Izrada platformske igre u okruženju Unity

Degmečić, Filip

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:886451

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-11-29**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Sveučilište Jurja Dobrile u Puli
Fakultet Informatike u Puli

Filip Degmečić

Izrada platformske igre u okruženju Unity

Završni rad

Pula, rujan, 2019. godine

Sveučilište Jurja Dobrile u Puli
Fakultet Informatike u Puli

Filip Degmečić

Izrada platformske igre u okruženju Unity

Završni rad

Filip Degmečić
Izrada platformske igre u okruženju Unity
Završni rad
JMBAG: 0303054201, redoviti student
Studijski smjer: Informatika
Predmet: Napredne tehnike programiranja
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Mentor: doc. dr. sc. Tihomir Orehovački

Pula, rujan, 2019. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Filip Degmečić, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, rujan, 2019. godine

IZJAVA

o korištenju autorskog djela

Ja, Filip Degmečić dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „Izrada platformske igre u okruženju Unity“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, rujan, 2019. godine

Potpis

_____

Sadržaj

1 Uvod 1
2 Motivacija 2
3 Scene i objekti 4
3.1 Scene 4
3.2 Objekti 8
4 Skripte 13
5 Animator 18
6 Izrada nivoa 19
7 Zaključak 20
Literatura 21
Popis slika 22
Sažetak 23
Abstract 24

1 Uvod

Na tržištu video igara postoji neprestana potražnja za novim proizvodima, bilo da su oni inovativne ideje ili novi oblici klasičnih tipova igara. Rogue Caves je igra koja pripada žanru 2D platformera te je izrađena u okruženju Unity. Za ovakvu bi igru ciljani korisnici bili ljubitelji klasičnih 2D igara poput „Super Mario Bros." ili „Sonic The Hedghog". Aplikacija sadrži razne poznate funkcionalnosti koje su očekivane u igrama ovog oblika. Uz progresivnije teže razine, igračima je dostupna mogućnost skakanja i pucanja pomoću kojih moraju prikupiti određeni broj ključeva koji im osigurava prijelaz na sljedeću razinu. Jedna od glavnih prednosti 2D igara je jednostavnost koja je odlika i ove igre. Korisnicima je odmah jasno što i kako moraju učiniti kako bi došli do cilja.

U današnje vrijeme složenu video igru izrađuju skupine od 100 do 200 pojedinaca koje na određenom projektu rade i do nekoliko godina. Takvi naslovi lako ostvaruju velike prodaje te donose veliku zaradu. U prošlosti su za izradu platformera ili manjih igara bile potrebne skupine od 20 do 30 pojedinaca dok danas, uz pomoć alata i okruženja, to može napraviti nekoliko ili pak jedna osoba. Takve se igre, nezavisne od velikih studija, izdavača i izvora financijskih sredstava, zovu „indie" igre (nezavisne igre) te su često usmjerene na inovativne ideje.

Pri razvoju video igara koriste se razni alati i okruženja dok je u ovom projektu korišteno okruženje Unity. Unity je alat koji je zbog svoje jednostavnosti najpopularnije besplatno okruženje za izradu video igara. Unity velikom broju korisnika pruža jednostavnost korištenja te velik broj materijala za pomoć. S obzirom da Unity sadrži veliki broj korisnika, korisnici su spremni pomoći jedni drugima te često izrađuju edukativne sadržaje poput pisanih vodiča ili videa. Korisnicima je na raspolaganju i velik broj drugih dodataka, a to su prije svega razni alati koje je moguće iskoristiti u Unity projektu.

Cilj je ovog projekta upoznati se s procesom izgradnje igre uz pomoć okruženja Unity. Također će biti riječi o brojnim mogućnostima alata Unity, preprekama te problemima koje je potrebno riješiti kako bi se došlo do gotove video igre te načinima na koje se koriste razni alati razvijeni kao podrška alatu Unity.

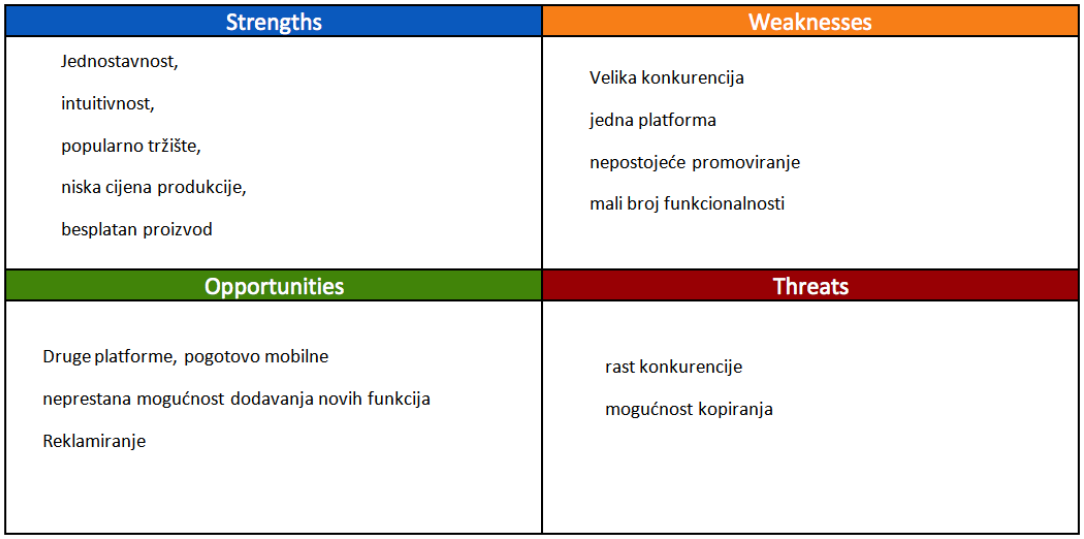
2 Motivacija

Kao što je već spomenuto, ciljano tržište su ljubitelji klasičnih igara, no to ne znači da Rogue Caves neće privući i ljubitelje novijih video igara. Postoji i mogućnost razvoja igre kao aplikacije za korištenje na mobilnim telefonima, ali to nije prvobitno tržište, već jedan od idućih koraka u razvoju. S obzirom da je ovakva vrsta igre na tržištu već 40 godina daje se naslutiti i da je konkurencija velika, no platformeri su i danas vrlo poželjni što se može vidjeti na primjerima igara poput „Hollow Knight-a“ (Slika 2) ili „Cuphead-a“. Neka od temeljnih obilježja ovih novih igara, koje ih čine konkurentnima na današnjem tržištu nakon toliko godina, su inovativna priča kao što je slučaj kod igre „Hollow Knight“. Uz već navedeno, postoji i određena količina zahtjevnosti igre koja je privlačna velikom broju igrača. Za neke je korisnike pak važan umjetnički stil odnosno postava u kojoj se igra odvija. Takva, zanimljiva postava nalazi se u video igri „CupHead“ koja je prikazana na Slici 1.

Slika 1. Prikaz sučelja iz igre „CupHead“

Slika 2. Prikaz sučelja iz video igre „Hollow Knight“

Usporedbom navedenih igara s Rogue caves može se zaključiti da sadrže puno više mogućnosti poput borbi protiv konačnih protivnika, životnih bodova i slično. Ova video igra uključuje osnovne, ključne funkcionalnosti 2D platformera što ju čini jednostavnom i intuitivnom. Swot analizom razrađene su prednosti i nedostaci ovakve video igre. (Slika 3)



Slika 3. SWOT analiza

Opis prednosti:

- Jednostavnost i intuitivnost vidljive su kroz brzo snalaženje u igri. Korisniku je odmah jasno tko je neprijatelj te što mora učiniti kako bi napredovao do sljedeće razine.
- Veliko tržište u stalnom razvoju omogućava velik broj prilika za prodaju.
- Pri izradi video igre korišten je alat Unity, koji je besplatan te nisu utrošena financijska sredstva na softver da bi izradili igru.
- Igra je besplatna i stoga privlačna velikom broju korisnika.

Koristi od ove aplikacije imali bi pretežno korisnici. Igra bi se također nalazila na platformi za preuzimanje kao što je npr. „Play Store“ te bi platformi za preuzimanje privuklo korisnike. Jedan od načina na koji bi ova video igra mogla osigurati određena sredstva bi bilo uvođenje plaćenih oglasa.

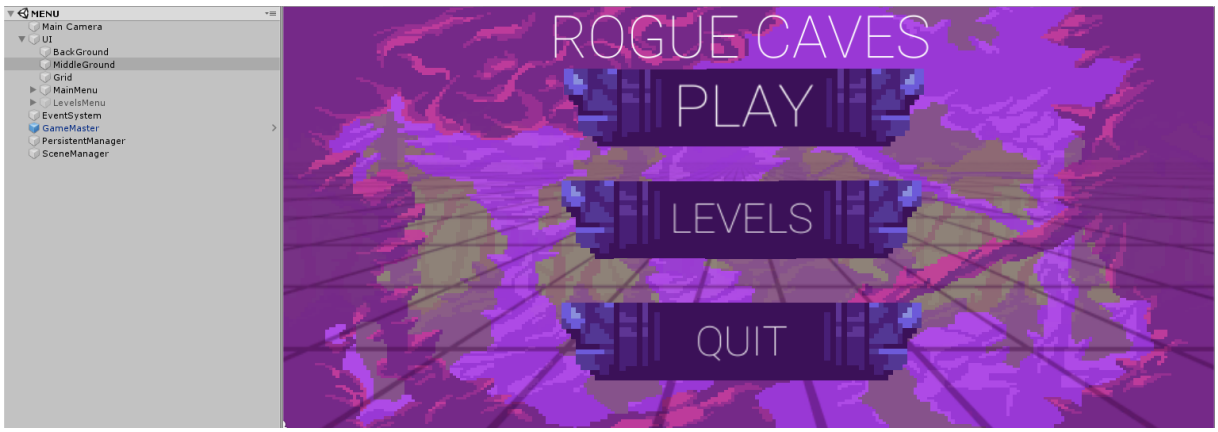
3 Scene i objekti

Alat Unity, jedan od najpopularnijih alata za izradu video igara korišten je pri izradi video igre Rogue Caves. Razlog odabira baš ovog alata njegova je jednostavnost te velik broj izvora koji mogu pomoći pri izradi video igre. Takvi izvori najčešće proizlaze iz velike zajednice korisnika. Često su to video uratci koji sadrže savjete za korištenje ovog alata. Unity alat privlačan je korisnicima i zbog mogućnosti razvoja na velikom broju platformi kao što su iOS, Windows te Android sustav.

Najvažnije sastavnice Unity alata su scene i objekti. Scena predstavlja skup objekata koji su organizirani tako da čine jednu razinu ili nešto drugo što želimo prikazati u igri. Kao takve, scene možemo koristiti za prikaz glavnog izbornika, scena s animacijom ili same video igre. Korisnik u sceni objektima dodjeljuje različite vrijednosti te im na taj način omogućava međusoban suodnos. Pomoću C# skripte moguće je dodijeliti različita svojstva objektima. Skripte omogućavaju izazivanje različitih događaja, mijenjanje vrijednosti različitih objekata ili upravljanje objektom. Na taj se način prikuplja i unos podataka samog korisnika. O scenama i objektima više će biti riječi u sljedećim poglavljima.

3.1 Scene

Glavna scena u projektu je glavni izbornik koji se korisniku pojavljuje nedugo nakon pokretanja video igre. Nakon izbornika slijede razine te scena koja sadrži priču u kojoj je igra smještena. Pregled objekata i glavnog izbornika vidljiv je na Slici 4.

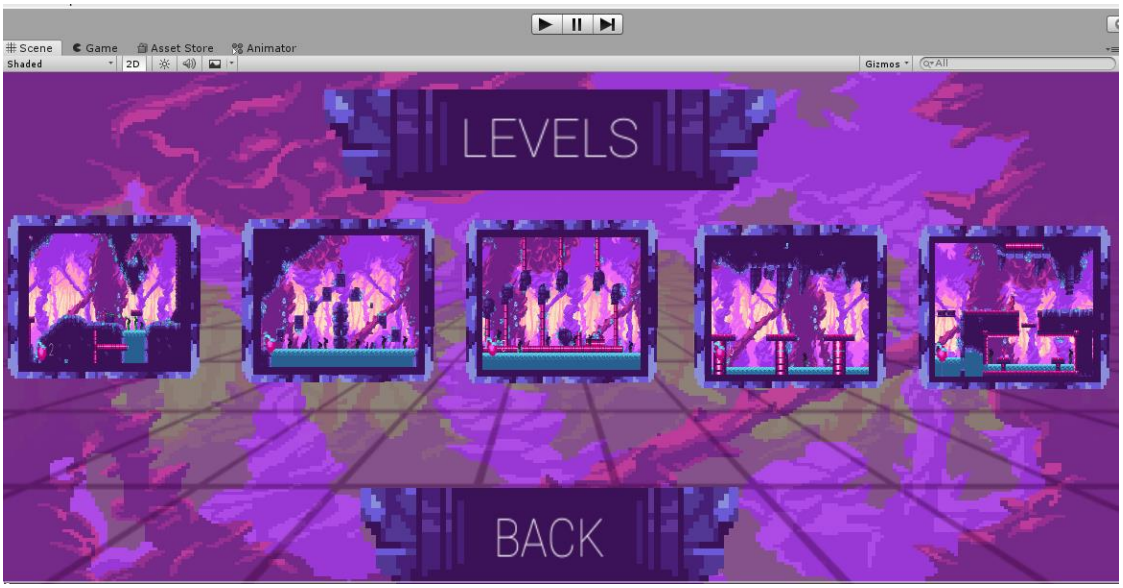


Slika 4. Pregled objekata i glavnog izbornika

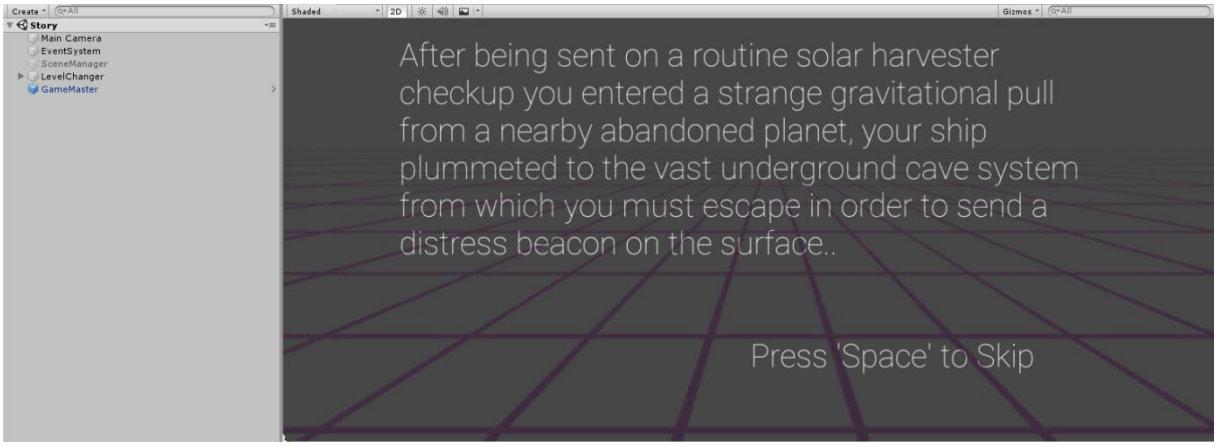
Na lijevoj strani nalaze se razni objekti koji tvore scenu. „Main Camera“ zaslužna je za ono što korisnik vidi na sučelju, dok se „Ui“ sastoji od gumbova koji omogućavaju

kretanje kroz izbornik. Objekti „Event System“, „GameMaster“, „Persistent Manager“ te „Scene Manager“ služe za upravljanje scenom i glazbom u izborniku.

Pritiskom na gumb „Play“ korisnik dolazi do prve scene (Slika 7) koja sadrži priču u kojoj je smještena igra, a zatim se pojavljuje scena prve razine. Pritiskom na gumb „Levels“ otvara se izbornik u kojem korisnik može odabrati određenu razinu kojom želi započeti igru što se može vidjeti na Slici 6.



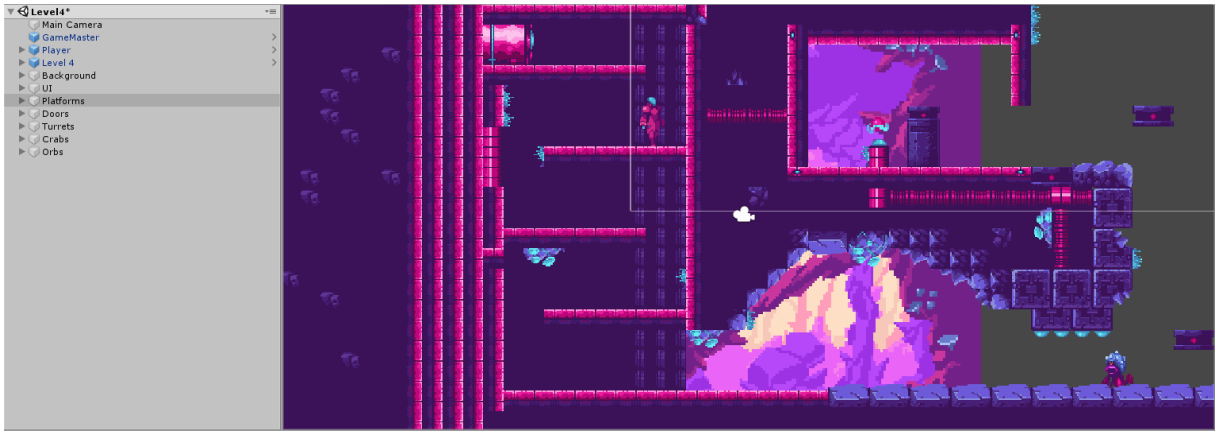
Slika 6. Izbornik za razine



Slika 7. Scena s pričom

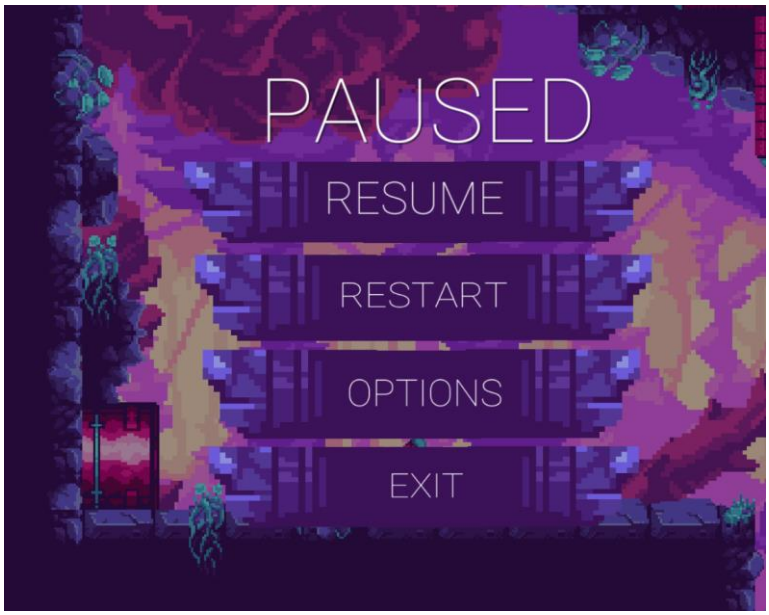
Scena s pričom (Slika 7) sadrži slične objekte kao scena glavnog izbornika. Objekti te scene su „Main Camera“, „Event System“, „Game Master“ i „Scene Manager“ te objekt „Level Changer“. Objekt „Level Changer“ korisniku s pritiskom gumba „Space“

daje mogućnost izostavljanja scene s pričom kako bi odmah mogao prijeći na igranje. Pregled objekata sadržanih unutar razine 4 prikazan je na Slici 8.



Slika 8. Pregled objekata na razini 4

Ostali objekti koji se nalaze u sceni za pojedine razine su: „Player“, „Level4“, „Background“, „Platforms“, „Doors“, „Turrets“, „Crabs“ i „Orbs“. Objekt „Player“ sadrži razne sastavnice koje omogućavaju upravljanje likom te mogućnost suodnosa lika s ostalim objektima u sceni. Unutar objekta „UI“ nalazi se izbornik pomoću kojeg se igra može trenutačno zaustaviti tijekom igranja. (Slika 9).



Slika 9. Pregled izbornika za stanku unutar razina

Pritiskom na gumb „Esacpe“ otvara se izbornik (Slika 9) koji sadrži nekoliko mogućnosti kao što su nastavak igre, ponovni početak iste razine, izazak u glavni

izbornik ili mogućnosti (Options). Mogućnosti omogućuju upravljanje zvukom u igri (Slika 10).

Slika 10. Pregled izbornika za podešavanje zvuka

Spomenuti izbornik sastoji se od 3 klizača pomoću kojih se može upravljati glasnoćom zvukova unutar igre. „Master“ upravlja sveukupnim zvukom u igri, dok se „Music“ odnosi samo na glazbu koja se može čuti prilikom igranja. Postavka „SFX“ odnosi se na zvukovne efekte koji se čuju tijekom igre. Takvi su zvukovi primjerice zvukovi pucanja, zvukove napada neprijatelja i slično. Kada igrač uspostavi zadovoljavajuću razinu zvuka, gumbom „Back“ dolazi do prethodnog izbornika.

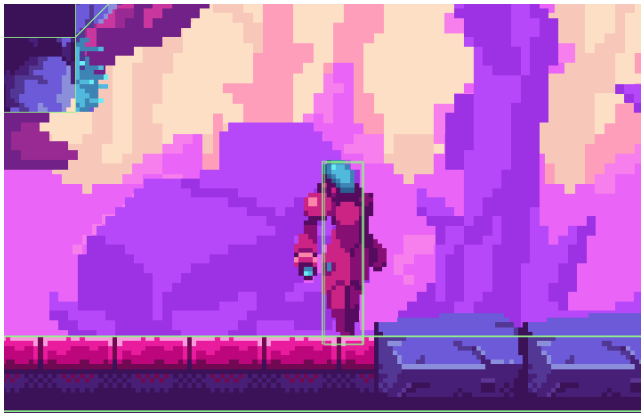
3.2 Objekti

Slijedi pregled raznih objekata korištenih u izgradnji video igre Rogue Caves, s naglaskom na objekt „Player“.

Slika 11. Pregled inspektora za objekt „Player“

Na Slici 11 prikazan je inspektor za objekte. Sastavnica „Animator“ zaslužna je za razne animacije koje model lika prikazuje prilikom igranja. „Rigidbody 2D“ liku pruža fizička svojstva te njime upravlja fizički sklop ugrađen u alat Unity. Taj sklop liku dodaje masu te na njega utječe sila gravitacije. Na lik može djelovati i određena sila koja ga može pokretati. Za kretanje lika na pojedinoj razini vrlo su važni izbornici „Box Collider 2D“ i „Circle Collider 2D“. Navedene sastavnice omogućavaju korelaciju lika i mape.

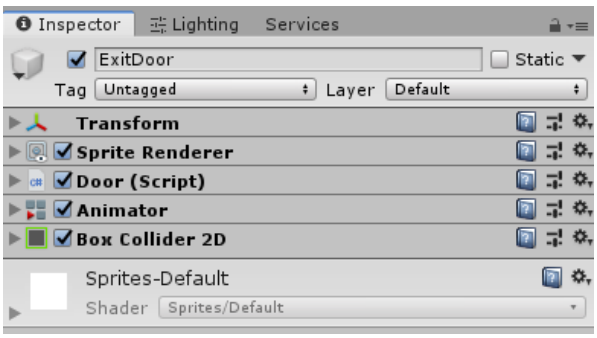
Slika 12 prikazuje sudarač (collider) lika te sudarač mape. „Collider“ lika prikazan je zelenim linijama koje se nalaze oko modela lika. Kada mapa ne bi imala „collider“ tada bi lik u video igri jednostavno propao kroz razinu. „Collider“ se također može koristiti i kao okidač. Ako lik dođe u dodir s objektom koji sadrži „collider“ i skriptu pomoću koje izazivamo nekakav efekt ili promjenu na liku, tada „collider“ može vršiti razne provjere. To se prvotno odnosi na provjeru je li „collider“ lika došao u međuodnos sa „collider-om“ koji sadrži skriptu.



Slika 12. „Collider“ objekta „Player“

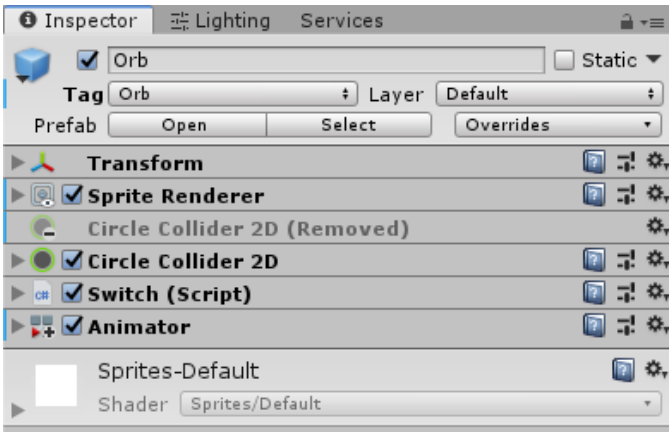
Nadalje, igrač sadrži „Sprite Renderer“ koji je zaslužan za vizualno prikazivanje lika. „Player Controller“ je skripta koja omogućava upravljanje likom i u sebi sadrži razne mogućnosti koje su pobliže prikazane u sljedećem poglavlju.

Objekti „Level4“ i „Background“ sadrže razne sastavnice koje čine mapu i njenu pozadinu. „Background“ također sadrži skriptu za neprestano stvaranje pozadine prilikom napretka lika kroz mapu. „Doors“ sadrži objekte koji predstavljaju ulaz i izlaz iz razine. Na kraju svake razine nalaze se vrata koja igrač mora otključati da bi mogao napredovati na sljedeću razinu. Inspektor objekta „Doors“, točnije izlaznih vrata, prikazan je na Slici 13.



Slika 13. Inspektor za objekt „ExitDoor“

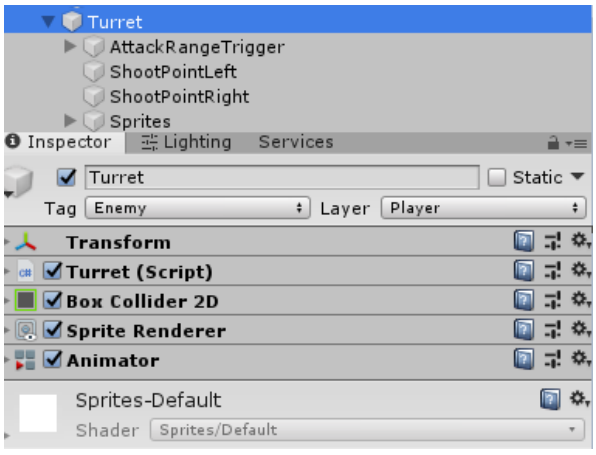
Prikupljanjem objekata „Orb“, koji se nalaze na raznim mjestima u video igri, igrač osigurava prijelaz na sljedeću razinu. Inspektor za objekt „Orb“ prikazan je na Slici 14.



Slika 14. Inspektor za objekt „Orb“

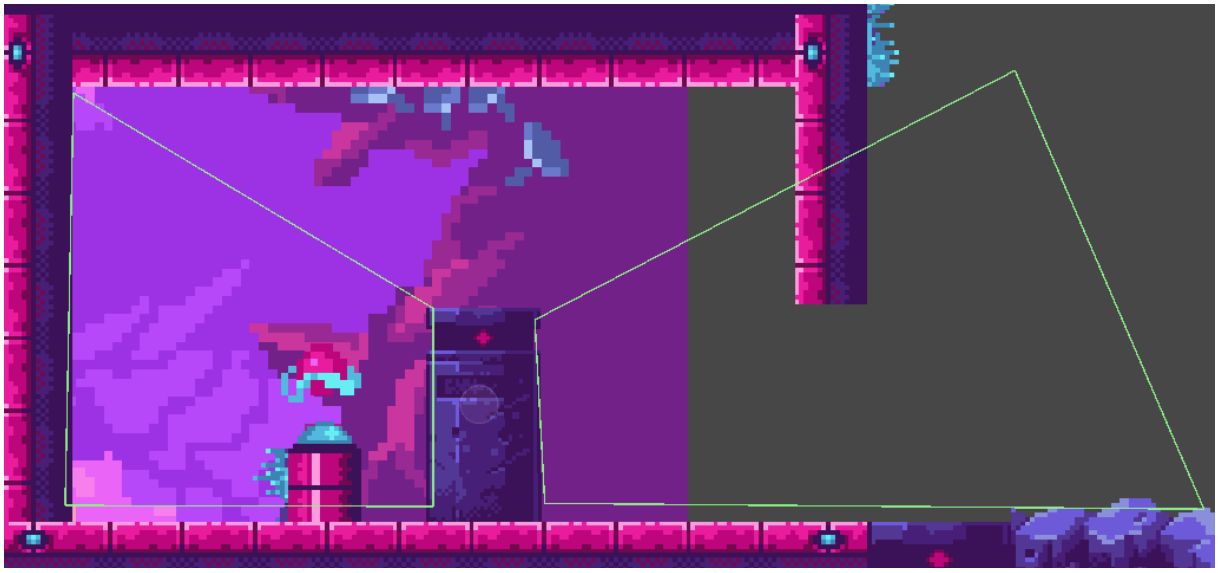
S napretkom kroz razine i prikupljanjem objekata „Orb“, njihov se broj smanjuje, što je prikazano pokazateljem na sučelju kojeg igrač može vidjeti dok je aktivna scena pojedine razine.

Neprijatelji protiv kojih se lik mora boriti nalaze se pod objektima „Turrets“ i „Crabs“. Objekt „Turret“ je neprijatelj koji u određenom dometu napada lika. Napad takvog objekta sastoji se od projektila koje ispaljuje te na taj način liku oduzima životne bodove. Objekt „Turret“ sadrži u sebi skriptu koja njime upravlja, što je prikazano u pregledu inspektora za taj objekt (Slika 15).



Slika 15. Inspektor i hijerarhija objekta „Turret“

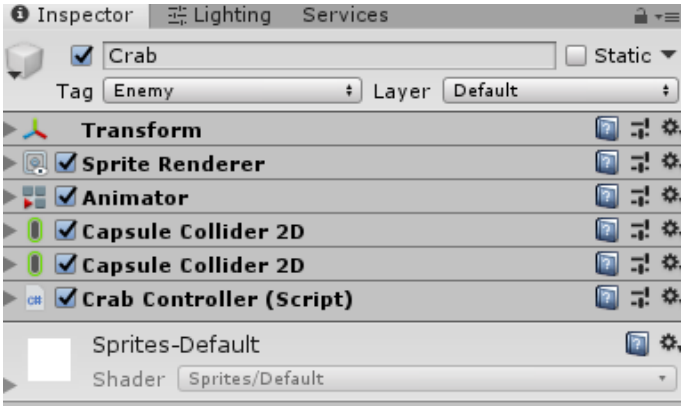
U gornjem dijelu Slike 15. prikazana je hijerarhija objekta koji čini jedan „turret“. Objekt „AttackRangeTrigger“ određuje domet objekta „turret“ (Slika 16).



Slika 16. Prikaz „collider-a“ objekta „ShootPointLeft“ i „ShootPointRight“

Objekti „ShootPointLeft“ i „ShootPointRight“ određuju na kojem će mjestu biti projektili ispaljeni na lik u video igri. Objekt „Sprite“ sadrži grafički prikaz ovog neprijatelja. Sastavnice koje sadrži ovaj objekt su skripta koja njime upravlja, „Box Collider 2D“, „Sprite Renderer“ i „Animator“.

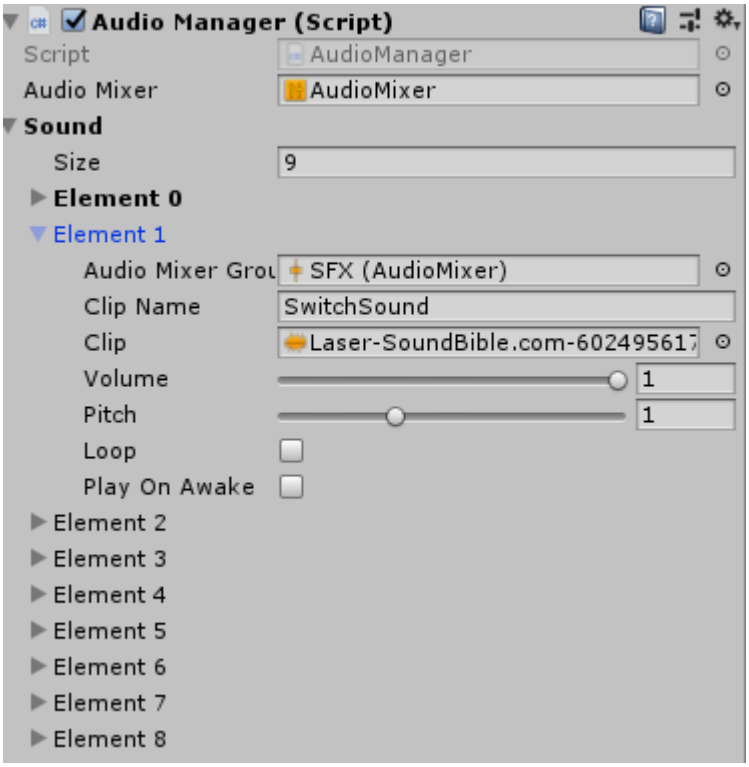
Sljedeći neprijatelj nalazi se pod objektom „Crab“.



Slika 17. Inspektor za objekt „Crab“

U inspektoru ovog objekta (Slika 17) pronalazimo sastavnicu: „Sprite Renderer“, „Animator“, dva „Capsule Collider 2D“ te skriptu „Crab Controller“. Skripta „Crab Controller“ upravlja tim neprijateljem te određuje način na koji se on kreće, njegov domet i količinu štete koju će nanijeti liku u video igri.

Upravljanje razinom zvuka nalazi se pod objektom „GameManager“. Objekt sadrži skriptu „Audio Manager“ (Slika 18) koja omogućava dodavanje novih zvukova u igri te podešavanje njihove glasnoće.



Slika 18. Skripta „Audio Manager“ u inspektoru objekta „gameManager“

4 Skripte

Skripte omogućavaju upravljanje raznim objektima te sposobnost dodavanja različitih svojstava. Sve skripte u ovom projektu napisane su u jeziku C#. Slijedi njihov popis i prikaz bitnijih značajki.

- AttackCone – skripta koja provjerava je li lik ušao u „collider“ unutar kojeg će „turret“ početi pucati projektile
- AudioManager – skipta koja je zaslužna za zvuk
- Bullet – skripta koja je zaslužna za ponašanje projektila lika
- BulletTurret – skripta koja upravlja ponašanjem projektila objekta „turret-“
- CameraFollow – skripta koja upravlja ponašanjem objekta „Main Camera“ i omogućava da kamera prati lik pri određenoj brzini
- CrabController – skripta koja je zaslužna za ponašanje objekta „Crab“
- Changer – skripta koja je zaslužna za promjenu sučelja s pričom u sučelje razine
- Door – skripta koja je zaslužna za prijelaz lika na iduću razinu u slučaju da je igrač prikupio sve „orbove“ te otvorio posljednja vrata
- gameMaster – skripta koja prati „orbove“ koje je igrač prikupio na toj razini te ih prikazuje kao dio sučelja
- MainMenu – skripta koja je zaslužna za glavni izbornik na početku video igre
- Menu – skripta koja je zaslužna za izbornik koji se može aktivirat tijekom razine te za izbornik koji se aktivira u slučaju da lik izgubi život
- Parallaxing – skripta zaslužna za kretanje pozadine različitim brzinama
- PlatformMovement – skripta koja je zaslužna za kretanje platformi
- PlayerController – skripta koja upravlja svim mogućnostima igrača
- PersistentManager – skripta koja ostaje aktivna tijekom mijenjanja scena
- Spikes – skripta koja je zaslužna za postavljanje života lika na 0 kada „collider“ igrača uđe u „collider“ vode
- Story – skripta koja sudjeluje u promjeni sučelja priče i razine
- Switch – skripta koja je zaslužna za ponašanje objekta „orb“ koje lik prikuplja kako bi omogućio prijelaz na sljedeću razinu
- Tiling – skripta zaslužna za stvaranje pozadine koja se ponavlja

- Turret – skripta zaslužna za ponašanje objekta „turret“, određuje interval u kojem se projektili ispaljuju, animaciju uništavanja te metu na koju se ispaljuju projektili

Slijedi prikaz važnijih mehanizama unutar igre koji se nalaze unutar ovih skripti.

Jedan od glavnih objekata je „Player“ na kojem se nalazi skripta „PlayerController“. Unutar skripte nalaze se razne značajke pomoću kojih korisnik upravlja likom te razni načini na koje ovaj objekt utječe na ostale objekte u video igri.

```
void FixedUpdate()
{
    grounded = Physics2D.OverlapCircle(groundCheck.position, groundRadius, whatIsGround);

    anim.SetBool("Ground", grounded);

    if (grounded)
        doubleJump = false;

    anim.SetFloat("Speed", GetComponent<Rigidbody2D>().velocity.y);

    float move = Input.GetAxis("Horizontal");

    GetComponent<Rigidbody2D>().velocity = new Vector2(move * maxSpeed, GetComponent<Rigidbody2D>().velocity.y);

    anim.SetFloat("Speed", Mathf.Abs(move));

    if (move > 0 && !facingRight)
        Flip();
    else if (move < 0 && facingRight)
        Flip();

}
```

Slika 19. Kretanje lika u skripti

Unutar funkcije „FixedUpdate“ (Slika 19) koja se poziva na svaki „frame“ nalaze se razne varijable pomoću kojih se lik kreće i animira. Varijabla „grounded“ tipa bool provjerava je li „collider“ igrača u dodiru s tlom te u tom slučaju animatoru šalje parametar „Ground“ koji se koristi kako bi se zadala animacija kada lik dotiče tlo. Osim toga, varijabla onemogućava liku dvostruki skok dok je u dodiru sa tlom. Animatoru parametar „Speed“ šalje vrijednost Y osi odnosno kazuje je li lik u zraku ili nije te na taj način animator može prikazati animaciju za skok, ako je potrebna. „Move“ tipa float u sebi sadrži vrijednost „Input.GetAxis(„Horizontal“)“ koja u sebe sprema vrijednost od -1 do 1 s obzirom na to koji je gumb igrač pritisnuo. Ako je igrač pritisnuo gumb za kretanje, na „RigidBody2D“ se „velocity“ za os X postavlja računanjem varijabli „move“ i „maxSpeed“ koja je prethodno napravljena, dok vrijednost osi Y ostaje jednaka. Zaključak je da će na lika utjecati sila na X osi odnosno da će se kretati lijevo ili desno, ovisno koji je gumb igrač pritisnuo. Za kretanje parametar „Speed“ se šalje animatoru

s vrijednošću „move“ te on grafiku lika mijenja u animaciju za kretanje. U ovoj funkciji također se provjerava u kojem je smjeru lik okrenut te se on pomoću mogućnosti „Flip“ okreće s obzirom je li okrenut lijevo ili desno. Sljedeća važna funkcija „GetInputMovement“ liku u video igri omogućava sposobnost pucanja i skakanja (Slika 20).

```
void GetInputMovement()
{
    if ((grounded || !doubleJump) && Input.GetKeyDown(KeyCode.Space))
    {

        anim.SetBool("Ground", false);

        GetComponent<Rigidbody2D>().AddForce(new Vector2(0, jumpForce));

        if (!doubleJump && !grounded)
            doubleJump = true;
    }

    if (Input.GetButtonDown("Fire1") && menu.isPaused==false){

        GameObject mBullet = Instantiate(Bullet, FirePosition.position, FirePosition.rotation);

        audiomanager.playSound("ShootSound");

        mBullet.transform.parent = GameObject.Find("GameMaster").transform;

        mBullet.GetComponent<Renderer>().sortingLayerName = "Player";

        anim.SetBool("Shooting", true);

    }
    if (Input.GetButtonUp("Fire1"))
    {
        anim.SetBool("Shooting", false);
    }
}
```

Slika 20. Pucanje i skakanje u skripti

U prvoj uvjetnoj naredbi provjerava se je li igrač pritisnuo gumb za skakanje te je li vrijednost „grounded“ (varijabla koja govori da li je lik u dodiru s tlom) i varijabla „doubleJump“ (varijabla koja govori da li je lik upravo u dvostrukom skoku) istinita ili ne. Ako su uvjeti zadovoljeni, na „Rigidbody2D“, odnosno fizičke osobine lika, primjenjuje se određena sila. Sila koja će biti primijenjena određuje se varijablom „jumpForce“ koja je od prije zadana. Parametar „Ground“ također se postavlja na „false“ što animatoru govori da prikaže animaciju za skok. Sljedeće dvije naredbe omogućuju igraču da puca pritiskom lijevog gumba na računalnom mišu. Prva uvjetna naredba aktivira se kada igrač stisne gumb te ako varijabla „isPaused“ (govori je li je trenutno upaljen izbornik) nije istinita. Navedena naredba djeluje tako da stvara metak na već utvrđenom objektu „FirePosition“ koji se nalazi na liku te govori gdje će se projektil stvoriti. Skripti koja je zaslužna za zvuk također se šalje naredba koja pokreće

zvuk namijenjen zvuku pucanja. Do animatora dolazi parametar „Shooting“ s vrijednosti „istinito“ što znači da treba prikazati grafiku za pucanje. Kada igrač otpusti gumb za pucanje odnosno prestane pucati, animator ponovno prikazuje lik u osnovnom položaju. Sljedeća skripta odnosi se na ponašanje neprijatelja „Crab“ odnosno govori o njegovom uništenju.

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.tag == "Bullet")
    {
        health -= 100;

        anim.SetBool("IsDead", true);
        audiomanager.playSound("CrabDeath");

        Invoke("DestroyCrab", 0.3f);
        attackDamage = 0;
    }
}

void DestroyCrab()
{
    Destroy(gameObject);
}
```

Slika 21. Funkcija „OnTriggerEnter2D“ za objekt „Crab“

Kada „collider“ objekta „Bullet“ odnosno ispucanog metka lika dotakne „collider“ objekta „crab“ tada se njegov život postavlja na nulu. Parametar „IsDead“ tada je istinit te skripta za zvuk proizvodi zvuk prikladan varijabli „CrabDeath“. Naredba „invoke“ nakon 0.3 sekunde poziva naredbu koja uništava objekt „Crab“ te je šteta koju neprijatelj izaziva liku postavljena na 0 (Slika 21). Na isti način djeluje i uništavanje objekta „Turret“. Sljedeća skripta djeluje na sličan način. (Slika 22).

```
void OnTriggerEnter2D(Collider2D collider)
{
    if (collider.CompareTag("Player"))
    {
        player.Damage(100);
    }
}
```

Slika 22. Funkcija „OnTriggerEnter2D“ za objekt „Spike“

Ova naredba život lika postavlja se na 0 u slučaju da „collider“ lika dođe u dodir sa „colliderom“ primjerice vode. Neprijatelj „turret“ u sebi sadrži skriptu koja djeluje na sljedeći način.

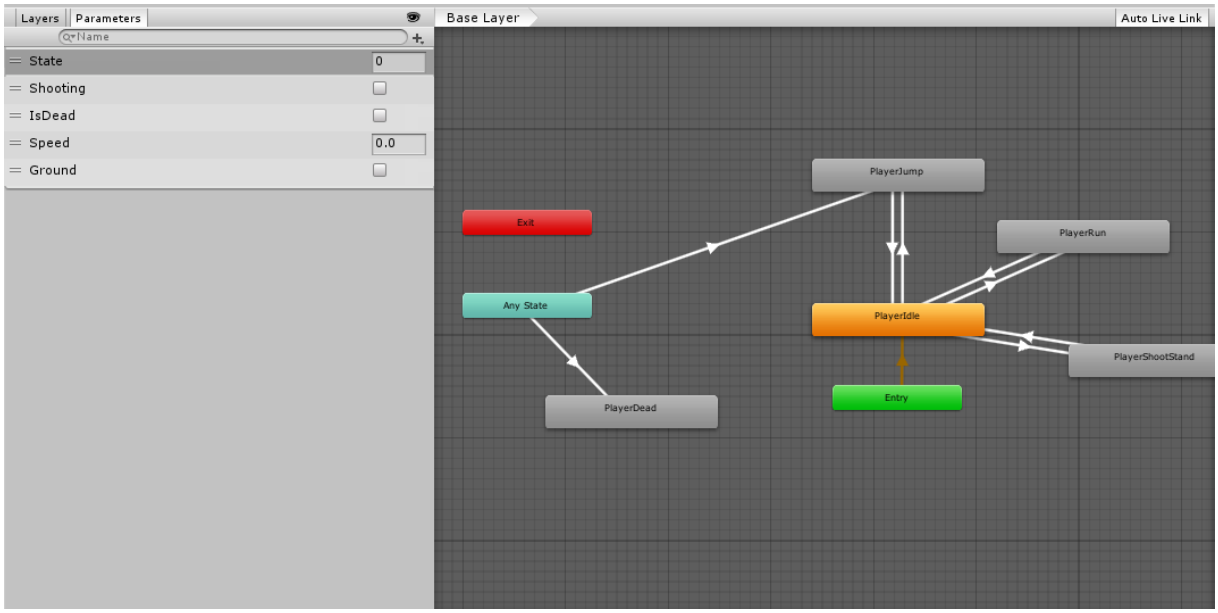
```
public void Attack(bool attackingRight)
{
    bulletTimer += Time.deltaTime;

    if (bulletTimer >= shootInterval)
    {
        Vector2 direction = target.transform.position - transform.position;

        direction.Normalize();

        if (!attackingRight)
        {
            GameObject bulletClone;
            bulletClone = Instantiate(bullet, shootPointLeft.transform.position, shootPointLeft.transform.rotation);
            audiomanager.playSound("TurretSound");
            bulletClone.GetComponent<Rigidbody2D>().velocity = direction * bulletSpeed;

            bulletTimer = 0;
        }
```

Slika 23. Funkcija „Attack“ objekta „Turret“

Funkcija „attack“ pripada objektu „Turret“ i zaslužna je za ispaljivanje projektila na lika ovisno na kojoj strani se on nalazi. Varijabli „bulletTimer“ zadaje se vrijeme u kojem je zadnji projektil ispaljen te se u prvoj uvjetnoj naredbi provjerava je li on veći od varijable „shootInterval“ koja je prije zadana. Na taj način može se odlučiti kojom će brzinom objekt ispaljivati projektile. Zatim se u varijablu „direction“ zadaje pozicija lika u odnosu na poziciju „turret-a“. Ako se lik nalazi na lijevoj strani onda se pojavljuje objekt „bulletClone“ unutar kojeg se nalazi objekt „bullet“ koji se stvara na unaprijed zadanoj poziciji „shootPointLeft“ i zadaje mu se određena vrtnja. Upravitelj zvuka reproducira zvuk prikladan pucanju „turret-a“ te se projektilu zadaje brzina, a „bulletTimer“ se ponovo vraća na vrijednost 0. Ako je lik unutar dometa pucanja funkcija „Attack“ će se ponoviti.

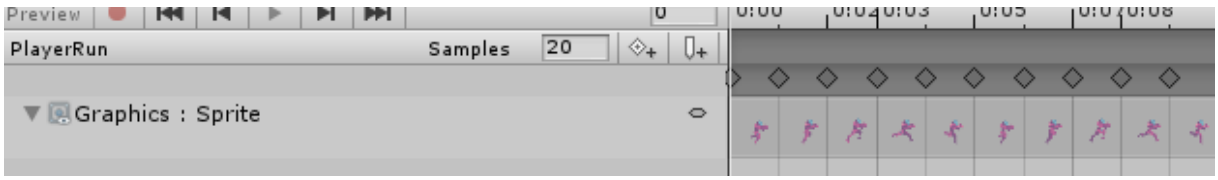
5 Animator

Animator (Slika 24), spomenut u prethodnim poglavljima, zaslužan je za animacije objekata.



Slika 24. Prikaz animatora

Slika 24 prikazuje izgled animatora. Na desnoj su strani prikazana stanja. Stanje označeno narančastom bojom odnosi se na ono stanje u kojem je model lika prikazan na početku video igre. Stanja su međusobno povezana strelicama, a promjene stanja određuju pokazatelji koje vidimo na lijevoj strani. Pokazatelji mogu biti numerički (float, int), pripadati varijabli s izborom istinito ili lažno (bool) ili okidači. Prijelazi se pokreću kada određeni pokazatelj dobije potrebnu vrijednost da promijeni trenutno stanje. Svako stanje ima svoju animaciju što je prikazano na Slici 25.

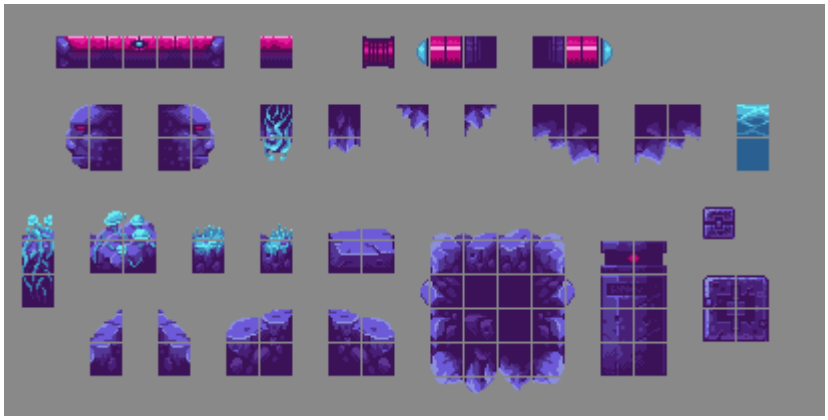


Slika 25. Prikaz animacije za objekt „Player“

Na Slici 25 vidimo kolekciju grafičkih prikaza („sprite-ova“) lika u različitim pozicijama tijekom animacije za trčanje. Svaka od tih grafika prikazana u djeliću sekunde te na taj način dobijemo dojam da se lik u video igri kreće.

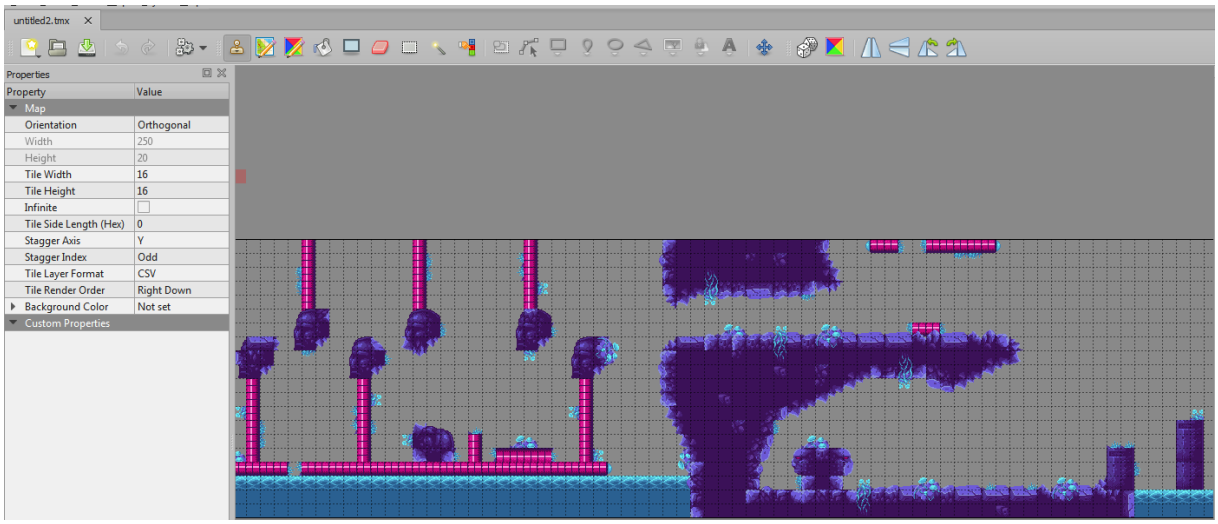
6 Izrada razina

Za izradu razina korišten je besplatni alat „Tiled“.„Tiled“ je program pomoću kojeg organiziramo dijelove („tile-ove“) tako da oni zajedno oblikuju jednu razinu. Razina se oblikuje uz pomoć oblika sadržanih unutar „tileset-a“ (Slika 26), odnosno oblika koje je moguće postavljati pri stvaranju mape.



Slika 26. „Tileset“ elementi

Nakon stvaranja mape pomoću alata „Tiled2Unity“ mapu pretvaramo u oblik datoteke koji je moguće koristiti u Unityu. Ovakvi su alati stvoreni u okrilju zajednice korisnika okruženja Unity, koji međusobno pridonose poboljšanju Unitya. Prikaz izrade jedne razine vidljiv je na Slici 27.



Slika 27. Program „Tiled“

Autor grafike korištene u ovom projektu „Warped Caves“ je Luis Zuno.

7 Zaključak

Izrada velikih naslova u svijetu video igara danas odnosi milijune dolara utrošenih na proizvodnju i marketing. Nasuprot tome, razvijači nezavisnih igara takve video igre izrađuju samostalno ili u skupinama od nekoliko pojedinaca. S obzirom na ogromnu konkurenciju većina nezavisnih razvijača odlučuje se za inovativne ideje u izradi video igara.

Razvijači igara primorani su koristiti jedno od besplatnih okruženja za izradu video igara poput Unity-a ili Unreal Engine-a. U suprotnom moraju samostalno napraviti okruženje što je vrlo zahtjevan i skup proces.

U izradi ove igre korišteno je okruženje Unity te programski jezik C#. Unity je besplatno okruženje koje sadrži mnoge alate korisne za razvoj video igre. Okruženje Unity iskazalo po svojoj jednostavnosti i velikom broju edukativnih sadržaja na internetu. Zajednica korisnika koji su spremni pomoći, velika je pomoć pri upoznavanju sa ovim okruženjem. Prednost Unitya je također zajednica korisnika koji prave alate i dodatke za Unity koji se mogu pronaći na internetu te u „Unity Store“ dijelu okruženja, bez kojih ovaj projekt ne bi bilo moguće ostvariti.

Nakon izrade video igre može se zaključiti da razvijači igara moraju uložiti veliku količinu vremena i strpljenja kako bi njihova igra bila blizu razine kakvu današnje igre zadovoljavaju. Osim toga, izradom video igre može se izraziti na različite načine, a za neke dijelove izrade poput izrade pojedinih razina ili izgleda mape potrebno je uložiti puno vremena i kreativnosti.

Iz procesa izgradnje igre može se zaključiti da je to dugotrajan i složen zadatak. Potrebno je puno ideja i različitih pogleda na isti problem kako bi se došlo do nečeg što može konkurirati trenutnom tržištu. Na sreću, igre izlaze godinama te alati i ideje pomoću kojih se može izgraditi vlastita igru i vizija je ovih dana lagano te je za to potrebno samo par programa.

Literatura

Matthieu Oger (2016) https://pixelnest.io/tutorials/2d-game-unity/parallax-scrolling/ datum pristupa 4.9.2019

Luis Zuno (2018) https://ansimuz.itch.io/warped-caves datum pristupa 4.9.2019

Tabitha Baker (2018) https://www.indiegamewebsite.com/2018/10/19/the-complete-history-of-indie-games/ datum pristupa 4.9.2019

Greg Lukosek (2016) Learning C# by Developing Games with Unity 5.x

Unity Documentation https://docs.unity3d.com/ScriptReference/Animator.html datum pristupa 2.9.2019

Unity Documentation https://docs.unity3d.com/ScriptReference/Collider.html datum pristupa 2.9.2019

Unity Documentation https://docs.unity3d.com/Manual/CreatingScenes.html datum pristupa 2.9.2019

Sean Barton (2014) https://seanba.com/introtiled2unity.html datum pristupa 4.9.2019

Unity Documentation https://unity3d.com/what-is-a-game-engine datum pristupa 4.9.2019

Popis slika

Slika 1. Prikaz sučelja iz igre „CupHead“ 2
Slika 2. Prikaz sučelja iz video igre „Hollow Knight“ 2
Slika 3. SWOT analiza 3
Slika 6. Izbornik za razine 5
Slika 7. Scena sa pričom 5
Slika 8. Pregled objekata na razini 4 6
Slika 9. Pregled izbornika za stanku unutar razina 6
Slika 10. Pregled izbornika za podešavanje zvuka 7
Slika 11. Pregled inspektora za objekt „Player“ 8
Slika 12. „Collider“ od objekta „Player“ 9
Slika 13. Inspektor za objekt „ExitDoor“ 9
Slika 14. Inspektor za objekt „Orb“ 10
Slika 15. Inspektor i hijerarhija objekta „Turret“ 10
Slika 16. Prikaz „collider-a“ od objekta „ShootPointLeft“ i „ShootPointRight“ 11
Slika 17. Inspektor za objekt „Crab“ 11
Slika 18. Skripta „Audio Manager“ u inspektoru objekta „gameManager“ 12
Slika 19. Kretanje lika u skripti 14
Slika 20. Pucanje i skakanje u skripti 15
Slika 21. Funkcija „OnTriggerEnter2D“ za objekt „Crab“ 16
Slika 22. Funckija „OnTriggerEnter2D“ za objekt „Spike“ 16
Slika 23. Funckija „Attack“ od objekta „Turret“ 17
Slika 24. Prikaz animatora 18
Slika 25. Prikaz animacije za objekt „Player“ 18
Slika 26. „Tileset“ elementi 19
Slika 27. Program „Tiled“ 19

Sažetak

Ovaj rad obuhvaća razvoj video igre u okruženju Unity te pobliže opisuje način izrade video igre žanra „platformer“. Sadrži pregled korištenih mogućnosti i alata u Unityu te dodataka koje je moguće koristiti uz ovaj alat.

Video igra se sastoji od razina i konceptualno različitih prepreka koje igrač mora savladati kako bi uspješno napredovao do sljedeće razine.

Opisana je cjelokupna provedba od prazne scene do gotove video igre što uključuje pregled svih objekata i mogućnosti korištenih za izgradnju ove video igre.

Ključne riječi: Unity, platformer, igra, programiranje

Abstract

This thesis depicts developing a video game using Unity and a closer look at the process of creating a platformer type game. It includes a look at all the functionalities and tools included in Unity, and all the extensions we can use alongside Unity.

The game consists of different levels and conceptually different obstacles which the player must overcome in order to complete the level.

Also included is the overall implementation starting from an empty scene to a finished game. This covers a look at all the scenes, objects and components used to create this game.

Keywords: Unity, platformer, game, programming