo bom napisal v programu Notepad++. Vse potrebne grafične like bom ustvaril sam v brezplačnem programu Aseprite, namenjenemu risanju s točkami (t.i. pixel art). Igro bo mogoče igrati na operacijskem sistemu Windows.

# Načrt

Preden sem se lotil izdelave igre sem dobro premislil, kaj bi hotel sam igrati. Odločil sem se, da bom ustvaril preprosto a zabavno arkadno igro, v kateri igralec ubija sovražnike in tako zbira točke. Moj cilj je bil predvsem, da je igralec motiviran, da izboljša svoj najvišji rezultat. Rezultate je zato treba seveda tudi shranjevati v računalnik. Odločil sem se, da bo igra potekala le v dveh scenah. Prva scena je glavni meni, druga scena pa je sama igra. Kontrole so zelo preproste. Igralec se s puščicami premika po ekranu, s tipko 'Space' pa napade. Vsak ubiti sovražnik igralcu prinese nekaj točk. Ko se igralca sovražnik dotakne je igre konec. Ker se igra dogaja le na enem ekranu, se mi z animiranjem ozadja ni bilo treba ukvarjati. Narisati sem moral le sličice za lik igralca in sličice za lik sovražnika.

# Razvojno okolje Unity

Unity je razvojno okolje, ki ga je razvil Unity Technologies. Prva verzija okolja je izšla 8. junija 2005. Unity ima poudarek na čim bolj dostopnemu razvoju za ogromno število platform. Omogoča razvoj aplikacij in iger za vse igralne konzole, osebne računalnike, mobilne telefone in spletne strani. Osebna verzija okolja Unity je na voljo vsem brez plačila, medtem ko je profesionalna verzija z več funkcijami plačljiva. Razvojno okolje postaja vedno bolj popularno. Unity pogon uporablja namreč veliko predvsem manjših popularnih iger, kot so npr. Firewatch, Oxenfree, Ori and the Blind Forest, Angry Birds 2 itd.



Slika 1: Izgled razvojnega okolja

# Izdelava likov

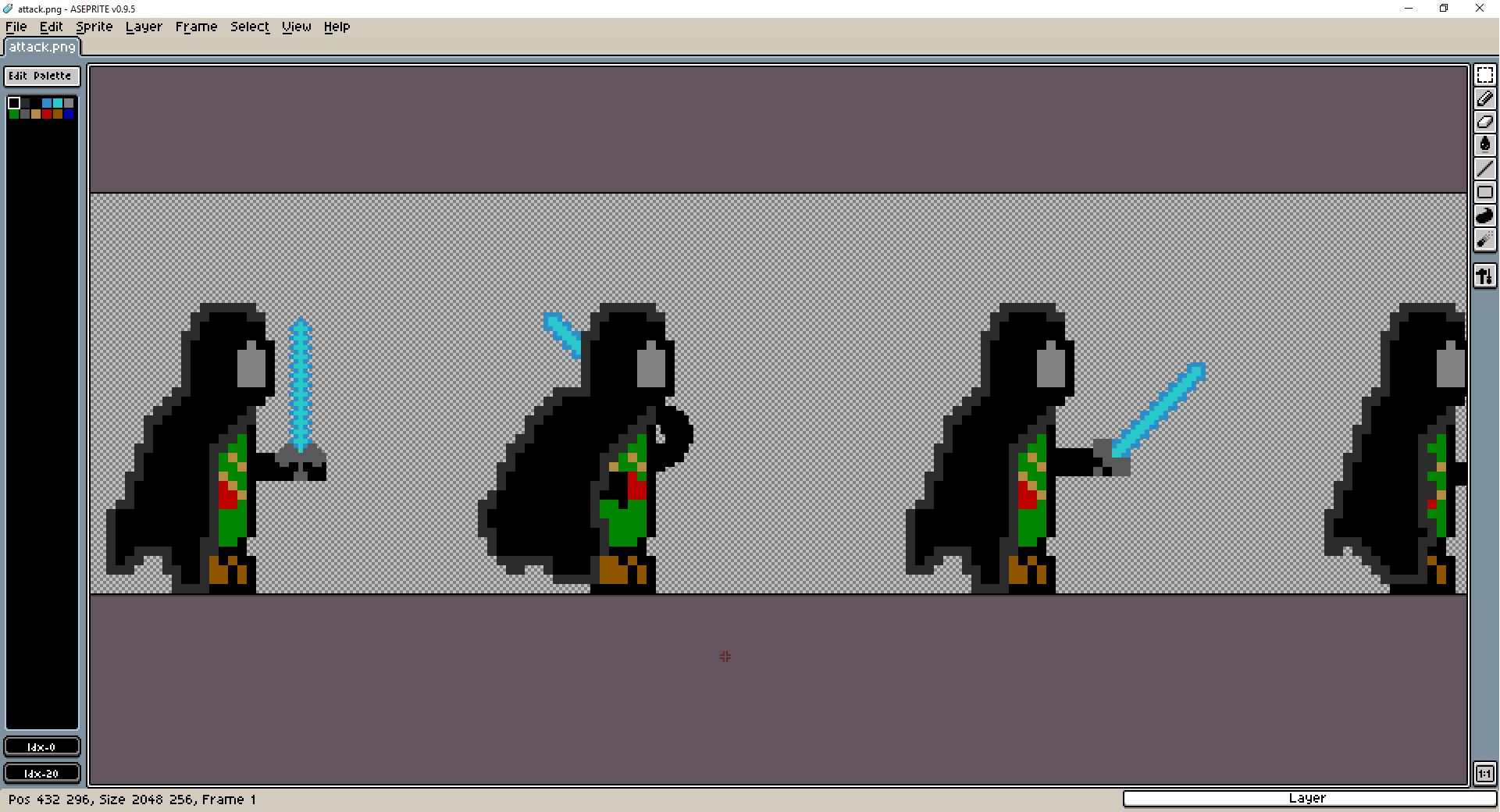
Ideja, ki sem jo dobil za lik igralca, je bojevnik z nekakšno zeleno srednjeveško tuniko, rjavimi škornji ter črnim ogrinjalom s kapuco, ki mu zakriva obraz. V roki ima moder meč, ki ga uporablja za spopad s sovražniki. Sovražnike sem si zamislil kot kepe zelene sluzi, ki skušajo ubiti bojevnika, ki ga kontrolira igralec. Za takšen izgled sovražnika me je verjetno navdihnila igra Titan Souls, ki sem jo igral nedolgo preden sem začel razmišljati o svoji igri.

## Risanje

Da sem se lahko kasneje osredotočil na samo izdelavo igre, sem najprej ustvaril vse potrebne grafične like. Risanja sem se lotil v brezplačnem programu Aseprite, ki je namenjen risanju s točkami. Program ni težaven za uporabo, a ima vseeno veliko funkcij. Moral sem ustvariti sličice za 6 različnih animacij. Za igralca sem potreboval 4 animacije:

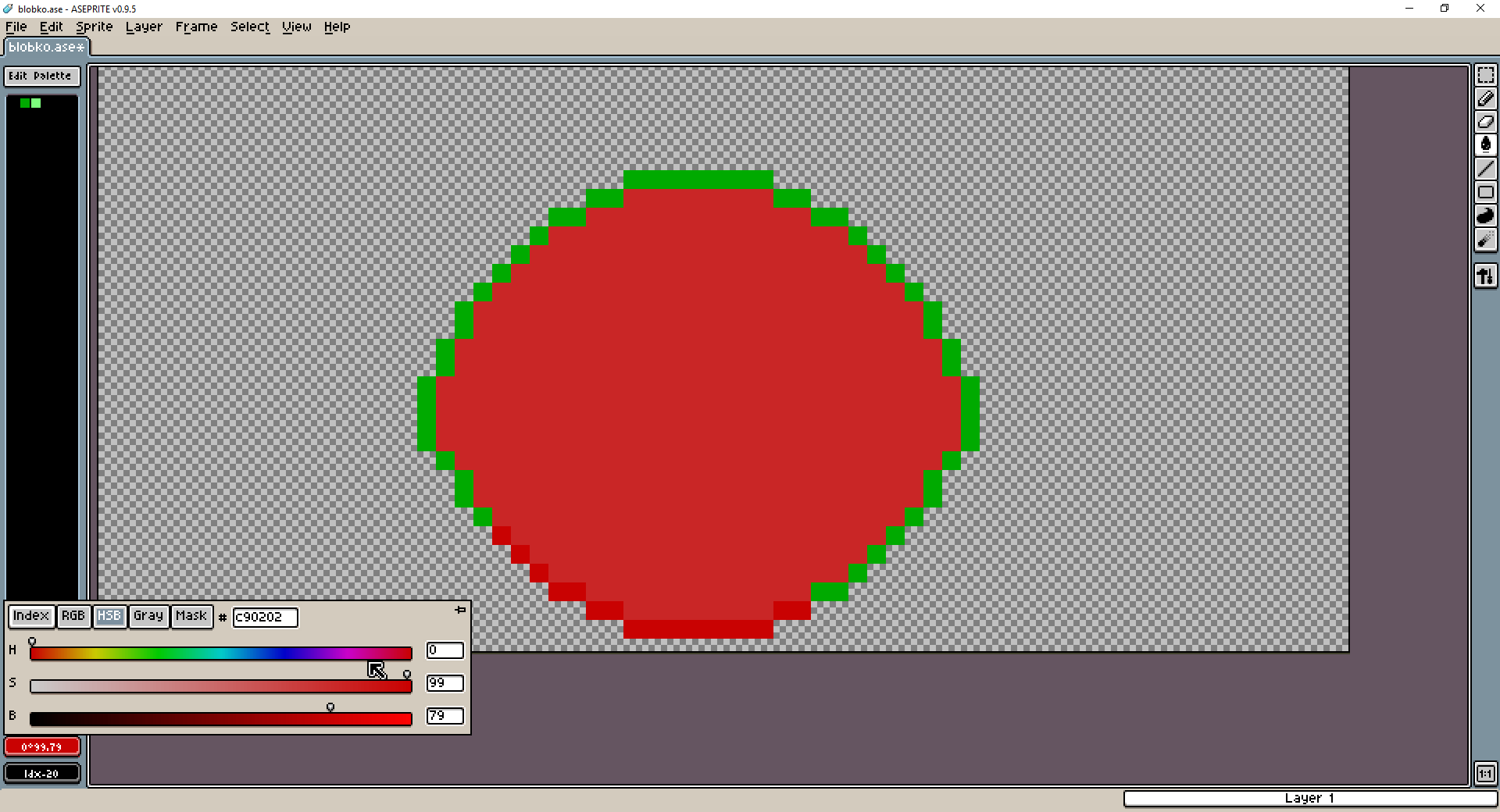
* animacijo, ki se predvaja, ko igralec ne dela ničesar
* animacijo za hojo
* animacijo za napad
* animacijo, ki se predvaja ob smrti.

Če bi hotel narisati grafične like še v primeru, ko je igralec obrnjen v levo smer, bi si delo podvojil, a lik samo obrnem v drugo smer s pomočjo kode, ko je potrebno. Posledica tega je, da ima lik po obratu meč v drugi roki.



Slika 2: Sličice animacije napada

Za lik sovražnika sem potreboval le 2 animaciji; animacijo za premikanje ter animacijo ob smrti. Risanje poševnih črt, krogov in elips zlahka zgleda čudno pri risanju s točkami, zato sem si pri animiranju sovražnika pomagal s programom, ki generira točkasto elipso glede na vneseno višino in širino.



Slika 3: Delo na sličici sovražnika

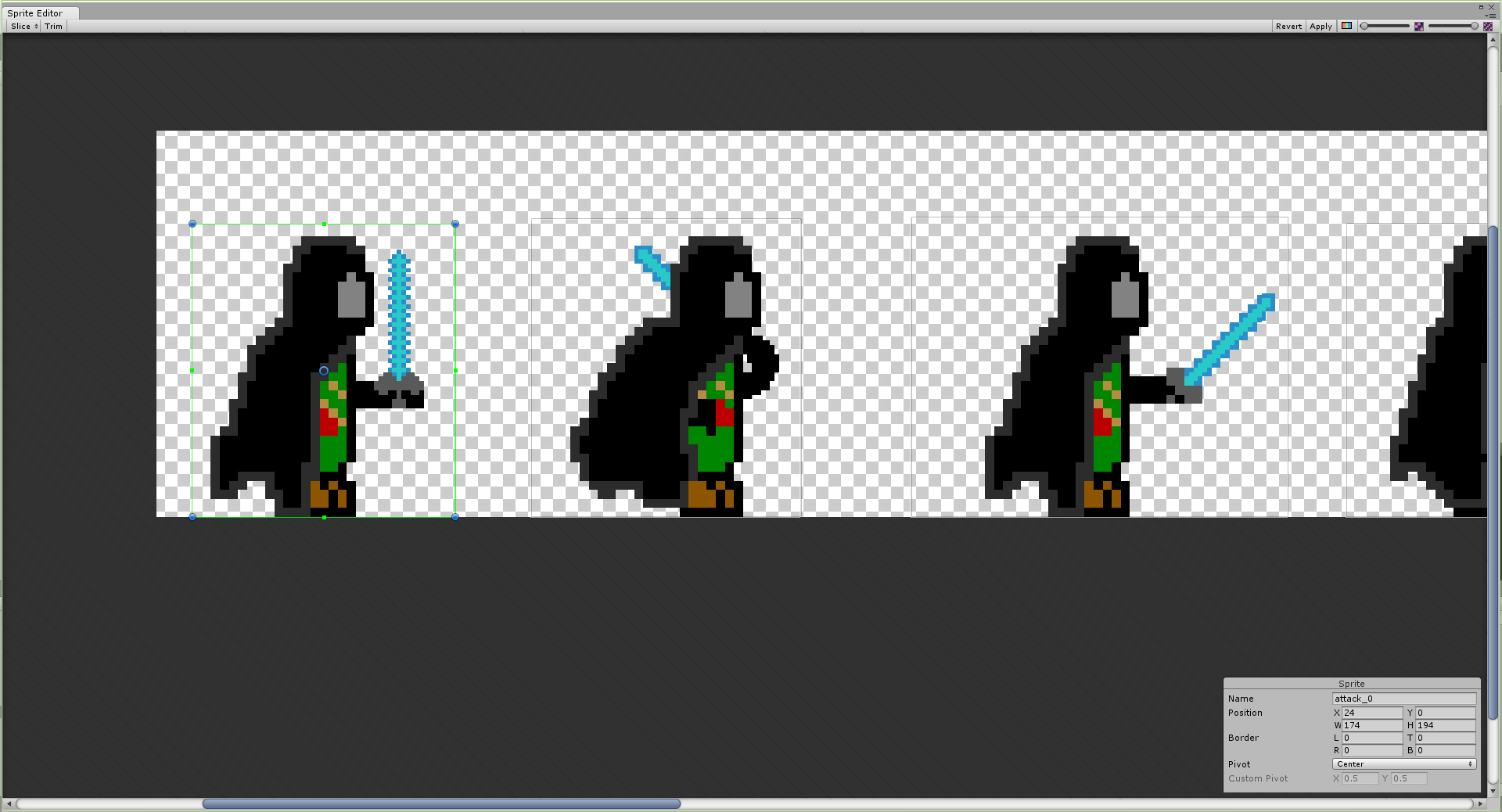
Ko sem končal z risanjem, sem sličice vsake animacije izvozil kot list sličic tipa .png in jih shranil v mapo Sprites.

# Animacije

Ustvaril sem nov Unity projekt in ga poimenoval Game. Prvo sceno, v kateri se bo dogajala dejanska igra sem poimenoval game. Sceno si lahko predstavljamo kot posamezen nivo v igri. Ustvaril sem dva nova prazna objekta;

* Player, objekt igralca
* Blobko, objekt sovražnika.

Preden sem se lotil dela na sami igri sem ustvaril vse potrebne animacije. V Sprite Editorju, ki je namenjen obdelavi sličic, sem odprl list sličic vsake animacije posebej. V programu sem nato iz njih izrezal vse sličice v pravilnem vrstnem redu. Pri tem sem moral paziti, da je bila točka rotacije v vseh sličicah na istem mestu, saj vse sličice niso iste velikosti, in sem jo moral nastaviti ročno. To je zelo pomembno za pravilno menjavanje sličic med animacijo.



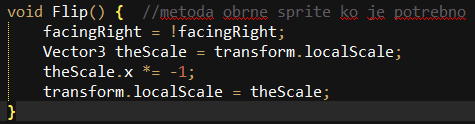
Slika 4: Rezanje sličic

Ko sem izrezal vse potrebne sličice, sem jih označil ter povlekel k pripadajočemu objektu, s čimer sem ustvaril animacije, ki sem jih shranjeval v mapo Animations. S kreacijo animacije na objektu se ustvari tudi Animation Controller, ki je potreben za menjanje animacij.

# Kontroliranje igralca in animacije

## Premikanje igralca

Animacijo playerStill, ki se predvaja, ko igralec stoji pri miru, sem v Animatorju nastavil kot privzeto animacijo, kar pomeni da se bo predvajala zmeraj, ko igralec ničesar ne počne. Ustvaril sem novo skripto za objekt Player in jo poimenoval PlayerController. S to skripto bom kontroliral obnašanje objekta. Najprej sem se lotil kodiranja premikanja igralca. V metodi Update(), ki se izvede vsako sličico igre, sem napisal 4 pogoje, za premikanje v vsako smer. V teh pogojih preverjam ali je bila pritisnjena določena tipka. Igralec se lahko premika s puščicami ali s tipkami 'A','W','S' in 'D'. Ob pritisku na tipko premikam objekt igralca v sorazmerni smeri. Ker se more lik igralca obrniti, ko se začne premikati v drugo smer, sem ustvaril novo spremenljivko tipa boolean facingRight, katere privzeta vrednost je true. V pogojih za premikanje v levo in desno smer preverim ali facingRight drži. Če je pritisnjen gumb za premikanje v levo ko je facingRight enak true, ali obratno, se kliče metoda Flip(). Metoda spremeni vrednost spremenljivke facingRight in zasuče lik igralca po x osi. To storim tako da x os objekta pomnožim z -1.



Slika 5: Metoda za obračanje lika

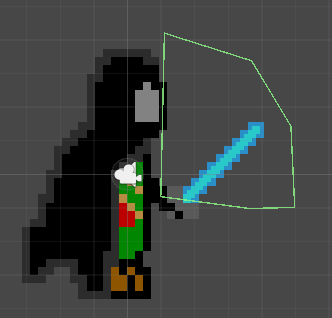
Zdaj ko se igralec lahko premika, sem moral poskrbeti še za predvajanje animacije hoje. V Animatorju objekta Player sem ustvaril novo spremenljivko tipa boolean Walk, in dodal preprost pogoj, da se animacija PlayerWalk predvaja ko je Walk true. Vsakič, ko je pritisnjen gumb za premikanje, se spremenljivka Walk spremeni v true, in animacija hoje se predvaja. Kmalu mi je postalo jasno, da se igralec premika po diagonali hitreje, kot normalno. To se zgodi, ker se igralec premika v obe smeri z isto hitrostjo, in tako naredi večjo pot. To hitrost si lahko predstavljamo kot hipotenuzo pravokotnega trikotnika, kjer kateti predstavljata normalno hitrost. Zaradi tega sem moral uvesti dve novi spremenljivki, s katerimi sem preverjal ali je pritisnjenih več tipk hkrati. V primeru da so, zmanjšam hitrost premikanja v obe smeri.

## Premikanje sovražnika

Upravljanje in delo z animacijami je mogoče v Animatorju objekta. Objektu Blobko sem v Animatorju določil animacijo BlobkoStill za privzeto animacijo. Za obnašanje sovražnika sem ustvaril skripto BlobkoController in jo dodal objektu. Za premikanje sovražnik uporablja le privzeto animacijo. Sprva sem imel v mislih, da bi se sovražniki premikali k igralcu s skoki, ki imajo naključno dolžino in višino, a sem ugotovil da je takšno premikanje zelo kompleksno, zato sovražnike le premikam k igralcu. Koda za premikanje je zelo preprosta. V metodi Update() primerjam poziciji igralca in sovražnika na x osi ter y osi, nato pa premikam sovražnika k igralcu.

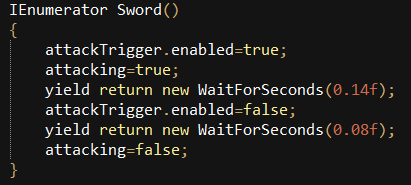
## Napad, smrt

Sedaj ko se igralec lahko premika, sem igri dodal napad. V Animatorju objekta Player sem ustvaril novo spremenljivko tipa boolean Attack. Animacija napada se predvaja vsakič, ko je Attack enak true. Animaciji sem dodelil fiksno dolžino, kar pomeni da se animacija zmeraj izvede do konca, tudi če je med izvajanjem pritisnjena kakšna druga tipka. Sedaj sem v skripti igralca ustvaril nov pogoj, da se ob pritisku tipke 'Space' spremenljivka Attack v Animatorju spremeni v true. Prav tako sem dopolnil pogoje ki kontrolirajo premikanje tako, da se igralec med animacijo napada ne more premikati. Sedaj ko igralec lahko napada, sem objektoma Player in Blobko dodal poligonski trkalnik. Trkalnik je komponenta objekta, ki zaznava trke drugih objektov. Objektu sovražnika sem določil oznako 'Enemy', kar bo pomembno kasneje, ko bom moral ugotavljati, česa se je igralec oz. meč dotaknil. Ker potrebuje meč svoj trkalnik, sem objektu Player ustvaril nov podrejen objekt AttackTrigger. Nato sem temu objektu dodal svoj trkalnik.



Slika 6: Trkalnik meča

Trkalnik meča mora biti aktiviran le takrat, ko igralec napada, zato sem objektu Player dodal novo skripto PlayerAttack, ki določa kdaj bo trkalnik meča aktiven. V metodi Update() preverjam, kdaj je pritisnjena tipka Space. Ob pritisku tipke kličem korutino Sword(). Korutine so posebne vrste metod, ki mi omogočajo, da pavziram izvajanje kode za določeno število sekund, kar je velikokrat uporabno. V korutini Sword() aktiviram trkalnik meča za 0.14 sekunde, nato pa ga deaktiviram za 0.08 sekunde.

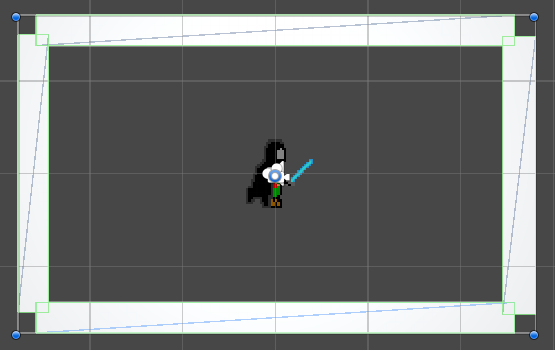


Slika 7: Korutina Sword()

Trkalnik bi lahko aktiviral za celotno trajanje animacije napada (0.22 sekunde), a takrat pogosto pride do čudnih smrti sovražnikov, če se dotaknejo vrha trkalnika meča, kljub temu da meča ni blizu. V skripti sovražnika sem ustvaril metodo Attacked(), ki se kliče ob njegovi smrti. Metoda predvaja animacijo smrti, potem pa objekt uniči. Sedaj ko deluje trkalnik meča sem ustvaril skripto za objekt AttackTrigger z istim imenom, v kateri preverjam kdaj pride trkalnik meča v stik z drugim trkalnikom. Za to uporabim metodo OnTriggerEnter2D(). Ko se metoda izvede, kliče v skripti objekta, s katerim je objekt AttackTrigger prišel v stik, metodo Attacked(). Treba je seveda omogočiti še smrt igralca. V skripti objekta Player sem uporabil metodo OnTriggerStay2D(), s katero preverjam kdaj se igralca dotakne objekt z oznako 'Enemy'. V tem primeru se predvaja animacija smrti, premikanje in napadanje igralca pa se onemogočita. Metoda OnTriggerStay2D() se od metode OnTriggerEnter2D() razlikuje v tem, da se izvede skoraj vsako sličico, ko sta trkalnika v stiku, kar bo potrebno pri zaznavanju meja arene.

## Omejitev igralnega prostora

Ker lahko igralec zapusti igralni prostor, sem ga moral omejiti. Ustvaril sem nov 3D objekt, kocko, ter ustvaril 3 kopije. Objekte sem namestil okoli igralnega prostora.



Slika 8: Meje prostora

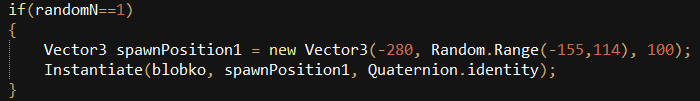
Vsem štirim objektom sem določil svojo oznako: BorderUp, BorderDown, BorderLeft in BorderRight. Metodo OnCollistionStay2D() v skripti igralca sem sedaj dopolnil s štirimi novimi pogoji. Ko se igralec zaleti v objekt z oznako BorderUp, ga premikam navzdol, kar negira igralčevo premikanje, in je pri miru. Enako sem storil še za ostale tri meje. Sedaj sem končal s temelji igranja, na katerih lahko začnem graditi igro.

# Programiranje igre

Ustvaril sem nov prazen objekt GameController in mu dodal skripto z istim imenom. S tem objektom bom kontroliral večino dogajanja v igri.

## Pojavitev sovražnikov

V skripti GameController sem ustvaril novo spremenljivko tipa int waveCount, s katero štejem valove sovražnikov. Ustvaril sem novo metodo SpawnWaves(), s katero bom ustvarjal sovražnike. V prvem valu se pojavi le eden sovražnik, v vsakem naslednjem pa eden več, dokler igralec ne doseže 8. vala. Od takrat naprej je število sovražnikov vedno enako. Zaradi tega sem v metodi ustvaril zanko, ki se izvede osemkrat (števec poteka od 0 do 7). Ustvaril sem pogoj, da se zanka prekine, če števec v zanki doseže vrednost waveCount, kar se zgodi v prvih sedmih valovih. V zanki vrednosti randomN določim naključno vrednost med 1 in 4, ker se lahko sovražnik pojavi na štirih različnih mestih (na vsaki meji arene). Prav tako se lahko pojavi kjerkoli na izbrani meji. Sovražniki se ustvarijo z metodo Instantiate(), ki naredi kopijo objekta na dani poziciji.



Slika 9: Primer pojavitve sovražnika

Ker sem hotel da se sovražniki po hitrosti in velikosti razlikujejo, sem jim ti dve spremenljivki določita naključno. V metodi Start() skripte sovražnika, ki se kliče, ko je objekt ustvarjen, objektu določim naključno hitrost med 20 in 60 ter naključno vrednost med 0.8 in 1.2, s katero nato pomnožim velikost lika.

## Valovi igre

Igra mora seveda vedeti, kdaj je igralec ubil vse sovražnike. Številu ubitih sovražnikov sledim s spremenljivko enemiesKilled. Ustvaril sem metodo IncreaseScore(), ki se kliče ob smrti sovražnika, v kateri povečam spremenljivko enemiesKilled za 1. Metodo Attacked() v skripti sovražnika sem dopolnil s klicom te metode. Sedaj v metodi Update() skripte GameController preverjam ali je spremenljivka enemiesKilled dosegla število vala ali pa vrednost 8, saj se po osmem valu vedno pojavi 8 sovražnikov. Če je pogoju zadoščeno ponovno kličem metodo, ki ustvari sovražnike, waveCount pa povečam za 1.

## Konec igre

Da se igra lahko konča, sem v skripti GameController ustvaril novo spremenljivko tipa boolean gameOver s privzeto vrednostjo false. Vrednost spremenljivke se v novi metodi GameOver() spremeni v true. Metodo GameOver() kličem v skripti igralca, ko ugotovim, da se ga je dotaknil sovražnik. Ob koncu igre sem igralcu moral še omogočiti, da ponovno naloži igro. Ustvaril sem novo spremenljivko tipa boolean restart s privzeto vrednostjo false . Ko je igre konec, se restart spremeni na true. Ko je restart enak true, lahko igralec s pritiskom na tipko 'R' ponovno naloži igro. Za to sem uporabil metodo LoadLevel().

## Sledenje rezultatu

Ker je cilj igre zbrati čim več točk, sem ustvaril dve novi spremenljivki tipa int; score in highScore. S prvo bom sledil točkam trenutne igre, z drugo pa bom sledil največjemu doseženemu številu točk. Metodi IncreaseScore(), ki se izvede ob smrti sovražnika, sem dodal parameter tipa int points. To vrednost prištejem spremenljivki points ko se metoda izvede. Število dobljenih točk je odvisno od hitrosti sovražnika. Najpočasnejši ubiti sovražnik je vreden 20 točk, najhitrejši pa 60 točk. Zdaj moram ob koncu igre še preveriti, ali je trenutni rezultat večji od najvišjega rezultata. Če je, spremenljivki highScore priredim vrednost score, in jo shranim z metodo PlayerPrefs.SetInt(). Razred PlayerPrefs omogoča shranjevanje podatkov, ki se ohranijo tudi po izhodu iz igre.

## Tekstovni objekti

Igralec mora med igro seveda slediti številu dobljenih točk, ob koncu pa je igra potrebovala še t.i. »game over« ekran, ki da igralcu vedeti, da je izgubil. Za delo z besedilnimi objekti je v okolju Unity potreben okvir, ki sem mu določil velikost kamere igre. Ustvaril sem 5 novih besedilnih objektov. Objekt ScoreText uporabljam za prikazovanje točk, ostali objekti pa se pokažejo le ob koncu igre. Točke se prikazane zgoraj, na sredini ekrana. Tekstu ScoreText določim novo vrednost vsakič, ko se kliče metoda IncreaseScore() oz. vsakič, ko umre sovražnik. Objekt LoseText je postavljen na sredini ekrana, in pove igralcu da je izgubil igro. Objekt HighScoreText sem postavil v desni zgornji kot. Ta igralcu ob koncu igre pokaže najvišji rezultat. Objekt RestartText je postavljen v levi zgornji kot. Ta tekst pove igralcu, kako lahko ponovno naloži igro. Objekt QuitText je postavljen v spodnji levi kot. Igralcu pove kako oditi iz igre. To možnost sem dodal kasneje, ko sem ustvaril glavni meni. Vsem tekstom sem določil rdečo barvo in točkasto pisavo, kar bolje ustreza izgledu in občutku igre.



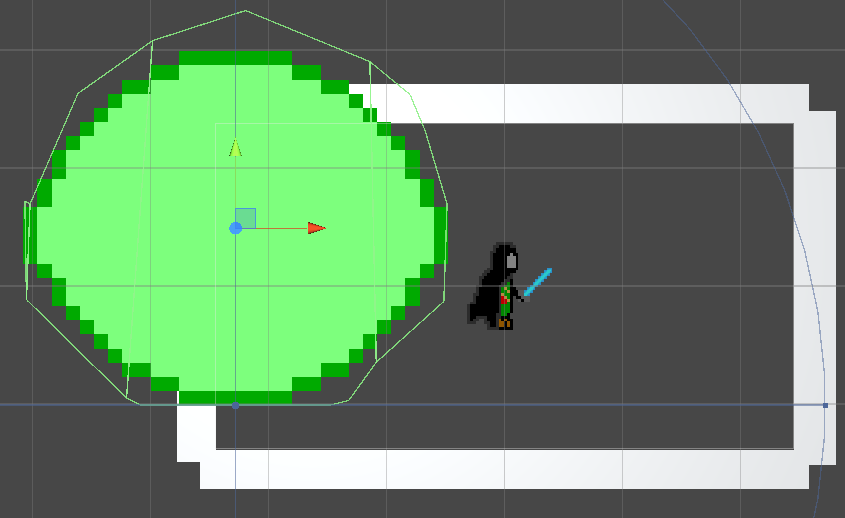
Slika 10: Ekran ob izgubi igre

# Izdelava šefov

Med ustvarjanjem igre sem dobil idejo, da bi dodal še nekaj šefov (angl. boss), ki bi igro naredili malo bolj zanimivo. Šefe je namreč težje ubiti kot normalne sovražnike, vredni pa so tudi več točk. Ustvaril sem štiri različne šefe. Njihovi objekti so v bistvu kopije objekta Blobko (objekt sovražnika), saj uporabljajo iste animacije.

## Prvi šef

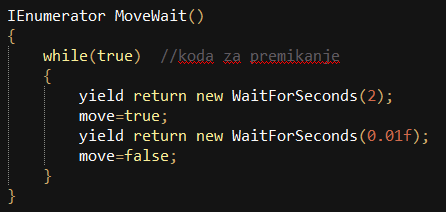
Objekt Boss1 sem povečal na približno sedemkratno velikost normalnega sovražnika. Za obnašanje šefa sem ustvaril novo skripto Boss1Controller. Kodo premikanja proti igralcu sem pri šefu ohranil. Edina sprememba je pri pogoju za premikanje gor in dol. Koordinata šefa je namreč na čistem dnu sličice, kar se pri normalnih sovražnikih ni opazno, pri šefu pa, ker je tako velik. To sem rešil tako, da sem pri primerjavi y koordinat koordinati igralca odštel 150, kar je isto, kot da bi koordinata šefa bila v njegovem centru. Ker imajo šefi več življenjskih točk kot normalni sovražniki, sem v ustvaril novo spremenljivko tipa int z imenom life, s katero sledim življenjskim točkam šefa. Prvega šefa je potrebno udariti petnajstkrat, da umre. Tudi skripte šefov vsebujejo metodo Attacked(), a malce spremenjeno. Ob klicu metode se vrednost spremenljivke life zmanjša za 1. Ko ta vrednost doseže 0, jih uničim kot normalne sovražnike. Prav tako kličem metodo IncreaseScore(). Vsi šefi imajo fiksno vrednost točk, ki jih igralec pridobi ob njihovi smrti. Prvi je vreden 700 točk, ker je najlažji.



Slika 11: Prvi šef

## Drugi šef

Drugi šef je malo manjši od prvega, gibanje pa je povsem drugačno. Šef je pri miru, vsaki dve sekundi pa se za stotinko sekunde zelo hitro premika proti igralcu. Zgleda, kot da se teleportira. Pri tem sem si spet pomagal s korutino, ker mi omogoča da za določen čas onemogočim izvajanje kode.



Slika 12: Koda ki omogoča in onemogoča premikanje

Igralec bo med prvim soočenjem s tem šefom verjetno umrl, ker še ne pozna njegovega načina gibanja. S tem se bo naučil, da mora biti pazljiv. Šef ima 10 življenjskih točk, vreden pa je 1000 točk, saj je težji kot prvi šef.

## Tretji šef

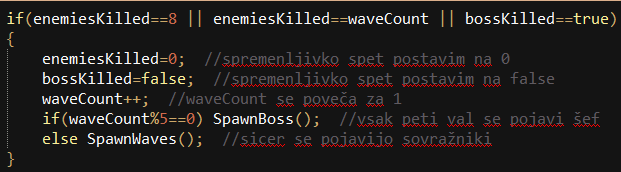
Tretji šef je enako velik kot drugi. Tudi premika se zelo podobno, samo da so presledki med premikanjem manjši, prav tako pa je manjša tudi hitrost premikanja. Ima 12 življenjskih točk, a je kljub temu lažji kot drugi šef, zato je vreden le 750 točk.

## Četrti šef

Zadnji šef je enako velik kot sta drugi in tretji, a je nedvomno najtežji. Hkrati je tudi edini, ki ima fiksno pot premikanja, neodvisno od igralca. Rotira namreč okoli arene, in se približuje njenemu centru, prav tako se nekajkrat hitro premakne v svojo desno smer, zato je zelo nepredvidljiv. Ima 10 življenjskih točk, ob smrti pa podari kar 1300 točk.

## Pojavljanje šefov

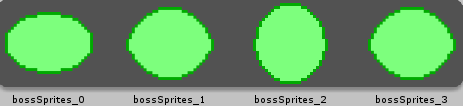
V skripti GameController sem ustvaril novo metodo SpawnBoss(). V metodi ustvarim objekt šefa na isti način kot ustvarim sovražnike, samo da se šefi lahko pojavijo le na desni ali levi strani arene. Z naključno vrednostjo med 1 in 4 določim, kateri šef se bo pojavil. Ker moram vedeti, kdaj je bil šef ubit, da se lahko začne naslednji val, sem ustvaril novo spremenljivko tipa boolean bossKilled, ki se ob smrti šefa spremeni v true. Nato sem pogoju, v katerem preverjam ali so bili ubiti vsi sovražniki, dodal še pogoj, ki preverja, ali je bil ubit šef. Šef se pojavi samo vsako peto rundo. To storim tako, da metodo SpawnBoss() kličem le takrat, ko je število vala deljivo s 5.



Slika 13: Celoten pogoj za konec vala

## Popravljanje trkalnikov

Med igranjem sem pri šefih opazil majhen problem. Trkalnik šefov je namreč fiksen, a ker so objekti veliko večji kot sovražniki, je razlika med najbolj stisnjeno in najbolj raztegnjeno sličico šefa zelo opazna. Zaradi tega lahko včasih pride do smrti, kljub temu da se igralec sličice šefa dejansko ne dotika, le njegovega trkalnika. To sem izboljšal tako, da sem trkalnike naredil malo bolj natančno, animacijam vseh šefov pa sem odstranil 4 sličice, da je razlika med najbolj stisnjeno in najbolj raztegnjeno sličico čim manjša.



Slika 14: Nove sličice šefov

# Prikaz dobljenih točk

## Prikaz točk ob smrti sovražnika

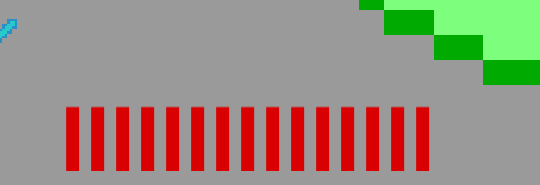
Ker sem hotel, da lahko igralec vidi, koliko točk mu je podaril ubiti sovražnik, sem objektu Blobko dodal nov podrejen besedilni objekt imenovan PointsText. Postavil sem ga nad sovražnika, besedilu pa sem določil isto barvo in pisavo, kot jo imajo vsa besedila v igri. Ob smrti sovražnika besedilu, ki mu pripada, spremenim vrednost, da prikaže hitrost objekta oz. dobljene točke. Tekst se uniči hkrati z objektom sovražnika, zato je viden le kratek čas.



Slika 15: Prikaz dobljenih točk

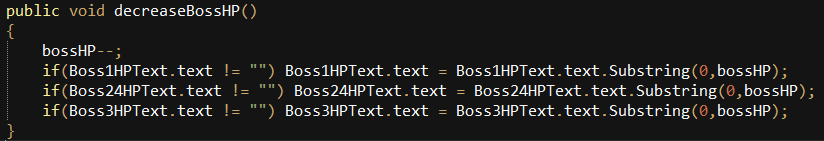
## Prikaz življenjskih točk šefov

Od prijatelja, ki je testiral igro, sem dobil predlog, da bi bile vidne življenjske točke šefov. Na to sem pomislil tudi sam, saj igralec ne ve, kdaj bo šef uničen. To lahko privede do nadležnih smrti, če ravno takrat igralec stoji na robu arene, kjer se pojavijo sovražniki. Glavni navdih za izgled prikaza življenjskih točk sem dobil od NES iger kot so Ninja Gaiden ali Castlevania. Ustvaril sem tri nove besedilne objekte, na katerih bodo prikazane življenjske točke šefov. Tri različna besedila potrebujem, ker imajo šefi različno število točk: 10, 12 in 15. Vseh ne morem prikazati na enem objektu, ker morajo imeti točke levo poravnavo, pozicionirane pa so na sredini ekrana. Za prikaz črt, ki predstavljajo življenjske točke, sem uporabil kar črko I.



Slika 16: Izgled življenjskih točk

Vrednost tekstov za prikaz življenjskih točk šefov spreminjam v skripti GameController. Ob pojavitvi šefa se spremeni vrednost pripadajočega teksta v določeno število I-jev, ki predstavljajo njegove življenjske točke. Hkrati pa novi spremenljivki tipa int bossHP dodelim število življenjskih točk šefa, ki se je pojavil. V metodi Attacked(), ki se izvede, ko igralec šefa udari, se vsakič kliče metoda decreaseBossHP() v skripti GameController. V tej metodi zmanjšam vrednost bossHP za ena, nato pa z metodo Substring() tekstu, ki prikazuje življenjske točke, odstranim zadnji znak.



Slika 17: Koda za spremembo besedila

# Zvok

## Zvočni efekti

Igri je manjkal le še zvok. Da bi komplementiral izgled igre, sem na spletnih straneh z zvočnimi efekti za prosto uporabo poiskal zvoke v stilu konzol Atari 2600 in NES. V igri so uporabljeni trije različni zvoki. Zvok meča, ki se predvaja vsakič, ko igralec napade. Zvok, ki se predvaja ob smrti sovražnika oz. ko igralec udari šefa. Zadnji uporabljen zvok se predvaja ob koncu igre.

## Glasba

Tudi glasba igre je v podobnem stilu kot zvočni efekti. Na internetu sem našel album Songs from an Unmade World izvajalca Visager, ki je na voljo za prosto uporabo. Za glasbo, ki se predvaja med igranjem, sem izbral dve pesmi s tega albuma. Ob začetku igre se naključno predvaja ena izmed pesmi.

# Izdelava glavnega menija

Sedaj, ko sem dokončal igro, sem moral narediti še glavni meni, v katerem lahko igralec zažene igro, pogleda navodila za igranje in kontrole ali pa zapre igro. Ustvaril sem novo sceno imenovano mainMenu. Ozadju sem določil temnosivo barvo in ustvaril nov okvir. Okvirju sem dodal novo skripto, MenuController s katero bom kontroliral vse dogajanje v tej sceni. Tudi meniju sem dodal svojo glasbo. V okvirju sem ustvaril štiri tekstovne objekte po katerih bo igralec lahko usmerjal, in dva dodatna tekstovna objekta. S tema dvema tekstoma prikazujem najvišji rezultat igralca in avtorja glasbe, ki je uporabljena v igri, na spodnji strani ekrana. Z pritiskom na gumb 'NOVA IGRA' lahko igralec zažene igro. Gumb 'NAVODILA' igralcu prikaže navodila za igranje igre, gumb 'KONTROLE' pa igralcu pove, kako se igra nadzoruje. Z izbiro gumba 'IZHOD' igralec lahko zapre igro. Igralec usmerja po meniju s pritiski na tipki gor in dol. Izbran tekst je obarvan belo, medtem ko so drugi svetlosive barve.



Slika 18: Glavni meni igre

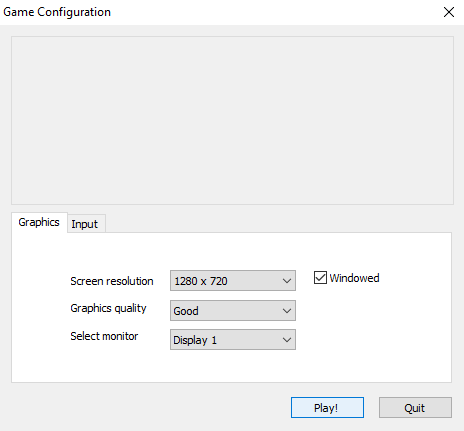
Izbranemu tekstu sledim s števcem, ki lahko ima vrednost od 1 do 4. Ob vsakem pritisku na puščico povečam ali zmanjšam števec, nato pa vse tekste v metodi ustrezno prebarvam. Te spremembe izvajam v korutini, da je med zaznanimi pritiski malo premora (0.15 sekunde). Če bi spremembe izvajal v normalni metodi, bi bilo navigiranje po meniju prehitro. Ob pritisku na tipko 'Enter' se vsebina tekstov glede na števec ustrezno spremeni. Za ogled navodil spreminjam vsebino prvih dveh tekstov menija, za ogled kontrol pa sem ustvaril dva nova tekstovna objekta, ker je vsebina predolga. Ob pritisku na tipko 'Esc' se igralec vrne nazaj na glavni meni. V meniju in sami igri sem onemogočil vidnost miške, ker vse kontroliranje poteka s tipkami.

# Testiranje

Ker je igra dokaj preprosta kakšno posebno testiranje ni bilo potrebno. Vso kodo sem med izdelavo sproti testiral, na kakšen večji problem pa nikoli nisem naletel. Ko sem z izdelavo zaključeval sem igro poslal sošolcem, da jo preizkusijo še na svojih računalnikih, in mi javijo kakšne probleme ki jih imajo z igre ali pa predloge za izboljšavo. Na podlagi povratnih informacij se nato igro še izboljšal.

# Navodila za uporabo

Igre za igranje ni treba inštalirati. Potreben je le zagon .exe datoteke. Ob zagonu si igralec izbere želeno resolucijo in kvaliteto grafike, ob pritisku na 'Play' pa se zažene igra. Igralec lahko po glavnem meniju usmerja z tipkama gor in dol, želeni gumb pa izbere s tipko 'Enter'. V sami igri se igralec premika s puščicami, napada pa s tipko 'Space'. Cilj igre je ubiti čim več sovražnikov, ob dotiku z njim pa je igre konec. Ob koncu igre lahko igralec ponovno zažene igro s pritiskom na tipko 'R' ali pa se vrne nazaj na glavni meni s pritiskom na tipko 'Q'.



Slika 19: Izbira resolucije

# Zaključek

Med izdelavo seminarske sem se naučil veliko o oblikovanju iger, risanju, programiranju z jezikom C# in delu s razvojnim okoljem Unity. Ugotovil sem da je izdelava videoigre zelo zahteven proces. Zelo hitro lahko pride do kakšne napake ali hrošča, še posebej, ker sem se med izdelavo igre sproti tudi učil. Velikokrat sem se več ur ukvarjal z nekim problemom, ki ga nisem znal rešiti. V takih situacijah sem si pomagal z internetom, na katerem sem lahko hitro našel pomoč. Sicer po izdelavi igre nisem bil čisto zadovoljen. Najbolj me motijo animacije igralca, ki bi lahko bile bolj realistično narisane. Prav tako si želim, da bi bilo premikanje sovražnikov bolj kompleksno. Igro bi lahko tudi izboljšal z več tipi sovražnikov, z več orožji ipd. A kljub temu menim da sem se naučil veliko uporabnih osnov, ki mi bodo koristile v mojih naslednjih projektih.

# Viri in literatura

* Space Shooter Tutorial. [online]. *Unity*. Dostopno na spletnem naslovu: <https://unity3d.com/learn/tutorials/projects/space-shooter-tutorial>
* 2D Roguelike tutorial. [online]. *Unity*. Dostopno na spletnem naslovu: <https://unity3d.com/learn/tutorials/projects/2d-roguelike-tutorial>
* Unity (game engine). [online]. *Wikipedia: the free encyclopedia*. [Zadnja sprememba 14.4.2016]. [Citirano 12.4.2016]. Dostopno na spletnem naslovu: <https://en.wikipedia.org/wiki/Unity_(game_engine)>

# Priloge

* Priloga 1: CD s programom in izvorno kodo