Računarska gimnazija

Beograd, Knez Mihailova 6

Maturski rad iz porgramiranja i

programskih jezika

2D RPG platformer u Unity-u

Escape the dungeon

Mentor:

Vladimir Kuzmanović

Učenik:

Lazar Bošković IV2

Beograd, 2022

Sadržaj

1. Uvod

1.1. Šta je game development

Razvoj igara (Game Development) je umetnost stvaranja igara i opisuje dizajn, razvoj i izdavanje igre. To može uključivati generisanje koncepta, dizajn, izgradnju, testiranje i izdavanje. Dok kreirate igru, važno je razmišljati o mehanici igre, nagradama, angažovanju igrača i dizajnu nivoa.

Da biste se uključili u proces razvoja igre, ne morate pisati kod. Umetnici mogu da kreiraju i dizajniraju sredstva, dok se programer može fokusirati na programiranje objekta za prikazivanje života (Health Bar). Tester se može uključiti da vidi da igra radi kako se očekuje.

Da bi se rešili problemi koje su imali okviri za igre, razvijeni su alati poput libGDKS i OpenGL. Oni su pomogli da razvoj igara bude mnogo brži i lakši, pružajući mnogo unapred napravljenih funkcija i karakteristika. Međutim, i dalje je bilo teško ući u

industriju ili razumeti okvir za nekoga ko dolazi iz neprogramerske pozadine, što je uobičajen slučaj na sceni razvoja igara.

Tada su razvijeni motori za igre poput Construct, Game Maker, Unity i Unreal. Generalno, mašine imaju sve što je imao okvir, ali sa više prijateljskim pristupom korišćenjem grafičkog korisničkog interfejsa (GUI) i pomaganjem u grafičkom razvoju igre.

1.2. Game Engines

Mašine za igre (Game Engines) predstavljaju program ili okruženje za razvoj softvera koji su ljudi prvobitno koristili za pravljenje i razvoj video igara. Danas se oni koriste i za vizueluzaciju, kolaboraciju itd. Mogućnosti ovih mašina su ogromne a neke uključuju alate za animaciju, veštačku inteligenciju, fiziku, koliziju, alate za zvuk i više.

Ukratko, ove mašine postavljaju okvire koji korisniku omogućavaju da kreira nešto lakše nego da to mora da pravi od nule. Okviri variraju od mašine do mašine, ali oni ugalvnom uključuju 2D i 3D mehanizme za renderovanje.

Ove softvere su prvobitno kreirali programeri igara u svrhu njihovog bržeg i lakšeg razvoja. Ipak, danas se ovi alati za renderovanje koriste i u drugim industrijama.

Neke od najpoznatijih motora za igru su: Unity (razvijen od strane Unity Technologies-a), Unreal Engine (razvijen od strane Epic Games-a), CryENGINE i ostali.

1.2. Šta je Unity

Unity je višeplatformska mašina za igre. Predstavlja platformu za razvoj 2D i 3D aplikacija i igara za računare, konzole, mobilne uređaje i web sajtove, sa mogućnošću izvršavanja na preko 20 podržanih platformi. Jezgro Unity-a napisano je u C/C++ programskom jeziku, dok je Unity UI Editor napisan u jeziku C#.

Za pristup najnižem sloju, odnosno jezgru i funkcijama Unity-a, dostupan je API za korišćenje u .NET Framework-u, i jezicima C#, Boo, ali i JavaScript. Za pisanje koda

se standardno koristi Monodevelop, ali je moguće korišćenje bilo kog drugog okruženja, npr. Microsoft Visual Studio.

Logo Unity Technologies-a

1.3. Istorijat Unity-a

Unity je osnovan u Kopenhagenu od strane troje ljudi- Nicholas Francis-a, Joachim Ante-a, i David Helgason-a. U maju 2002. godine Francis je na OpenGL forumu objavio poziv za saradnike na kompajleru šejdera otvorenog koda za nižu populaciju programera igara zasnovanih na Mac-u poput njega. Tadašnji srednjoškolac iz Berlina, Joakim Ante se odzavao pozivu.

On je svojim idejama za pozadinsku arhitekturu dopunio Francisov smisao za grafiku i igrivost. Jedno vreme su radili zajedno samo povremeno ali posle sastanka uživo su odlučili da se potpuno posvete radu zajedno. Na početku su kodirali u Helgasonovom stanu dok je on bio vani. Tu su došli na ideju da pokrenu sopstveni studio za igre.

Helgason i Francis su se znali iz srednje škole, gde su zajedno pokušavali pravljenje veb-sajtova. Helgason je napustio Univerzitet da bi radio kao veb programmer. Tek kasnije se i on potpuno priključio zajedničkom radu. On je kasnije postao generalni diretkor.

Zbog velike raznolikosti ideje koju su imali, uspeli su da naprave motor koji je uspeo da rešava razne problem. Hteli su da reklamiraju dotadašnji rad hit igrom koja bi pokazala pun potencijal njihovog motora.

1.4. C# u Unity-u

Skripte su element koje priključujemo našem objektu u igrici (GameObject) da bi ih Unity mogao pozvati. Skripte su napisane na posebnom jeziku koji Unity može da razume. I kroz ovaj jezik možemo razgovarati sa mašinom (engine-om) i dati mu uputstva.

C# je jezik koji krosti Unity Engine. Svi jezici sa kojima Unity radi su objektno orijentisani skriptni jezici. Kao i svaki drugi jezik, skriptni jezici imaju sintaksu a primarni delovi se nazivaju promenljive, funkcije i klase.

Sve verzije Unity-a koju se izašle pre verzije 2017.3, uključujući i nju, imaju MoonDevelop kao glavni program za editovanje teksta. On nam je mogao pomoći da napišemo naš kod, obavesti nas ako pišemo pogrešan kod i pravimo greške i omogućava korišćenje prečica. Počevši od verzije 2018.1, kao program za editovanje teksta dostupni su i Visual Studio, Notepad ili Sublime tekst.

1.5. Promenljive, funkcije i klase

Tokom pisanja koda u okviru skripte, programeri najčešće koriste promeljive, funkcije i klase.

1.5.1. Promenljive

„Promenljive sadrže vrednosti i reference na objekte. Oni su kao kutija u kojoj se nalazi nešto što možemo da koristimo. Promenljive počinju malim slovom.“

Promenljive mogu biti javne (public) ili privatne (private), i oba pružaju različite mogućnosti. Privatne promenljive mogu biti kontrolisane samo unutar koda i one olakšavaju kodiranje i rešavanje problema, dok javne promenljive mogu praviti problem prilikom traženja greške, ali omogućavaju međusobnu komunikaciju kao i vidljivost iz samog editora.

1.5.2. Funkcije

“Funkcije su kolekcije koda koje upoređuju i manipulišu ovim promenljivim. Funkcije počinju velikim slovom. Organizujemo kod u funkcije tako da se mogu lako ponovo koristiti više puta u različitim delovima programa.”

Unity sadrži puno funkcija koje se pokreću automatski pri pokretanju vašeg koda. To su Awake, Start, Update, FixedUpdate i LateUpdate.

1.5.3. Klase

"Klase su način da se strukturira kod za omotavanje kolekcija promenljivih i funkcija zajedno kako bi se napravio šablon koji definiše svojstva objekta."

Upoređivanje ovih objekata i njihovih stanja i vrednosti predstavlja skriptovanje. Ta logika je zasnovana na određivanju ishoda ili rešenja.

Priemer skripte iz mog projekta:

// PortalS je skirpta/klasa

```
public class PortalS : Collidable
{
    public string[] sceneNames; // sceneName je promenljiva

    //OnCollide je funkcija
    protected override void OnCollide(Collider2D coll){
        if(coll.name == "Player"){
            //teleport the player
            GameManager.instance.SaveState();
            string sceneName = sceneNames[Random.Range(0, sceneNames.Length)];
            UnityEngine.SceneManagement.SceneManager.LoadScene(sceneName);
        }
    }
}
```

2. O igri

2.1. Opis igre

Igra koju sam napravio predstavlja 2D Indie RPG (role-playing game) platformer.

U igri kontrolišete igrača koji se kreće po napravljenim nivoima. Na samom ekranu pored dizanja nivoa u donjem levom uglu nalazi se i dugme za HUD – interfejs i objekat za prikazivanje života (Health Bar). U HUD – interfejsu se nalaze informacije

o trenutnom oružju i potrebnim novcem za unapređivanje do boljeg, izgled (skin) karaktera kao i mogućnost promene istog i informacije o životu, nivou i parama.

HUD - interfejs

Postoji i sistem za borbu sa neprijateljima koji počnu da vas jure ako im se približite, prestanu ako izađete iz njihovog dometa i bacaju poene za podizanje nivoa kada ih ubijete što služi za podizanje nivoa(level up). Pored živih elemenata igre postoji i fontana za vraćanje života, portali koji vode između nivoa i kutije koje možete slomiti oružjem.

2.2. Način igranja

Igrača možete pokrenuti pomoću dugmića w/a/s/d ili strelica i možete zamahnuti oružjem na protivnike pritiskom na dugme razmak (space). Takođe postoji opcija pristupavnja HUD – interfejsu levim klikom miša.

2.3. Moguća poboljšanja

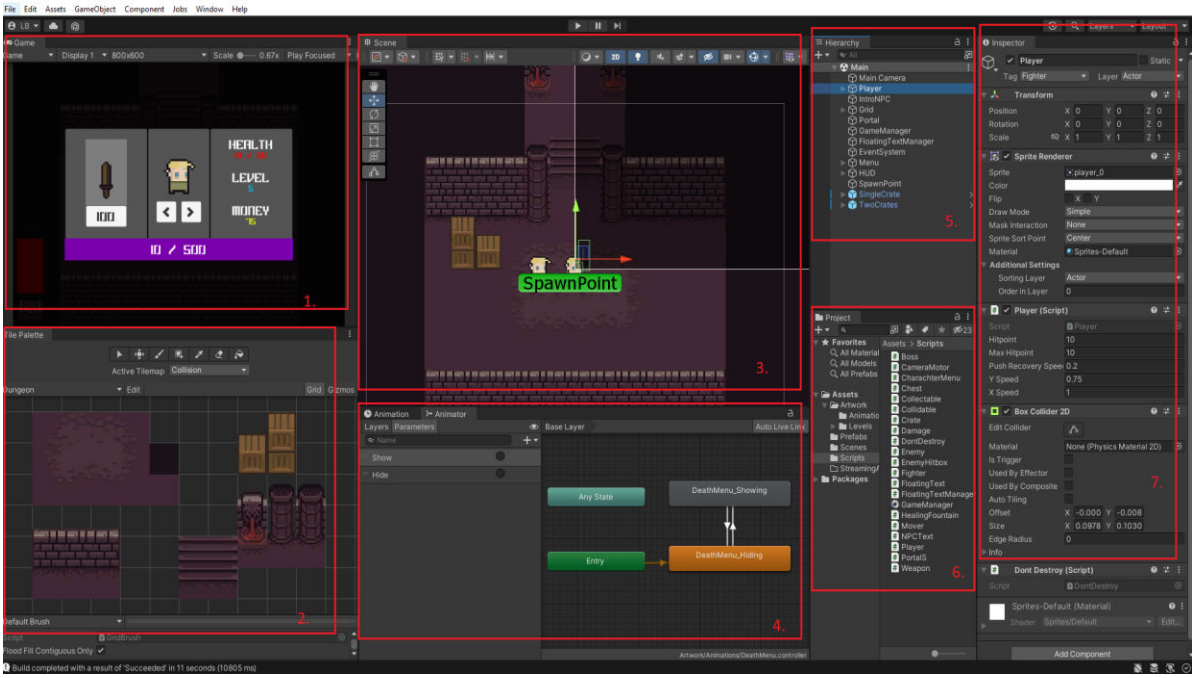
Igrica radi sasvim prihvatljivo ali dužim igranjem sam često nailazio na sitne greške, kao što su: greška stvaranja igrača van nivoa, raspored tekstura (neke teksture su bile preko nekih objekata većeg statusa). Pored rešavanja grešaka dodao bih i glavni meni u kojem bi mogli da birate da li hoćete da igrate ili izađete iz igre, odabir skina pre početka umesto u toku igre i slično. Takođe igra trenutno čuva podatke pri izlasku iz nje ili njenim prelaskom. Bilo je zamišljeno da to radi pri izlaženju jer ima fnkciju za čuvanje podataka, ali pri završetku bih dodao da restartuje sve vrednosti na nulu.

3. INTERFEJS

3.1. InGame izgled

1.Karaktera kojeg kontrolišemo 2. HealthBar i dugme za HUD - interfejs

3.2. Unity Engine interfejs



1. Prozor igre 2. Paleta 3. Scena 4. Animacije I animator 5. Hierarhija 6. Folder projekta 7. Inspektor

3.2.1. Game Scene prozor

Prozor igre (Game Scene) predstavlja prozor u kojem testiramo i igramo našu igru ili porjekat.

3.2.2. Paleta

Paleta (Tile Palette) predstavlja komponentu koja sadrži teksture koje se koriste tokom izrade nivoa.

3.2.3. Scena

Scena (Scene View) predstavlja prozor u kojem pravimo naše nivoe dodavanjem objekata.

3.2.4. Animacije i animator

Prozor za animacije služi pri pravljenu animiranih pokreta koje koristimo u toku igre kao što su zamah oružja, animacija sveće, animirani glavni neprijatelji (Boss) i slično. Animator služi sa nameštanje i realizovanje tih animacija na scenu.

3.2.5. Hierarhija

Hierarhija sadrži svaki objekat (GameObject) u trenutnoj sceni. Objekti mogu da postanu deca drugih objekata prevlačenjem jednih u druge (Parenting).

3.2.6. Folder projekta

U ovom prozoru se nalaze svi folderi i dokumenti našeg projekta. To su sve skripte, teksture, scene itd.

3.2.7. Inspektor

Inspektor (Inspector) prozor prikazuje sve podatke odabranog objekta i daje nam mogućnost za menjanje istih, kao što su dimenzije, pozicije, izgled, fizikom i sličnim elementima.

4. Izrada igre

4.1. Vector3

Tokom izrade igre sam koristio vektore. U Unity-u 3D vektori se opisuju Dekartovim koordinatama. To su veličine koje imaju pravac, smer i intezitet. Mogu se koristiti za izračunavanje razdaljine, uglova između objekata i slično. Unity ima strukturu Vector3 kao prezentaciju vektora i tačaka.

4.2. Kamera

Kamera je važan deo 2D i 3D igara. One se mogu postaviti bilo gde u prostoru, fiksiraju za jedan objekat i animiraju. Glavna kamera (Main Camera) je tu u svakom trenutku i ona prikazuje ono što igrač vidi, dok postoji opcija za dodavanje dodatnih kamera.

Postoje dva režima renderovanja u 2D i 3D i to su Perspective i Orthographic režimi. Perspektivni režim je standarni. Ortografski režim se uglavnom koristi za razvo 2D igara ili 2D elemeata 3D igara.

U radu dodao sam mogućnost glavnoj kameri da prati igrača i to na sledeći način:

```
 private Transform lookAt;
public float boundX = 0.15f;
public float boundY = 0.05f;
```

```
private void Start(){
    lookAt = GameObject.Find("Player").transform;
}
private void LateUpdate(){
    // Moramo da pomerimo kameru ako igrač izađe iz okvira

    Vector3 delta = Vector3.zero;

    // Proverimo da li smo u granciama na X osi
    float deltaX = lookAt.position.x - transform.position.x;
    if(deltaX > boundX || deltaX < -boundX)
    {
        if(transform.position.x < lookAt.position.x)
        {
            delta.x = deltaX - boundX;
        }
        else
        {
            delta.x = deltaX + boundX;
        }
    }

    // Proverimo da li smo u granciama na X osi
    float deltaY = lookAt.position.y - transform.position.y;
    if(deltaY > boundY || deltaY < -boundY)
    {
        if(transform.position.y < lookAt.position.y)
        {
            delta.y = deltaY - boundY;
        }
        else
        {
            delta.y = deltaY + boundY;
        }
    }

    transform.position += new Vector3(delta.x , delta.y, 0);
}
```

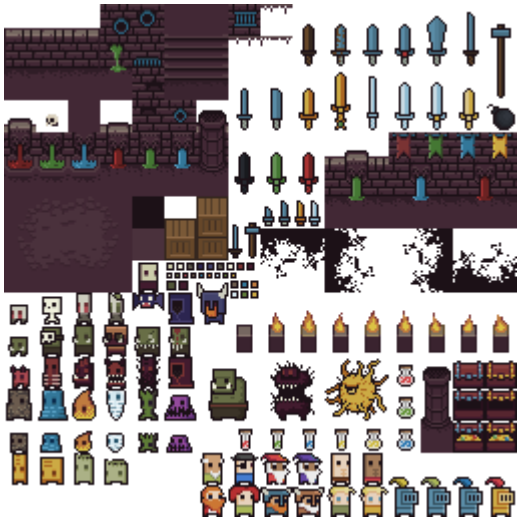
4.3. Teksture

Teksture predstavljaju bitmap-u slike koje možemo postaviti na naše objekte da bi im se promenila vizuelizacija.

Sve teksture mog projekta kao što su pozadina, pod, karakteri ,neprijatelji, oružja i ostalo sam iskoristio iz već postojeće kolekcije tekstura pod nazivom 16 x 16 Dungeon Tileset.

Cela slika je napravljenja tako da se svaka tekstura može iseći u 16 x 16 veličini i tako lakše koristiti za kasniju izradu. Da bi se pojedinačne teksture mogle iseći iz slike koristimo Sprite Editor. U njemu mežemo kvadratnim ivicama označiti željenu teksturu i pritiskom na dugme primeni (Apply) ona će se sačuvati u naš folder tekstura (Artwork).

Takodje je potrebno u inspektoru dok nam je željena slika sa teksturama selektovana da Sprite Mod postavimo na višestruko (Multiple) da bi uopšte mogli seći sliku na manje.



Teksture korišćene pri izradi igre

Nakon čuvanja novonastalih tekstura možemo ih premestiti u Paletu (Tile Palatte) iz koje ih kasnije možemo primenjivati na naše nivoe. Postoje posebni nivoi palate koji razlikuju pod, zidove, i neprobojne delove nivoa. Ovo je urađeno jer ponekad na

jednom polju može biti više tekstura jedne preko drugih i da bi se to preklapanje moglo videti.

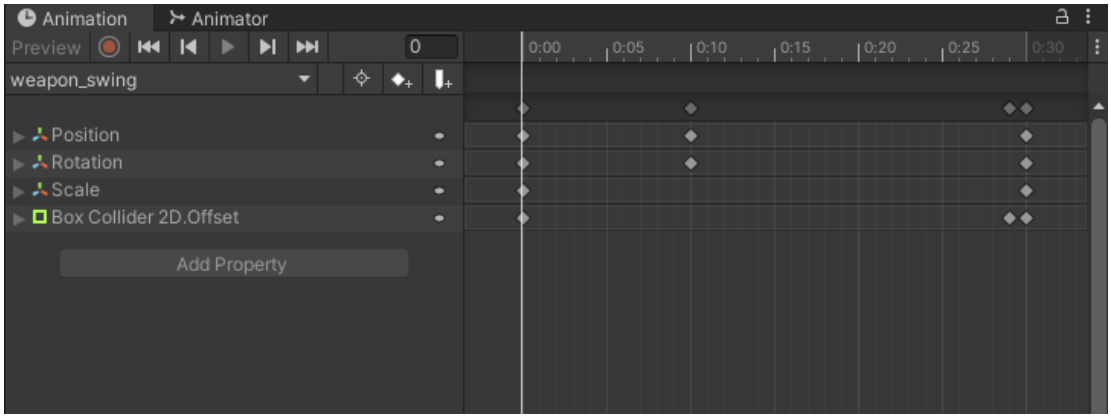
4.4. Layers

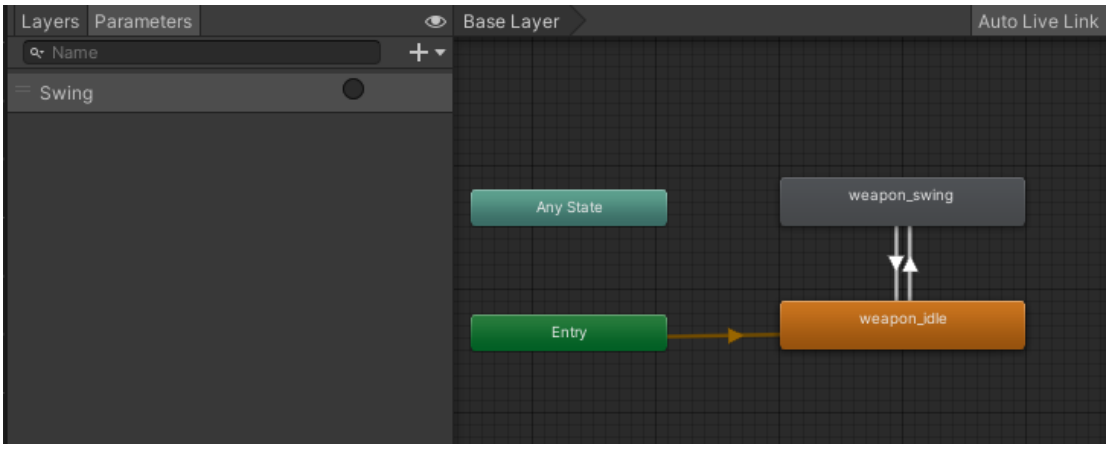
Slojevi (Layers) služe za prikazivanje funkcionalnosti zajedničke raličitim objektima. Oni ukuzuju na to koji je sloj skriven, koji nije, koji je iscrtan, kroz koji se nemože proći itd. Takođe postoji deo slojeva za sortiranje koji služi da bi se znao raspored objekata na nivou. Npr. Igrač treba da hoda po podu ali ne sme po zidovima.

Moj rad sadrži dva sloja: Actor i Blocking i pet slojeva za sortiranje: Floor, Wall, Interectable, Acotr I Weapon.

4.5. Animacije

Animacije objekata igre su uspešno izvedene uz pomoć već pomenutog animatora i prozora za animacije. Animacije pravimo tako što snimamo pokrete koje hoćemo da zabeležimo kao animaciju. Animacije imaju i opciju neprestanog ponavljanja (Loop Time). U mom slučaju postoje četri animacije. To su baklja, zamah oružja, animirani glavni neprijatelji, i pokretni HUD – interfejs.



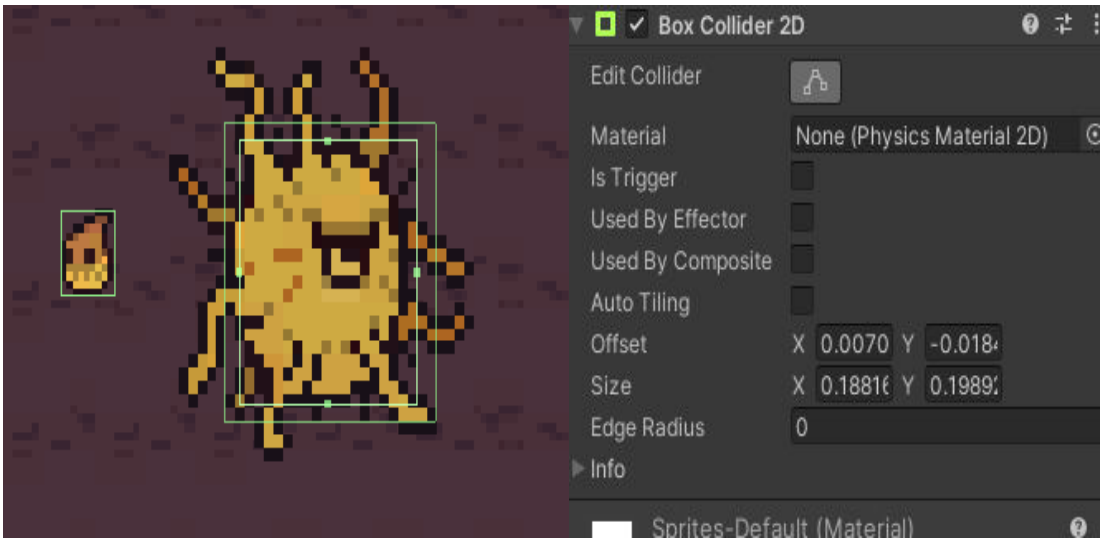


Prikaz stvaranja animacije I animatora za pokret oružja

4.6. Provera sudara objekata

Pri detekciji dodira objekata Unity pruža mogućnost korišćenja Collider-a, komponente koje stvara nevidljivu mrežu oko objekta. Korisnik može da promeni dimenzije i poziciju ovih mreža po svom izboru u inspektoru pod stavkom izmeni collider (Edit Collider).

Korišćenjem Collider2D komponente na objektima i kodom uspeo sam da postignem sudar igrača sa elementima u igri kao i zidom. Pored ovog elementa, objektima sam dodaoi sloj blocking koji služi baš za fiziku sudara.



Pirkaz Collider2D na neprijatelju i u inspektoru

4.7. Pokretanje igrača

Pokretanje objekata je olakšano u Unity-u uz pomoć komponenata Horizontal i Vertical i metode GetAxisRaw. Ovim skraćujemo sam kod jer iz ovih komponenata program sam prepoznaje u kom smeru se trebamo kretati.

Pokretanje igrača sam postigao kroz dve skripte, tako što jedna nasleđuje podatke iz druge.

Klasa Mover:

```
protected virtual void UpdateMotor(Vector3 input){
        moveDelta = new Vector3(input.x * xSpeed, input.y * ySpeed, 0);

        if(moveDelta.x > 0)
        {
            transform.localScale = originalSize;
        }
        else if(moveDelta.x < 0)
        {
            transform.localScale = new Vector3(originalSize.x * -1,
originalSize.y, originalSize.z);
        }

        //Proverava da li možemo da idemo u tom pravcu, neda prolaz kroz kutije
        hit = Physics2D.BoxCast(transform.position, boxCollider.size, 0, new
Vector2(0, moveDelta.y), Mathf.Abs(moveDelta.y * Time.deltaTime),
LayerMask.GetMask("Actor","Blocking"));
        if(hit.collider == null)
        {
            // Pokretanje igrača
            transform.Translate(0, moveDelta.y * Time.deltaTime, 0);
        }

        hit = Physics2D.BoxCast(transform.position, boxCollider.size, 0, new
Vector2(moveDelta.x, 0), Mathf.Abs(moveDelta.x * Time.deltaTime),
LayerMask.GetMask("Actor","Blocking"));
        if(hit.collider == null)
```

```
        {
            transform.Translate(moveDelta.x * Time.deltaTime, 0, 0);
        }
    }
```

Klasa Player:

```
private void FixedUpdate(){
        float x = Input.GetAxisRaw("Horizontal");
        float y = Input.GetAxisRaw("Vertical");

        if(isAlive)
            UpdateMotor(new Vector3(x, y, 0));
    }
```

5. Razni objekti u igri

5.1. Kovčezi

Kovčezi su objekti u igri koji vam dodaju novac kada dođete u dodir sa njima. Oni su važan deo igre jer je novac potreban kako bi dobili bolje oružje koje vam omogućava prelazak igre. Svaki kovčeg sadrži odredjenu količinu novca koju možemo dodati u isnpektoru.

```
public Sprite emptyChest;
    public int markeAmount = 5;

    protected override void OnCollect(){
        if(!collected){
            collected = true;
            GetComponent<SpriteRenderer>().sprite = emptyChest;
            GameManager.instance.marke += markeAmount;
            GameManager.instance.ShowText("+"    +    markeAmount    +    "
marki!", 25, Color.yellow , transform.position, Vector3.up * 25, 1.5f);
        }
    }
```

Kovčeg

5.2. Fontane za vraćanje života

Fontana može igraču vratiti izgubljeni život.

```
public int healingAmount = 1;
```

Fontana za vraćanje života

```
    public float healCooldown = 1.0f;
    private float lastHeal;

    protected override void OnCollide(Collider2D coll){
        if(coll.name != "Player")
            return;
        if(Time.time - lastHeal > healCooldown){
            lastHeal = Time.time;
            GameManager.instance.player.Heal(healingAmount);
        }
    }
```

5.3. Kutije

Kutije su objekti koji predstavljaju prepreku igraču. On ih može slomiti svojim oružjem.

Kutije

```
protected override void Death(){
        Destroy(gameObject);


    }
```

5.4. Neprijatelji

Neprijatelji su podeljeni u dve kategorije: male neprijatelje i glavne neprijatelje (Boss). Mali neprijatelji imaju skriptu Enemy dok glavni neprijatelji imaju skriptu Boss koja povalči podtke iz klase Enemy.

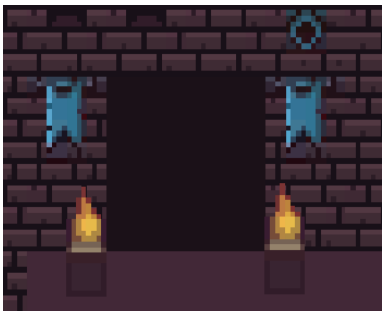
Svi neprijatelji imaju sposobnost da vas jure kada uđete u njihov domet i to je regulisano u skripti Enemy.

Pir dodiru sa igračem skidaju mu život ali pri njihovm ubianju bacaju poene koji služe za podizanje nivoa igrača i povećavaju maksimalan život igrača za jedan.

5.5. Portali

Portali postoje na raznim delovima mape i oni vas šalju na sledeći nivo ili nazad na početak igre.

```
public string[] sceneNames;

    protected override void OnCollide(Collider2D coll){
        if(coll.name == "Player"){
            GameManager.instance.SaveState();
            string sceneName = sceneNames[Random.Range(0,
sceneNames.Length)];

UnityEngine.SceneManagement.SceneManager.LoadScene(sceneName);
```



Portal

5.6. Izgled karaktera i oružja

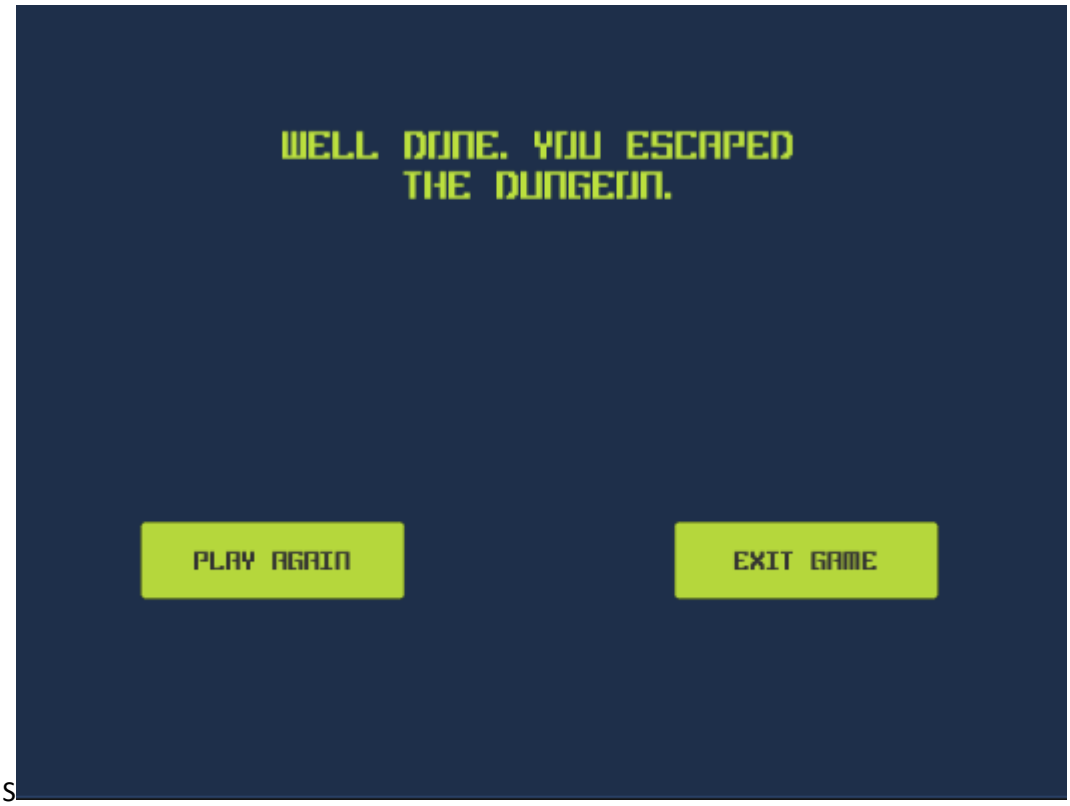
```
public void SwapSprite(int skinId){
        spriteRenderer.sprite =
GameManager.instance.playerSprites[skinId];
    }

public bool TryUpgradeWeapon(){
        if(weaponPrices.Count <= weapon.weaponLevel)
            return false;
        if(marke >= weaponPrices[weapon.weaponLevel]){
            marke -= weaponPrices[weapon.weaponLevel];
            weapon.UpgradeWeapon();
            return true;
        }
        return false;
    }

public void UpgradeWeapon(){
        weaponLevel++;
        spriteRenderer.sprite =
GameManager.instance.weaponSprites[weaponLevel];
    }
```

Različiti izgledi karaktera

Različiti izgledi oružja

6. Ekran na kraju igre

Završni ekran se prikazuje kada se prođe kroz poslednji portal na kraju trećeg nivoa. Sastoji se od teksta i dva dugmeta. Oni daju izbor između igranja ponovo i izlaska iz igre.



S

Završni ekran

Exit Game dugme pokreće metodu unapred definisanu u Unity-u Application.Quit() koja izgasi vašu aplikaciju.

7. Zaključak

Unity Game Engine je veoma funkcionalna aplikacija koja pruža veoma puno različitih mogućnosti u prostoru razvoja 2D i 3D igara. Veoma je ravnopravna i prema dobrim programerima i prema početnicima koji tek ulaze u svet pravljenja igara. Interfejs Engine-a je veoma dorbo urađen i funkcionalan za izvršavanje više stvari odjednom. Kao neko ko ranije nije korsitio aplikaciju snašao sam se veoma dobro.

Rad je podeljen na nekoliko celina. Prva obuhvata opis razvoja igara i istoriju Unity-a, kao i njegovu konekciju sa jezikom C#. Sledeća sadrži detaljan opis igre, način za igranje kao i potencijalna poboljšanja i greške na koje sam nailazio tokom izrade. Finalna sekcija obuhvata opis fizike nekih delova igre kao i opis delova Unity-a koje sam koristio.

Ideju sam dobio iz raznih izvora, od kojih je jedna da i sam igram slične igre i želja za učenjem izrade samih, podstakla je izradu ovog projekta. Izrada ove igre dosta je pomogla mom programerskom razvoju jer sam utvrdio stvario koje sam znao a one koje nisam sam uspeo da naučim.

8. Literatura

- https://www.perforce.com/resources/vcs/game-engine-overview
- https://unity.com/how-to/learning-c-sharp-unity-beginners#:~:text=The%20language%20that's%20used%20in,variables%2C%20functions%2C%20and%20classes.
- https://techcrunch.com/2019/10/17/how-unity-built-the-worlds-most-popular-game-engine/#:~:text=Unity%20was%20founded%20in%20Copenhagen,based%20game%20developers%20like%20himself.
- https://0x72.itch.io/16x16-dungeon-tileset