Pustolovna mozgalica iz ptičije perspektive

Prga, Josip

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Josip Prga

PUSTOLOVNA MOZGALICA IZ PTIČJE PERSPEKTIVE

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Josip Prga

Matični broj: 0016140108

Studij: Informacijski sustavi

PUSTOLOVNA MOZGALICA IZ PTIČJE PERSPEKTIVE

ZAVRŠNI RAD

Mentor/Mentorica:

Doc. dr. sc. Robert Kudelić

Varaždin, kolovoz 2021.

Josip Prga

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

„Pustolovna mozgalica iz ptičje perspektive“ je video igra napravljena u Unity alatu koristeći C# skripte. Koristi se Unity zbog lakoće pristupa i rada alata. Unutar Unity-a je korišteno Unity2D radno sučelje. U projektu smo se fokusirali na rad s C# skriptama unutar Unity projekta. Istraživali smo međusobno utjecanje Unity objekta unutar C# skripti te njihovo implementiranje u same razine. Video igra se sastoji od 5 razina. Prva razina je uvodna razina u kojoj se igrač upoznaje s kretnjama glavnog lika. Druga razina je izborna razina u kojoj možemo odabrati između 3 druge razine. Ostatak razina su mozgalice koje je potrebno riješiti kako bi završili i time prešli video igru.

Ključne riječi: Unity, C#, Unity2D, video igre, ptičja perspektiva, mozgalica, skriptno programiranje

Sadržaj

1. Uvod

„Pustolovna mozgalica iz ptičje perspektive“ je 2D video igra izrađena u alatu Unity. Sami alat koristi C# skripte da bi upravljao dijelove igre. Video igre su postale veoma značajne u današnjem svijetu. Samom zaradom pridonose više od glazbe i filmova zajedno. Mnogi smatraju kako su video igre kombinacija svih umjetnosti u tehnologiji. Moja motivacija za odabir ove teme je ulazak u svijet kreacije video igara. Video igre danas su popularnije nego ikada te smatram da će biti veliki dio budućnosti zabave.

Ovaj projekt se sastoji od 5 razina i početnog izbornika. Početni izbornik se sastoji od dvije tipke: „Start“ i „Quit“. Klikom na „Quit“ napuštamo igru, a klikom na „Start“ učitava se prva razina. Prva razina predstavlja način da se igrač upozna s kretnjom igrača i interakcijom s okolinom. Nakon prolaska kroz prva vrata učitava se druga razina koja služi kao izbornik za 3 mozgalice. Igrač mora riješiti sve tri mozgalice počevši od prve kako bi završio igru.

2. Korišteni alati

2.1. Unity

Unity je cross-platform game engine napravljen od istoimenog poduzeća smještenog u San Franciso-u. Sami game engine je napisan u C++-u no za skripte koristi C#. Danas je jedan od najkorištenijih engine-a za indie video igre. Neki od primjera su „Cuphead“, „PokemonGo“ i „Beat Saber“. [1]

Slika 1. Unity logotip

2.1.1. Unity Software Inc.

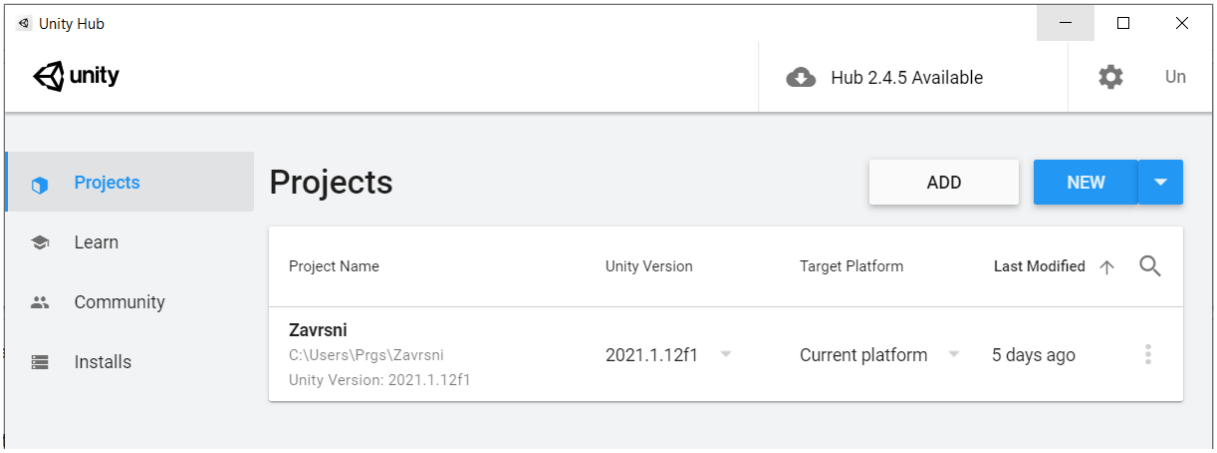
Američko poduzeće smješteno u San Francisco-u, osnovano u Danskoj 2004., fokusirano na izradu software-a za video igre. Orginalno ime poduzeća je „Over the Edge Entertainment“ (OTEE) te je promjenilo ime na Unity Software Inc. 2007. Prva video igra od Unity-a je bila „GooBall“ 2005. Sama video igra je bila komercijalno loša ali su poduzetnici shvatili da s engine-om koji su napravili mogu veoma olakšati budući razvoj video igrara te im je to postao novi cilj [2]. Kroz godine razvoja samog enigne-a Unity je zavladao marketom mobilnih video igrica. Sami razvoj u Unity-u je bio mnogo lakši od konkurencije te su sva indie poduzeća prešla na Unity. Do 2018. Unity je već bio u mogućnosti razvijati video igre za 25 platforme uključujući mobitele, konzole i osobna računala.

2.1.2. Unity game engine

Iako je Unity game engine trenutno cross-platform te podržava 25 platforme u početku je bio isljučivo za Mac OS X. Danas je najpopularniji za mobilne igre. Ima mogućnost razvoja 3D i 2D video igara kao i interaktivnih simulacija. Unity primarno koristi C# za skriptiranje. Prije C# se koristio Boo no nakon Unity 5 je zamjenjen.

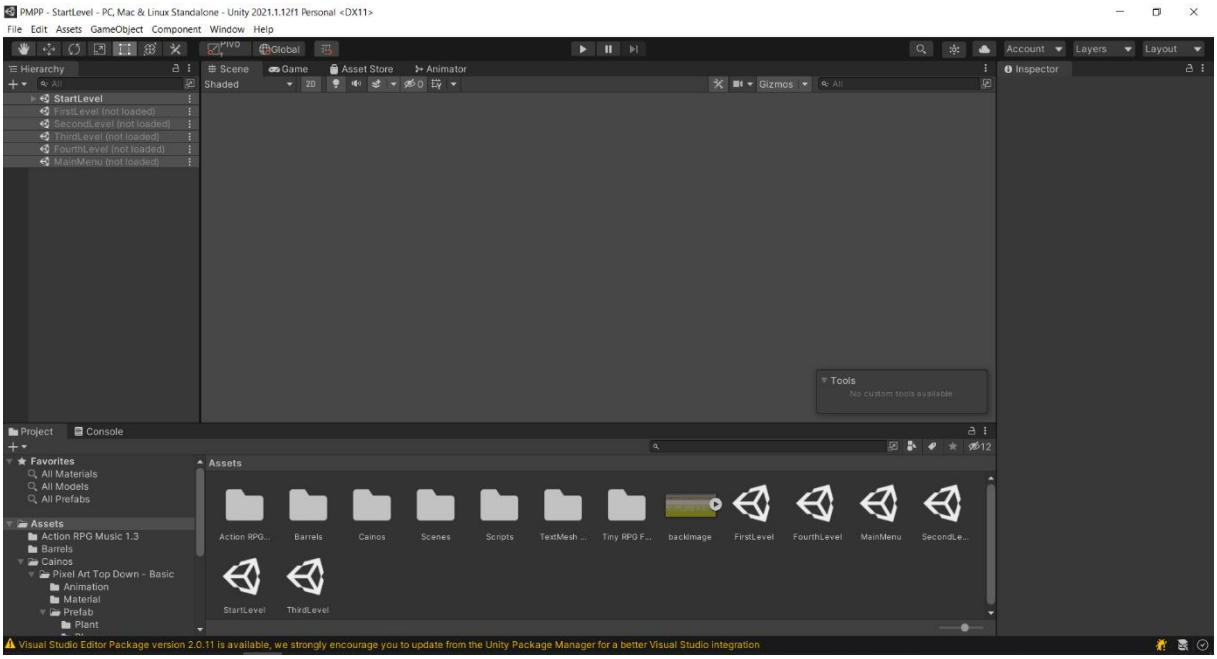
2.1.2.1. Sučelje

Pri samom otvaranju Unity-a otvara se „Unity Hub“ u kojem imamo izbor konfiguracije Unity verzije, kreiranje projekta, socijalni hub te lekcije u Unity-u.



Slika 2. Sučelje Unity Hub-a

U slici iznad u izborniku lijevo vidimo navedene opcije. Klikom na „NEW“ dobijemo izbornik koji nam omogućuje izbor između „2D“, „3D“, „3D With Extras“, „High Definition RP“ i „LWRP Templatea“ za naš projekt. Također odaberemo lokaciju gdje ćemo spremit projekt kao i njegov naziv. Mi smo kreirali 2D video igru tako da ćemo nadalje govoriti o tome.



Slika 3. Otvoreni projekt

U slici iznad je otvoren projekt „PMPP“. Lijevo pod „Hierarchy“ prozorom vidimo hijerarhiju svih scena te vidimo da je trentno učitan „StartLevel“. U prostoru ispod imamo dva prozora: „Project“ i „Console“. Po prozorom „Project“ imamo sve korištene materijale za samu video igru. Pod prozorom „Console“ je popis grešaka i upozorenja ako postoje.

2.2. C# skripte

Unity upravlja svojim objektima koristeći C# skripte. Pri kreiranju nove skripte automatski kreira dvije prazne metode „Start“ i „Update“. „Start“ se aktivira jednom pri pokretanju objekta povezanog s tom skriptom. „Update“ se pokreće jednom po svakom frame-u. Nakon pisanja skripte potrebno je pridjeliti objektu kako bi bila aktivna. Također imamo statične skripte koje nije potrebno pridjeliti objektu te se koriste za prenošenje informacija kroz scene.

3. Razrada teme

Motivacija za ovu video igru su bile igre kao „Chrono Trigger“(1995.) i „Undertale“(2015.). To što su ove video igre 20 godina razlike dokazuje to koliko dugo je popularan stil ptičje perspektive. U nastavku ću opisati implementaciju video igre u ptičjoj perspektivi u alatu Unity.

3.1. Tematika

Video igra počinje s glavnim likom zatočenim u zamku. Na samom početku se nalazi u vanjskom djelu zamka te jedini put je ući u zamak. Unutar zamka se nalaze tri prolaza s mozgalicom iza svakog prolaza. Potrebno je riješiti prvu mozgalicu kako bi mogli pristupiti drugoj, te riješiti drugu kako bi mogli pristupiti trećoj. Nakon rješavanja treće mozgalice pojavi se portal s kojim igra završava.

3.2. Paketi

Ovaj rad se fokusira na implementaciju logike video igre u alatu Unity te smo zbog toga koristili već postojeće pakete s „Unity Asset Store-a“. Pakete koje smo koristili su:

- „Pixel Art Top Down - Basic“ [3]
- „Tiny RPG – Forest“ [4]
- „Action RPG Music 1.3“ [5]

„Pixel Art Top Down-Basic“ je korišten za sve glavne teksture video igre. Glavni lik ovog paketa nije imao potrebne animacije i teksture te je zbog toga korišten „Tiny RPG – Forest“. Iz tog paketa je korišten glavni lik koji ima teksturu pucanja strijele iz luka što je nama potrebno. Iz „Action RPG Music 1.3“ je korištena glazba za video igru.

3.3. Kretanje igrača

3.3.1. Skripte za kretanje

Za igrača smo ubacili teksturu te napravili skripte za kretnju. Ukupno imamo 5 različitih skripta za kretnju. Više skripta smo pravili jer igrač ima različite mogućnosti u različitim razinama. U početnoj razini koristi se „Start Character“ skripta. Ona omogućuje kretanje u svim smjerovima te interakciju s vratima.

```
public class StartCharacter : MonoBehaviour
{
    public float speed;

    private Animator animator;
    private GameObject player;

    private void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        if (StaticScript.level == "FirstLevel")
        {
            player.transform.position = new Vector2(-3.85f, 13.55f);
        }
        animator = GetComponent<Animator>();

    }

    private void Update()
    {

        Vector2 dir = Vector2.zero;
        if (Input.GetKey(KeyCode.A))
        {
            dir.x = -1;
            animator.SetInteger("Direction", 3);
        }
        else if (Input.GetKey(KeyCode.D))
        {
            dir.x = 1;
            animator.SetInteger("Direction", 2);
        }

        if (Input.GetKey(KeyCode.W))
        {
            dir.y = 1;
            animator.SetInteger("Direction", 1);
        }
        else if (Input.GetKey(KeyCode.S))
        {
            dir.y = -1;
            animator.SetInteger("Direction", 0);
        }

        dir.Normalize();
        animator.SetBool("IsMoving", dir.magnitude > 0);
        GetComponent<Rigidbody2D>().velocity = speed * dir;

    }
}
```

Kod iznad je čitava skripta za kretanje igrača. Objekt „speed“ je varijabla koja određuje brzinu igrača. Objekt „animator“ služi za kontroliranje animacija igrača a objekt „player“ je naš glavni

lik. U „Start“ metodi inicijaliziramo naše objekte te u statičnoj skripti provjeravamo da li se igrač vraća iz „FirstLevel“ scene ili ne. Pomoću toga postavljamo igrača kraj prolaza za novu scenu umjesto na početnu lokaciju. U update djelu provjeravamo koju tipku igrač drži te na osnovu toga postavljamo smjer kretanja i animaciju za taj smjer kretanja. „dir.Normalize“ zaustavlja smjer kretanja i postavlja igrača u *idle*.

Za skriptu igrača u „FirstLevel“ morali smo dodati još čitanja iz statičke skripte. Sve ostalo je isto kao i skripti iznad.

```
if (StaticScript.level == "StartLevel")
            {
                player.transform.position = new Vector2(-0.05f, -3.91f);
            }
            if (StaticScript.level == "SecondLevel")
            {
                player.transform.position = new Vector2(-6.35f, 2.26f);
            }
            if (StaticScript.level == "ThirdLevel")
            {
                player.transform.position = new Vector2(-0.02006817f, 2.26f);
            }
            if(StaticScript.level == "FourthLevel")
            {
                player.transform.position = new Vector2(5.98f, 2.26f);
            }
```

Dakle dodali smo čitanja statičke skripte kako bi znali iz koje scene dolazi igrač te ga postavili na odgovarajuću lokaciju. Skripta koja zapisuje naziv scene u statičku skriptu izgleda ovako:

```
public class LevelName : MonoBehaviour
{

    public static string PreviousLevel { get; private set; }

    private void OnDestroy()
    {
        PreviousLevel = gameObject.scene.name;
    }

    void Start()
    {
        StaticScript.level = PreviousLevel;
    }

}
```

U „SecondLevel“ sceni igrač dobije mogućnost pucanja iz luka i strijele. Kako bi to omogućili u skriptu za kretanje izgleda ovako:

```
public class CharacterAiming : MonoBehaviour{
    public float speed;
    public float speedArw;
    private float timeLeft = 1f;
    private Animator animator;
    private GameObject arrow;
    private GameObject bow;

    private void Start()
    {
        animator = GetComponent<Animator>();
        arrow = GameObject.FindGameObjectWithTag("Arrow");
        bow = GameObject.FindGameObjectWithTag("Bow");
    }
    private void Update()
    {
        timeLeft -= Time.deltaTime;
        if (Input.GetKeyDown(KeyCode.Mouse0) && !bow.activeSelf && timeLeft<0) {
            animator.SetTrigger("Shoot");
            Vector2 target = Camera.main.ScreenToWorldPoint(new
            Vector2(Input.mousePosition.x, Input.mousePosition.y));
            Vector2 myPos = new Vector2(transform.position.x, transform.position.y);
            Vector2 direction = target - myPos;
            direction.Normalize();
            Quaternion rotation = Quaternion.Euler(0, 0, Mathf.Atan2(direction.y,
       direction.x) * Mathf.Rad2Deg);
            GameObject projectile = (GameObject)Instantiate(arrow, myPos, rotation);
            projectile.GetComponent<Rigidbody2D>().velocity = direction * speedArw;
            timeLeft = 1f;
        }
        Vector2 dir = Vector2.zero;
        if (Input.GetKey(KeyCode.A))
        {
            dir.x = -1;
            animator.SetInteger("Direction", 3);
        }
        else if (Input.GetKey(KeyCode.D))
        {
            dir.x = 1;
            animator.SetInteger("Direction", 2);
        }

        if (Input.GetKey(KeyCode.W))
        {
            dir.y = 1;
            animator.SetInteger("Direction", 1);
        }
        else if (Input.GetKey(KeyCode.S))
        {
            dir.y = -1;
            animator.SetInteger("Direction", 0);
        }
        dir.Normalize();
        animator.SetBool("IsMoving", dir.magnitude > 0);

        GetComponent<Rigidbody2D>().velocity = speed * dir;
    }
}
```

Osim ponovljenog koda za kretanje sada vidimo i nove objekte za luk i strijelu te brzinu strijele(„speedArw“). U metodi „Start“ smo incijalizirali potrebne objekte. U metodi „Update“ osim funkionalnosti za kretnju dodali smo *if* koji provjerava da li smo kliknuli lijevu tipku miša, da li je luk pokupljen unutar scene i da li je istekao „timeLeft“. „timeLeft“ je dodan kako igrač

ne bi mogao ispucati velik broj strijela odjednom nego jednu po sekundi. Unutar *if* postavljamo animatora na „Shoot“ te pronalazimo lokaciju miša. Nakon pronađene lokacije instanciramo strijelu te je pošaljemo prema lokaciji miša brzinom određenom s „speedArw“.

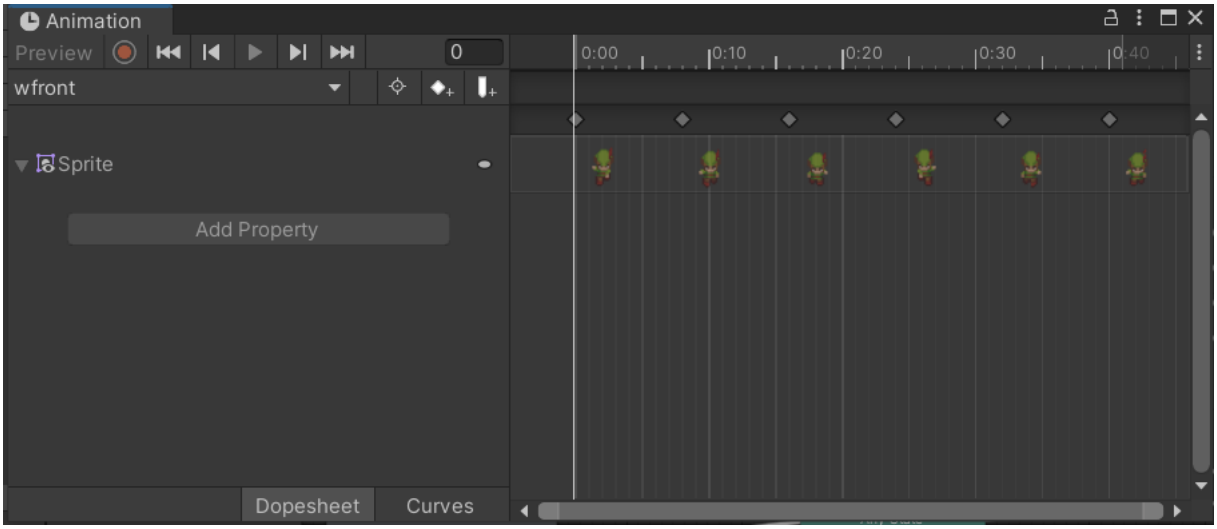
```
if (Input.GetKey(KeyCode.LeftShift))
{
    speed = 8;
    GetComponent<Rigidbody2D>().velocity = speed * dir;
}
else
{
    speed = 4;
    GetComponent<Rigidbody2D>().velocity = speed * dir;
}
```

U 4. skripti za kretanje smo na orginalnu skriptu samo dodali mogućnost bržeg kretanja. Držeći tipku „LeftShift“ brzina igrača se podupla. U petoj i zadnjoj skripti kretanja smo samo uklonili provjeru naziva prošle scene jer je samo jedan ulaz u razinu.

3.3.2. Animacije

Za animacije smo koristili niz skripti koje se ponavljaju u određenim intervalima i uvjetima. Unity za animacije ima ugrađene alate „Animator“ i „Animation“.

Same animacije kreiramo preko „Animation“ prozora. U prozor dodamo teksture koje želimo da budu u animaciji te ih rasporedimo po potrebi.



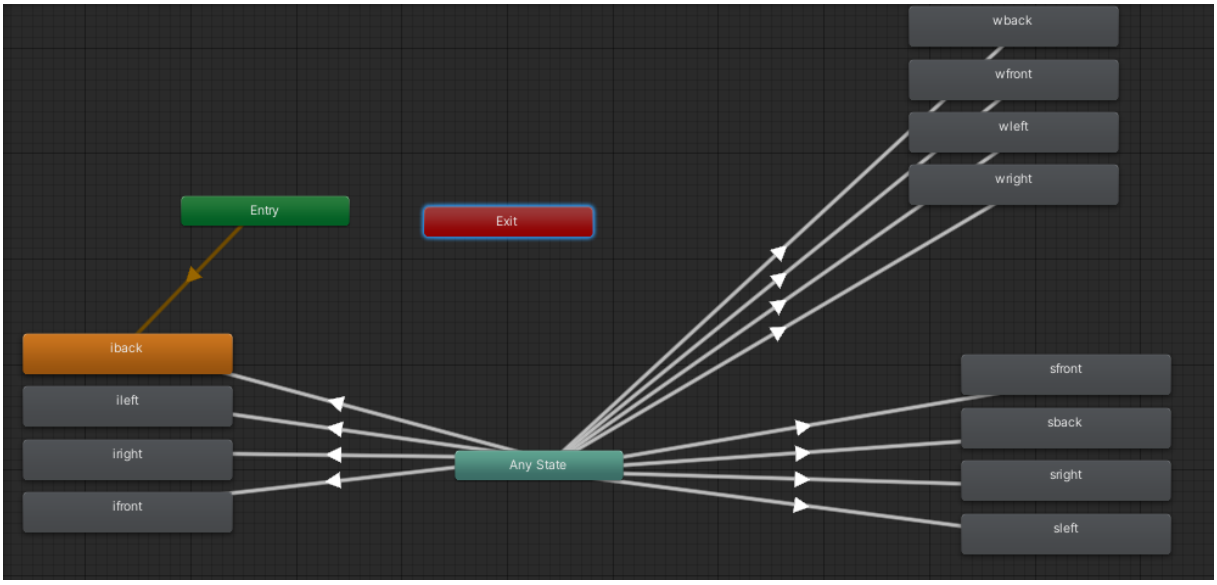
Slika 4. Animation prozor

Na slici iznad vidimo 6 različitih tekstura našeg glavnog lika te uz pomoć njih smo kreirali jednu od animacija. Na slici je prikazana animacija „wfront“ što je animacija za igračevo hodanje prema naprijed. No to je samo animacija, kako bi je koristili moramo je ubaciti u „Animator“.

„Animator“ spaja sve animacije te određuje varijable i uvjete po kojima se te animacije aktiviraju. U našem kodu imamo 3 uvjeta za aktivacije animacija:

- „Direction“
- „IsMoving“
- „Shoot“

„Direction“ je integer koji služi za određivanje smjera kretanja za animaciju. Na primjer ako je „Direction“ 2 onda je smjer animacije desno. Za kretanje osim „Direction“ se koristi i boolean „IsMoving“. U kodovima iznad smo vidjeli da se „IsMoving“ aktivira kada je brzina kretanja veća od 0. „Shoot“ je *Trigger* koji se aktivira pri pucanju s lukom i strijelom. U „Animator“ prozoru postavimo veze između animacija te njihove uvjete pokretanja. U mom radu to izgleda kao na slici ispod.



Slika 5. Animator

Kao što vidimo imamo 4 animacije za stanje mirovanja (*idle*), 4 animacija za kretanje (*walk*) i 4 animacije za pucanje luka (*shoot*). Početna animacija je „iback“ što je animacija mirovanja okrenuta leđima. Iz bilo kojeg stanja animacija se može u sva druga stanja animacije. Za to služi „Any State“. Uvjeti da se ide u animaciju kretanja su da je „IsMoving“ postavljen na *true* i da je određen „Direction“. Uvjet za stanje mirovanja je da je „IsMoving“ postavljen na *false*. Uvjet za pucanje looka je trigger „Shoot“. Za sve pozive animacija u skriptama pozivamo objekt „Animator“ te pozivamo metode kojima podesimo uvjete.

3.4. Ostale skripte

3.4.1. Statična skripta

Statična skripta je skripta na koju ne utječe promjena scene te se vrijednosti unutar skripte prenose iz scene u scenu.

```
public static class StaticScript
{
    public static bool key1 = false;
    public static bool key2 = false;
    public static bool moveBarrels = false;
    public static string level = "StartLevel";
}
```

U njoj imamo 3 *boolean* varijable te jednu *string* varijablu. Varijable „key1“ i „key2“ služe za zaključavanje druge i treće razine dok se ne pokupe ključevi. *Boolean* „moveBarrels“ služi za pokretanje pomičnih bačava koje služe kao mete za luk i strijelu. *String* „level“ upisuje zadnju posjećenu scenu te služi za postavljanje igrača kraj potrebnih ulaza pri povratku u prošlu scenu.

3.4.2. Skripta za imena scene

Skripta „LevelName“ služi za upisivanje zadnje posjećene scene u statičnu skriptu.

```
public class LevelName : MonoBehaviour
{
    public static string PreviousLevel { get; private set; }

    private void OnDestroy()
    {
        PreviousLevel = gameObject.scene.name;
    }

    void Start()
    {
        StaticScript.level = PreviousLevel;
    }

}
```

3.4.3. Skripte za vrata

U projektu imamo 5 skripti za vrata te je jedina razlika u nazivu scene koju učitaju.

```
public class Door : MonoBehaviour
{
    void OnTriggerStay2D(Collider2D other)
    {
        if(other.gameObject.name == "Player" && Input.GetKey(KeyCode.F))
        {
            SceneManager.LoadScene("FirstLevel");
        }
    }
}
```

Skripta omogućava učitavanje iduće scene ako igrač stoji na okidaču vrata i klikne tipku „F“.

3.4.4.Skripte za ključeve

Postoje dvije skripte za ključeve te je jedina razlika između dvije skripte promjena drugačijeg *booleana* u statičnoj skripti.

```
public class Key : MonoBehaviour
{
    GameObject key;

    void Start()
    {
        key = GameObject.FindGameObjectWithTag("Key");
    }
    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.name == "Player")
        {
            key.SetActive(false);
            StaticScript.key1 = true;
        }
    }
}
```

Skripta u metodi „Start“ incijalizira objekt ključa. Kada igrač dodirne ključ on postaje neaktivan u sceni te se u statičnoj skripti *boolean* „key1“ postavi na *true*.

3.4.5.Skripte za bačve

Postoji šest skripti za bačve. Skripte služe kako bi se bačve micale po zadanom uzorku. Šest skripti postoji jer svaka bačva ima svoj uzorak micanja.

```
public class Barrel : MonoBehaviour
{
    public float speed;

    GameObject arrow;
    GameObject barrel;

    Vector2 dir = Vector2.zero;
    int time=200;
    int timer;
    bool side;

    void Start()
    {
        arrow = GameObject.FindGameObjectWithTag("Arrow");
        barrel = GameObject.FindGameObjectWithTag("Barrel");
        timer = time;
        side = true;
    }

    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.tag == "Arrow")
        {
            barrel.SetActive(false);
        }
    }
    void Update()
    {
        if (StaticScript.moveBarrels)
        {
            if (side)
            {
                timer -= 1;
                dir.x = 1;
                dir.Normalize();
                GetComponent<Rigidbody2D>().velocity = speed * dir;
                if (timer == 0)
                {
                    side = false;
                    timer = time;
                }
            }
            if (!side)
            {
                timer -= 1;
                dir.x = -1;
                dir.Normalize();
                GetComponent<Rigidbody2D>().velocity = speed * dir;
                if (timer == 0)
                {
                    side = true;
                    timer = time;
                }
            }
        }
    }
}
```

U kodu iznad prije metoda smo postavili strijele(„arrow“), bačve(„barrel“), smjera(„dir“), vremena(„time“), brojaca(„timer“) i strane(„side“). Sve potrebne smo u metodi „Start“

inicijalizirali. U metodi „OnTriggerEnter2D“ stavili smo da ako strijela dodirne bačvu da bačva postane neaktiva(nestane). U metodi „Update“ provjeravamo da li je u statičnoj skripti *boolean* „moveBarrels“ postavljen na *true* a postavlja se kada pokupimo luk. Nakon toga imamo dva *if* koji provjeravaju varijablu „side“. Unutar *if* smanjujemo timer te mičemo bačvu u određenom smjeru. Varijabla „side“ služi za promjenu smjera kretanja, a varijabla „timer“ koliko se bačva kreće u jednom smjeru.

3.4.6.Skripta za luk

Skripta za luk mjenja *boolean* statične skripte te deaktivira svoj objekt unutar scene.

```
public class Bow : MonoBehaviour
{
    private GameObject bow;

    void Start()
    {
        bow = GameObject.FindGameObjectWithTag("Bow");
    }

    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.name == "Player")
        {
            bow.SetActive(false);
            StaticScript.moveBarrels = true;
        }
    }
}
```

Kao što vidimo na kodu ako igrač dodirne luk on on se deaktivira i bačve se počnu pomicati.

3.4.7.Skripta za blokade

Imamo dvije skripte za blokade ključeva. Obe skripte rade na sličan način ali s različitim uvjetima. Prva skripta uklanja 3 stuba nakon što igrač pogodi sve bačve.

```
public class Gate : MonoBehaviour
{
    GameObject barrel;
    GameObject barrel1;
    GameObject barrel2;
    GameObject barrel3;
    GameObject barrel4;
    GameObject barrel5;
    GameObject leftPillar;
    GameObject middlePillar;
    GameObject rightPillar;

    void Start()
    {
        barrel = GameObject.FindGameObjectWithTag("Barrel");
        barrel1 = GameObject.FindGameObjectWithTag("Barrel1");
        barrel2 = GameObject.FindGameObjectWithTag("Barrel2");
        barrel3 = GameObject.FindGameObjectWithTag("Barrel3");
        barrel4 = GameObject.FindGameObjectWithTag("Barrel4");
        barrel5 = GameObject.FindGameObjectWithTag("Barrel5");
        leftPillar = GameObject.FindGameObjectWithTag("LeftPillar");
        middlePillar = GameObject.FindGameObjectWithTag("MiddlePillar");
        rightPillar = GameObject.FindGameObjectWithTag("RightPillar");

    }

    void Update()
    {
        if (!barrel.activeInHierarchy && !barrel1.activeInHierarchy &&
!barrel2.activeInHierarchy && !barrel3.activeInHierarchy && !barrel4.activeInHierarchy
&& !barrel5.activeInHierarchy)
        {
            middlePillar.SetActive(false);
            leftPillar.SetActive(false);
            rightPillar.SetActive(false);
        }
    }
}
```

Kao što vidimo kod provjerava postojanje svih 6 bačava unutar scene te tek kada su sve bačve neaktivne oslobodi se prolaz prema ključu. U drugoj skripti blokade rješava se mozgalica u kojoj imamo tri objekta i dva se moraju deaktivirati a treći ostati aktivan kako bi se blokada uklonila.

```
public class Gateway : MonoBehaviour
{
    public GameObject gateway;
    private GameObject trigger1;
    private GameObject trigger2;
    private GameObject trigger3;

    void Start()
    {
        gateway = GameObject.FindGameObjectWithTag("Gateway");
        trigger1 = GameObject.FindGameObjectWithTag("Trigger1");
        trigger2 = GameObject.FindGameObjectWithTag("Trigger2");
        trigger3 = GameObject.FindGameObjectWithTag("Trigger3");

    }

    void Update()
    {
        if(!trigger1.activeSelf && trigger2.activeSelf && !trigger3.activeSelf)
        {
            gateway.SetActive(false);
        }
    }
}
```

Kao što vidimo u kodu potrebno je deaktivirati „trigger1“ i „trigger3“ a ostaviti aktivnim „trigger2“.

3.4.8.Skripte za prekidače

Skripte za prekidače se koriste uz drugu skriptu za blokadu. Imamo tri skripte za prekidače.

```
public class Switch1 : MonoBehaviour
{

    public GameObject trigger;

    void Start()
    {
        trigger = GameObject.FindGameObjectWithTag("Trigger1");
    }

    private void OnTriggerStay2D(Collider2D other)
    {
        if (other.gameObject.tag == "Box")
        {
            trigger.SetActive(false);
        }
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.gameObject.tag == "Box")
        {
            trigger.SetActive(true);
        }
    }

}
```

U skripti imamo objekt „trigger“ koji se deaktivira kada dođe u kontakt s kutijom. Također kada se kutija skloni s „trigger“ objekta onda se on opet aktivira.

3.4.9.Skripta za kutiju

Kako bi omogućili micanje kutije po sceni na željeni način svakoj kutiji smo dodjelili skriptu „Box“.

```
public class Box : MonoBehaviour
{
    private Rigidbody2D body;
    void Start()
    {
        body = GetComponent<Rigidbody2D>();
    }

    void OnCollisionEnter2D(Collision2D coll)
    {
        if (coll.gameObject.tag == "Player")
        {
            body.isKinematic = false;
            body.constraints = RigidbodyConstraints2D.None;
            body.constraints = RigidbodyConstraints2D.FreezeRotation;
        }
    }

    void OnCollisionExit2D(Collision2D coll)
    {
        if (coll.gameObject.tag == "Player")
        {
            body.isKinematic = true;
            body.constraints = RigidbodyConstraints2D.FreezePosition;
        }

    }

}
```

Postavljamo da kada igrač dođe u dodir s kutijom omogućava joj se kretanje a kada se odmakne od kutije ona se zamrzne na mjestu gdje stoji.

3.4.10. Skripta za početni izbornik i kraj igre

Skripta za početni izbornik ima dvije metode koje odgovaraju tipkama na početnom izborniku.

```
public class MainMenu : MonoBehaviour
{
    public void StartGame()
    {
        SceneManager.LoadScene("StartLevel");
    }

    public void ExitGame()
    {
        Application.Quit();
    }
}
```

Kod iznad pokazuje metode koje su povezane na tipke.

```
public class EndGame : MonoBehaviour
{
    void OnTriggerStay2D(Collider2D other)
    {
        if (other.gameObject.name == "Player")
        {
            SceneManager.LoadScene("MainMenu");
        }
    }
}
```

Kod iznad pokazuje kada uđemo u portal na kraju zadnje scene igra nas vraća natrag na početni izbornik.

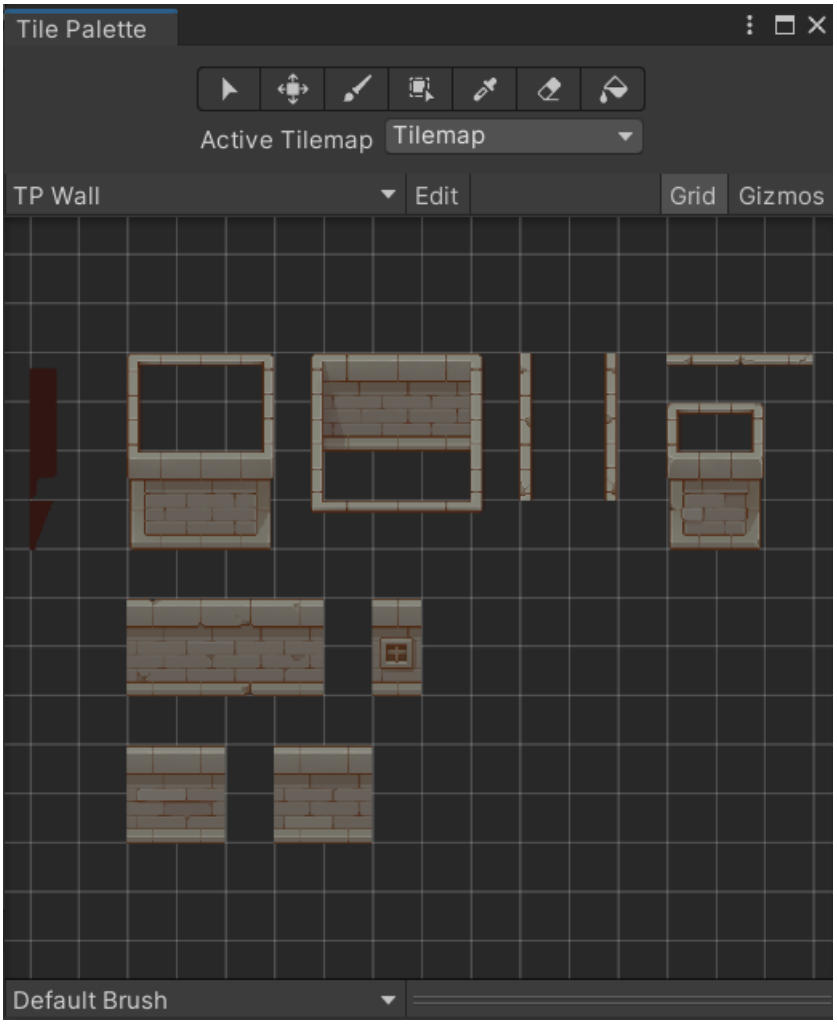
3.4.11. Skripta za pokazivanje teksta

Kroz razine nam se pojavljuje tekst koji govori što trebamo raditi. Kako bi taj tekst nestao kada se odmaknemo morali smo mu dodati skriptu.

```
public class TextAppear : MonoBehaviour
{
    private GameObject popup;

    void Start()
    {
        popup = GameObject.FindGameObjectWithTag("Popup");
        popup.SetActive(false);
    }

    private void OnTriggerStay2D(Collider2D other)
    {
        if (other.gameObject.tag == "Player")
        {
            popup.SetActive(true);
        }
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.gameObject.tag == "Player")
        {
            popup.SetActive(false);
        }
    }
}
```

3.5. Dizajniranje razina

Za dizajn razina sam koristio *Grid* (mrežu) te *Tile Palette*. Ubacivanjem *grid* u video igru dobijemo mogućnost ubacivanja *Tile*. Teksture za dizajn su korištene iz paketa.



Slika 6. Tile Palette

Na slici iznad vidimo izgled samog alata za *Tile*. Kvadratići iza tekstura su dimenzija našeg postavljenog *grid* te možemo odabrati željene kvadratiće koje kistom prebacimo na naš *grid*. Osim kista i selekcije imamo mogućnost micanja selekcije, punjenja prostora s selekcijom te čitavu razinu prekriti selekcijom. Također možemo vidjeti da je odabrani *tilemap* „Tilemap“ te da smo na paleti „TP Wall“. Osim „TP Wall“ koristili smo dvije druge palete. Također smo definirali drugi *tilemap* na kojeg smo ubacili sudarač (*collider)* koji blokira igrača u prolasku.

3.5.1.Dizajn početnog izbornika

Početni izbornik se sastoji od pozadine napravljene s „TilePalette“ alatom. Ispred pozadine imamo naziv video igre i dvije tipke. Prva tipka služi za početak igre dok druga za napuštanje.

Slika 7. Početni izbornik

3.5.2. Dizajn prve razine

Prva razina predstavlja jedini vanjski dio video igre. Služi kako bi se igrač upoznao s početnom kretnjom i interakcijom s vratima.

Slika 8. Prva razina

Kao što možemo vidjeti na slici prva razina nazvana „StartLevel“ se sastoji od ukrasnih djelova, igrača te vrata. Mogućnosti igrača na ovoj razini su kretanje i prolazak kroz vrata. Ostatak levela je ukrasan. Na djelu gdje igrač počne imamo tekstualne instrukcije koje se deaktiviraju kada se igrač odmakne.

3.5.3. Dizajn druge razine

Druga razina služi kao izbornik za tri mozgalice. Unutar Unity-a je nazvana „FirstLevel“.

Slika 9. Druga razina

Pri ulasku u drugu razinu imamo tekstualne instrukcije koje se deaktiviraju odmicanjem i ponovno aktiviraju prilaskom. Kada igrač prvi put posjeti drugu razinu ima mogućnost ući samo u lijeva vrata koja vode u treću razinu. Pri izlasku iz lijevih vrata, nakon rješene mozgalice, igrač otključava vrata u sredini. Također se koriste statična skripta i skripta za naziv razina kako bi se igrač postavio ispred vrata iz kojih je došao, a ne na početnu poziciju. Nakon rješene druge mozgalice igraču se otključavaju treća i zadnja vrata.

3.5.4. Dizajn treće razine

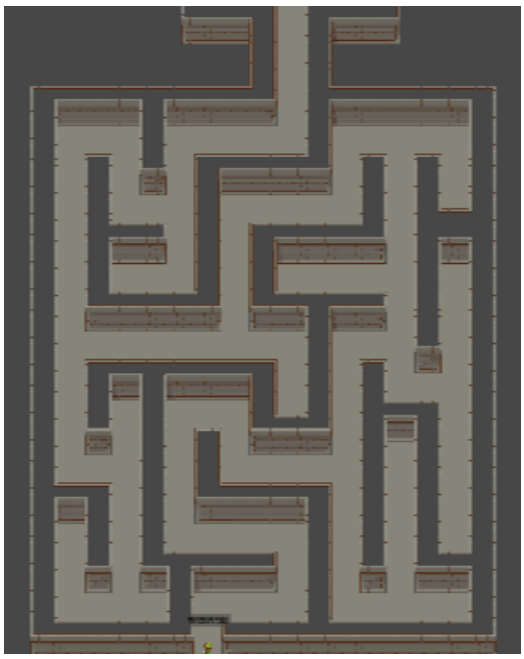
Treća razina, u Unity-u nazvana „SecondLevel“, sadrži mozgalicu pucanja bačvi.

Slika 10. Treća razina

Pri ulasku u razinu igrač je predstavljen s lukom i zaključanim ključem. Prilaskom do luka aktivira se tekstualna pomoč te igrač uzme luk. Nakon što je igrač uzeo luk bačve se počnu micati te ih igrač mora sve pogoditi kako bi otključao prolaz dalje. Svaka bačva ima vlastitu skriptu micanja te se nijedna ne miče identično.

3.5.5. Dizajn četvrte razine

Četvrta razina je labirint. U projektu nosi naziv „ThirdLevel“.

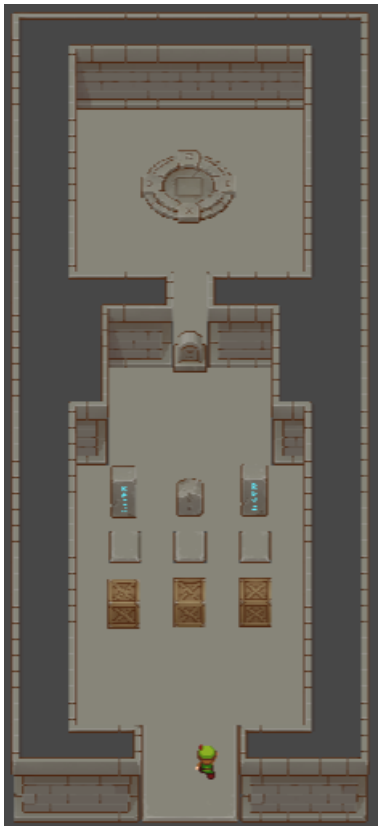


Slika 11. Četvrta razina

Slika iznad pokazuje dizajn samog labirinta. Pri ulazu u labirint pojavljuje se tekst s instrukcijama. U ovoj razini igrač dobije mogućnost brzog kretanja. Držeći tipku „Lshift“ igračeva brzina se povećava za 50%. Na kraju labirinta igrač uzima ključ te se mora vratiti nazad do ulaza. Brzo kretanje je dodano kako bi skratilo dužinu ovog djela video igre.

3.5.6. Dizajn pete razine

Zadnja razina nazvana „FourthLevel“ u projektu je razina s mozgalicom i portalom za kraj video igre.



Slika 12. Peta razina

U razini imamo tri kutije, tri prekidača i tri stuba. Prolaz dalje je blokiran te se može pretpostaviti da je potrebno gurati kutije na prekidače. Ako gurnemo sve kutije prolaz je i dalje blokiran. Potrebno je gurnuti kutije jedan i tri kako bi se prolaz oslobodio. Prekidač dva treba biti neaktiviran. Kao pomoć pri rješavanju mozgalice postoji stubovi. Stub jedan i tri svijetle te je to indikator da je potrebno aktivirati prekidač jedan i tri.

4. Zaključak

Video igra „Pustolovna mozgalica iz ptičije perspektive“ je izrađena koristeći alat Unity i C# skripti jezik. Igra je izrađena u Unity2D radnom prostoru. U projektu se fokusiramo na korištenje C# skripti kao interakciju igrača s okolinom.

Cilj igre je rješiti 3 mozgalice i ući u završni portal. Prva mozgalica je gađanje meta lukom i strijelom. Druga mozgalica je labirint. Treća mozgalica je zagonetka s kutijama. Svaka mozgalica donosi nešto novo u samu igru. U prvoj igrač dobije mogućnost pucati iz luka. U drugoj dobije mogućnost brzog trčanja. U trećoj dobije mogućnost guranja kutija.

Cilj završnog rada je bio naučiti izradu video igara u Unity alatu. Smatram da sam izradom ovog rada naučio dovoljno da se uvedem u izradu video igara.

Popis literature

[1] *Unity (game engine)* (kolovoz 2021.), U Wikipedia. Preuzeto 16. kolovoza 2021. s https://en.wikipedia.org/wiki/Unity_(game_engine)

[2] *Unity Technologies* (kolovoz 2021.), U Wikipedia. Preuzeto 16. kolovoza 2021. s https://en.wikipedia.org/wiki/Unity_Technologies

[3] *Pixel Art Top Down – Basic* (kolovoz 2021.), U Unity Asset Store, Preuzeto 1. kolovoza 2021. s https://assetstore.unity.com/packages/2d/environments/pixel-art-top-down-basic-187605

[4] *Tiny RPG – Forest* (kolovoz 2021.) U Unity Asset Store, Preuzeto 1. kolovoza 2021. s https://assetstore.unity.com/packages/2d/characters/tiny-rpg-forest-114685

[5] *Action RPG Music Free* (kolovoz 2021.) U Unity Asset Store, Preuzeto 1. kolovoza 2021. s https://assetstore.unity.com/packages/audio/music/action-rpg-music-free-85434

Popis slika