SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I

INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

DRAGON LAND ADVENTURES – 2D PLATFORMER GAME

Seminarski rad

Matej Peršić

Osijek, 2022.

##### SADRŽAJ

[1. UVOD 1](#_Toc106271628)

[1.1. Ciljevi igre 1](#_Toc106271629)

[2. IGRAČ 2](#_Toc106271630)

[2.1. Animacije 4](#_Toc106271631)

[2.2. Skripte 5](#_Toc106271632)

[2.3. Healthbar 6](#_Toc106271633)

[3. ZAMKE 6](#_Toc106271634)

[3.1. ArrowTrap 6](#_Toc106271635)

[3.2. Firetrap 7](#_Toc106271636)

[3.3. Saw 8](#_Toc106271637)

[3.4. Spikehead 9](#_Toc106271638)

[3.5. Spikes 10](#_Toc106271639)

[4. NEPRIJATELJ 11](#_Toc106271640)

[4.1. Patrola 11](#_Toc106271641)

[4.2. Melee Enemy 12](#_Toc106271642)

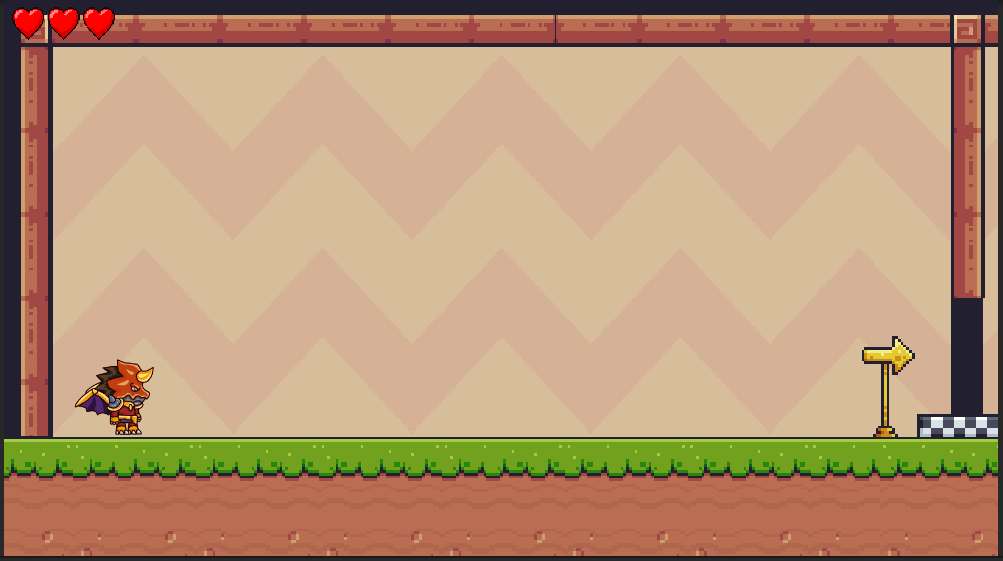
[4.3. Ranged Enemy 12](#_Toc106271643)

[5. KAMERA 15](#_Toc106271644)

[6. ZVUK 16](#_Toc106271645)

[ZAKLJUČAK 17](#_Toc106271646)

# UVOD

Dragon land adventures je 2D platformer napravljen u Unityju. Na početku same igrač se upoznaje sa zmajom, s kojim će se kretati kroz igru i svladavati njene prepreke. Igrač na raspolaganju ima kontrole za kretanje lijevo-desno, opciju za skok i vatrene kugle s kojima će se moći boriti protiv različitih neprijatelja. Njegov model, kao i inicijalna scena iz igre se mogu vidjeti na slici dolje.

Za model igrača, neprijatelja, zamki i elemenata sa scene poput zidova i tla, korišten je moćan Unity asset store i njegovi besplatni asseti. Radi lakšeg snalaženja unutar projekta, njegova struktura je podijeljena u šest mapa, u kojima se nalaze pripadajuće datoteke, spriteovi, audio datoteke, C# skripte, animacije i ostalo. Navedena struktura se može vidjeti na slici dolje.  


## Ciljevi igre

**C**ilj igre je savladati što više prepreka i sa što više preostalih života doći do kraja igre, kako bi se zmaj sigurno i u komadu vratio kući. Na tom putu će naići na mnoge prepreke, zamke i neprijatelje.

# IGRAČ

Igrač se može kretati lijevo-desno, skakati i ispaljivati vatrene kugle. Kada se igrač kreće lijevo ili desno, sprite Prefaba player također promijeni x-scale, što rezultira u tome da se dobije efekt koji izgleda kao da se zmaj uputio u pravom smjeru na odgovarajuću kontrolu. Navedeno se može vidjeti na slici dolje.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika na kojoj se prikazuje tekst, isječak crteža

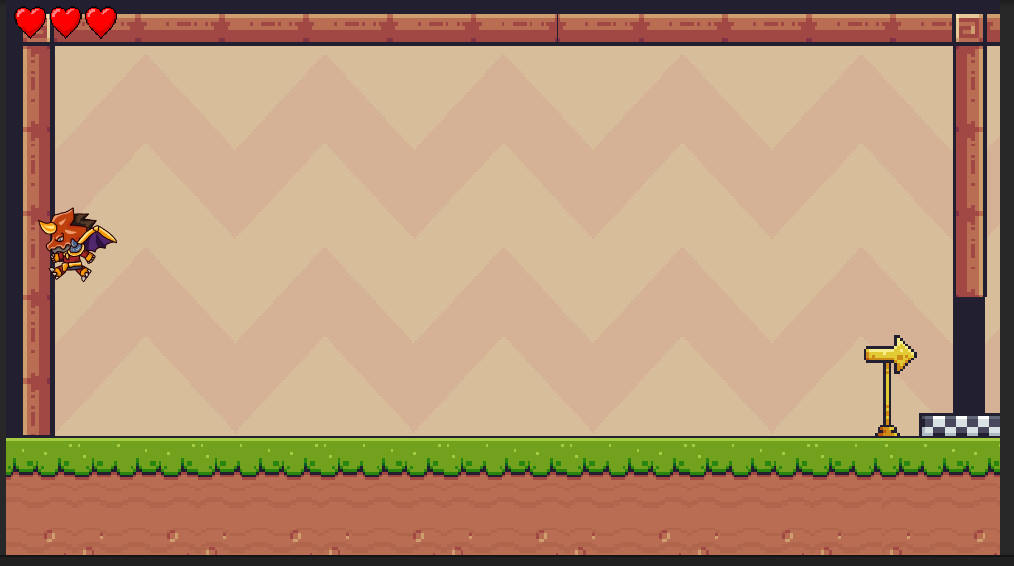
Opis je automatski generiran

Klikom na lijevu tipku miša, igrač može ispaliti vatrenu kuglu. Ona je korisna u borbi s neprijateljima, prati pravac dok njen collider nešto ne pogodi ili joj istekne vrijeme trajanja. Za ispaljivanje projektila i postavljanje njegove inicijalne pozicije korišten je child objekt na Playeru, firepoint.





Za skok, igrač može pritisnuti spacebar. Sa skokom igrač može izbjeći neprijatelje, preskočiti prepreke i penjati se po zidu. Penjanje po zidu može biti korisno u situaciji kada je na ekranu puno neprijatelja, pa im igrač želi pobjeći iz dometa napada, a konačni efekt izgleda kao da se igrač zalijepio na zid, dok ne pritisne drugu tipku. Rezultat se može vidjeti na slici dolje.





## Animacije

Za animacije igrača, na Prefabu Player definirano je 6 stanja. Svakom od njih pridjeljena je odgovarajuća animacija, popraćena zvukom. Prijelaz iz pojedinog stanja može se vidjeti na slici dolje.

Slika na kojoj se prikazuje tekst, na zatvorenom

Opis je automatski generiran

Zbog njene najmanje kompleksnosti, prikazana je Attack animacija. U njoj se u rasponu od 0:00-0:11 na Playeru u sprite rendereru mijenjaju 3 različita spritea. Zbog malog vremenskog intervala, to izgleda kao fluidna animacija. Navedeni spriteovi se mogu vidjeti na slikama dolje.

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

Slika na kojoj se prikazuje tekst

Opis je automatski generiran

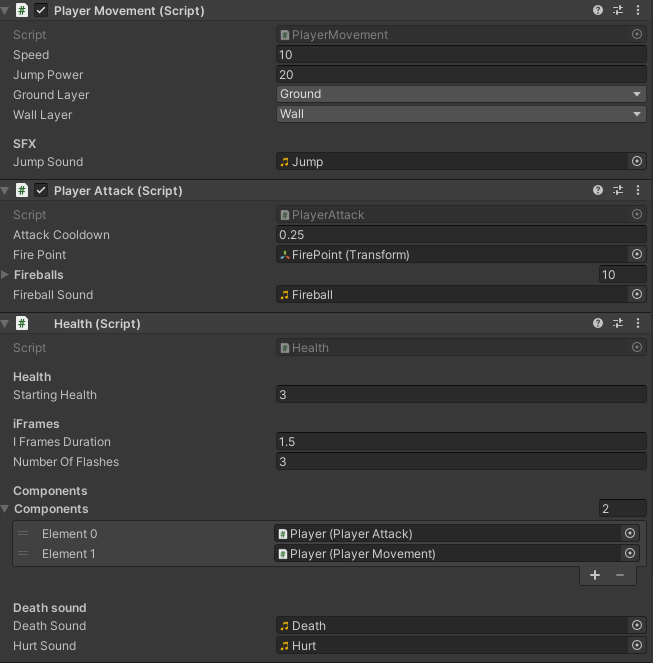
Slika na kojoj se prikazuje tekst, isječak crteža

Opis je automatski generiran

Ostale animacije za Playera, pa i ostale elemente su napravljene slično, uz malo veću kompleksnost zbog većeg broja spriteova i dužeg vremenskog intervala animacije.

## Skripte

Na prefabu Player, nalaze se 3 skripte, Player Attack, Player Movement i Health skripta. Unutar njih je odrađena logika za kretanja igrača, postavljene su vrijednosti poput brzine kretanja igrača, zvukova za pojedine animacije, odrađena je logika za collidere s ostalim objektima u sceni i ostalo. Na prefabu su definirani i layeri igrača i njegov tag, koji će biti korišten u collider metodama ostalih elemenata. Izgled SerializedFieldova i skripti na igraču sa svim definiranim vrijednostima, može se vidjeti na slici iz Unityja dolje.



## Healthbar

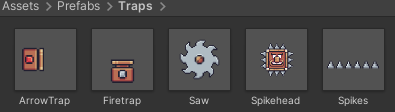
Za indikaciju koliko je igraču preostalo života, u gornjem lijevom kutu se nalazi Healthbar. U ovom slučaju, to su 3 srca jer je Health igrača 3. Navedeno se ostvaruje pomoću bindanja Player Health skripte na HealthBar objekt u sceni, a rezultat se može vidjeti na slici dolje.





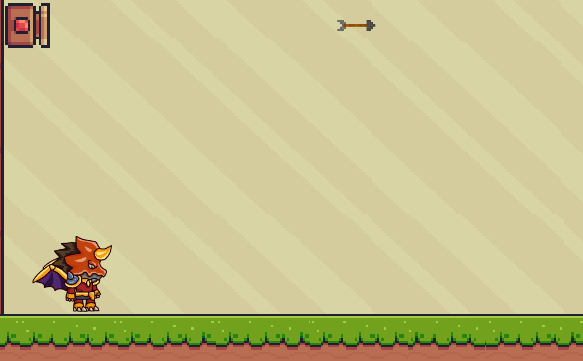
# ZAMKE

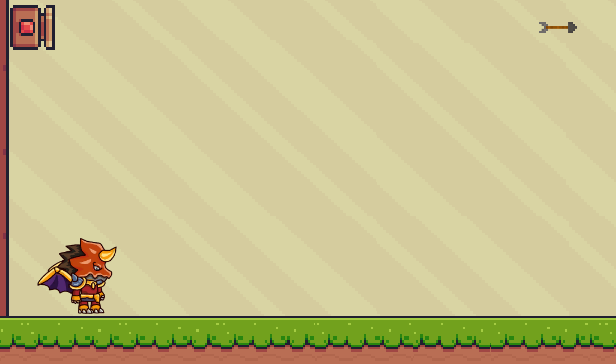
U igri igrač može izgubiti živote na različite načine, a jedan od njih je preko zamki. Svaka od zamki različito radi, a u igri ih ima 5 vrsti. Zamke se mogu vidjeti na slici dolje.



## ArrowTrap

ArrowTrap prefab je zamka koja prema svom AttackCooldownu ispaljuje strijelu u jednom pravcu. Poput igrača, ona ima firepoint s kojeg strijele kreće letjeti. Na slikama dolje se može vidjeti kako to izgleda.





Na slici iz Unityja dolje se može vidjeti malo informacija o ArrowTrapu kao i njegova pripadajuća skripta.

Slika na kojoj se prikazuje tekst, snimka zaslona, monitor

Opis je automatski generiran

## Firetrap

Firetrap je prefab koji kada igrač stane na njega, nakon delaya ispali vatru. Aktivan je određeno vrijeme i ako igrač stoji na njemu kada je aktivan, oduzima mu health, tj radi damage.   
Kada je u aktivnom stanju, točnije triggeran, trap poprima crvenu boju, čime se igraču daje do znanja da će se nešto dogoditi. Nakon toga, ispušta se vatra, i prijelazi nazad u idle stanje. Primjer animacije se nalazi na slici dolje.



Slika na kojoj se prikazuje tekst, isječak crteža

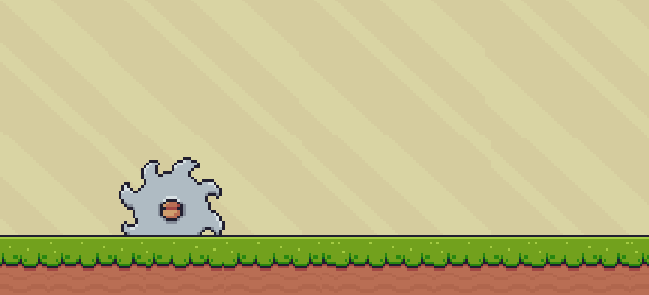
Opis je automatski generiran

Slika na kojoj se prikazuje tekst, isječak crteža

Opis je automatski generiran

## Saw

Saw je prefab koji se kreće određenog definiranog dometa preko fielda Movement Distance, rotira i ako dodirne igrača, oduzima mu health. Može se vidjeti u akciji na slici dolje.







## Spikehead

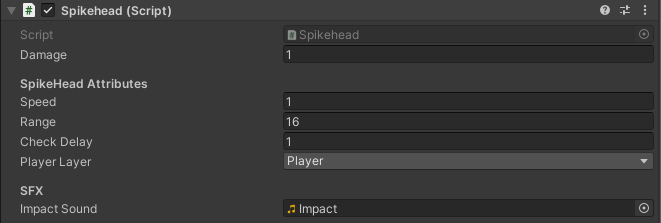
Spikehead je prefab koji prati Playera unutar određenog definiranog dometa. Prati ga pravocrtno u svim smjerovima, ako mu je unutar dometa vidljivosti. Na slici se može vidjeti u inicijalnom stanju, i aktiviranom stanju gdje je promijenio poziciju.



Slika na kojoj se prikazuje tekst, igračka

Opis je automatski generiran

Na slici iz Unityja dolje se mogu vidjeti fieldovi iz pripadajuće skripte spikehead prefaba.



## Spikes

Najjednostavniji trap prefab, koji je statičan i radi damage na Playeru kada ga on dotakne. Može se vidjeti u akciji na slici dolje.

Slika na kojoj se prikazuje tekst

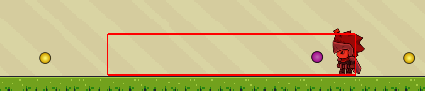
Opis je automatski generiran

# NEPRIJATELJ

Postoje dvije vrste s kojima se igrač može susresti, a to su melee i range, oba izgledaju poput viteza. Melee enemy ima manji domet napada i njegovo je oružje mač, dok ranged enemy ima veći domet napada i za napad koristi vatrene kugle, kao i igrač. Oba neprijatelja koriste jednaku logiku, ranged enemy je napravljen iz melee enemyja, pa je zbog toga ranged enemy prefab napravljen iz melee enemy prefaba, uz manje izmjene zbog napada.

## Patrola

Neprijatelji mogu patrolirati unutar dometa definiranim lijevim i desnim rubom. Unutar prostora definiranog rubovima, oni se mogu kretati i napasti igrača. Primjer ranged enemy patrole (kao i ranged enemy dometa unutar nje) može se vidjeti na slici dolje. Žuti kružići predstavljaju lijevi i desni rub, na njima enemy iz patrole prelazi u idle stanje na definirano vrijeme, nakon čega se okreće na drugu stranu i nastavlja s patrolom.



## Melee Enemy

Kao prethodno navedeno, melee enemy ima jednako definirana stanja, logiku i skoro sve animacije kao ranged enemy. Koriste isti prefab, a jedina je razlika u attack animaciji i logici. Primjer melee enemyja u akciji se može vidjeti na slici dolje.



Slika na kojoj se prikazuje tekst

Opis je automatski generiran

## Ranged Enemy

Kako bi se razlikovao od melee enemyja, ranged enemy ima crvenkastu boju. Za napad su mu definirani cooldown, range i damage, koji se nalaze unutar Ranged Enemy skripte. Primjer ranged enemyja u patroli se nalazi na slici dolje.

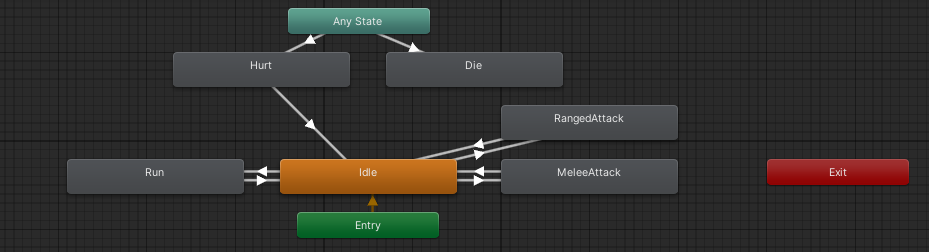




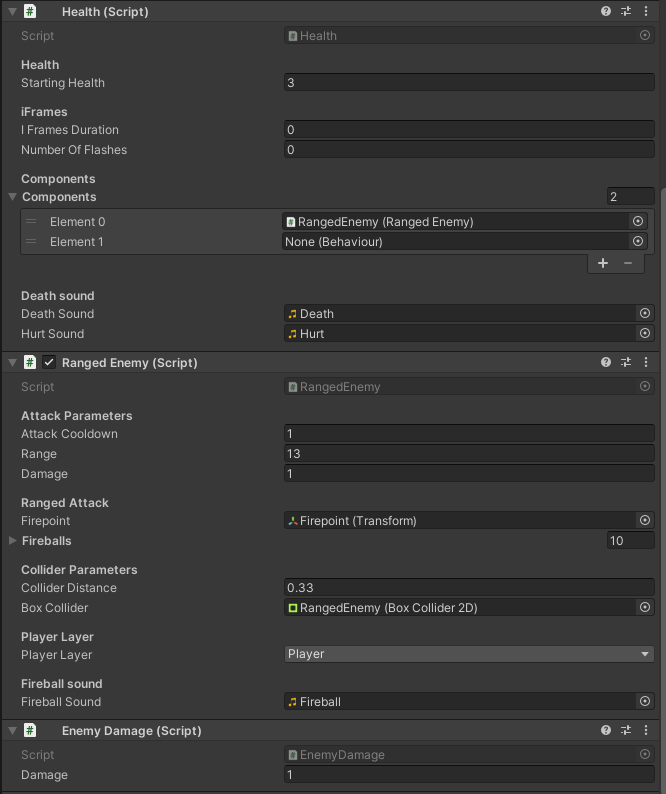
Slika na kojoj se prikazuje tekst, vektorska grafika

Opis je automatski generiran

Ranged enemyju su definirana ista stanja za animacije kao i melee enemyju, a mogu se vidjeti na slici dolje.

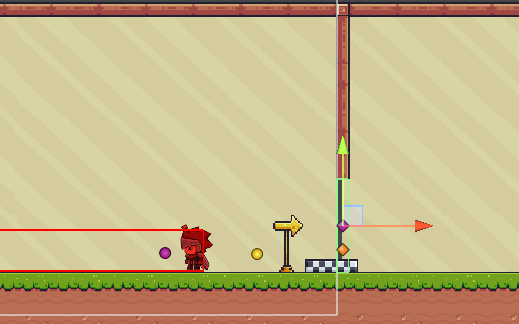


Na slici se dolje nalaze skripte i fieldovi kojima se definira ponašanje ranged enemyja unutar Unityja.

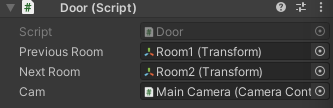


# KAMERA

Za kretnju unutar igre postoje dva pristupa, kamera koja fluidno prati igrača po ekranu dok se kreće i kamera koja „skače“ po ekranu, u ovom slučaju, između soba, budući da su nivoi unutar igre podijeljeni po prostorijama. Unutar igre je korišten drugi pristup, jer se pokazao kao fluidnije rješenje problema. Za odrađivanje navedenog, napravljena je Camera Controller skripta i između prostorija je postavljen Door element, koji ima referencu na Main Camera objekt i prostorije na nivou. Door element (ljubičasti romb) na nivou se može vidjeti skupa s njegovim box colliderom na slici dolje.



Dio Door elementa vezan uz logiku prebacivanja između prostorija se može vidjeti unutar Unityja na slici dolje.



# ZVUK

Za zvuk unutar igre, pozadinsku glazbu i sviranje zvuka na pojedine događaje, kao što su ispaljivanje strijele, vatrenih kugli i ostalog, koristi se SoundManager objekt u sceni. Na njemu je Sound Manager skripta, koja je singleton kako bi se mogla pozivati unutar ostalih skripti, kada je potrebno svirati glazbu za pojedini događaj. Kod vezan uz navedeno se nalazi na slici dolje.

**using** **UnityEngine**;

**public** **class** **SoundManager** : MonoBehaviour

{

**public** **static** SoundManager instance { **get**; **private** **set**; }

**private** AudioSource source;

**private** **void** **Awake**()

{

instance = **this**;

source = GetComponent<AudioSource>();

//keep this object even when we go to new scene

**if**(instance == **null**)

{

instance = **this**;

DontDestroyOnLoad(gameObject);

}

// destroy duplicate obejct

**else** **if**(instance != **null** && instance != **this**)

{

Destroy(gameObject);

}

}

**public** **void** **PlaySound**(AudioClip \_sound)

{

source.PlayOneShot(\_sound);

}

}

# ZAKLJUČAK

Napravljena igra pokriva dosta ključnih elemenata za rad unutar Unityja, počevši od spriteova, pa sve do složenijih animacija, zvukova i ostalih stvari. Ima sve elemente za prilično kompleksnu igru, a jedino ograničenje s projektom u ovakvom stanju je mašta osobe koja dizajnira nivoe. Projekt je dobro strukturiran s podjelom u mape, pa je snalaženje po istom brzo i jednostavno. Iz dosad kreiranih elemenata lako je napraviti nove, budući da je logika za različite vrste neprijatelja i zamki pokrivena. Unity je moćan alat, koji skupa sa svojim asset storeom omogućuje programeru stvaranje kompletnih, lijepih i zanimljivih igara u relativno malo vremena. Projekt se nalazi na githubu na linku:

https://github.com/mpersic/DragonLandAdventures