Razvoj arenske platformske 2D računalne igre u Unity okruženju

Mažar, Krešimir

Sveučilište Jurja Dobrileu Puli
Fakultet informatike

KREŠIMIR MAŽAR

RAZVOJ ARENSKE PLATFORMSKE RAČUNALNE 2D IGRE U UNITY OKRUŽENJU

Završni rad

Rujan, 2020.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

KREŠIMIR MAŽAR

RAZVOJ ARENSKE PLATFORMSKE RAČUNALNE 2D IGRE U UNITY OKRUŽENJU

Završni rad

JMBAG: 0016102024, izvanredni student
Studijski smjer: Informatika

Predmet: Programiranje
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Mentor: izv. prof. dr. sc. Tihomir Orehovački

Rujan, 2020.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Krešimir Mažar, 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,_____,_____godine

IZJAVA

o korištenju autorskog djela

Ja, Krešimir Mažar dajem odobrenje Fakultetu informatike u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom Razvoj arenske plaformske 2D računalne igre u Unity okruženju 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,_____(datum)

Potpis

_____

Sažetak

Ovaj rad opisuje stvaranje 2D igre Arena Battle u Unity okruženju. U radu je opisan alat Unity, te neke njegove funkcionalnosti. Igra se sastoji od 5 scena. Prva scena prikazuje glavni izbornik. Prva mogućnost prikazuje kratak video s prikazom igre i kontrola koje se koriste. Iduća mogućnost je odigravanje igre, gdje igrača napadaju protivnici, koje igrač mora uništiti da bi preživio. Slijede mogućnosti pregleda rezultata koje je igrač postigao. U radu se detaljno opisuju skripte uz pomoć kojih je nastala igra.
Ključne riječi: Unity Engine, Arena Battle, C#

Abstract

This paper describes the creation of 2D Arena Battle game in a Unity environment. The paper describes the Unity tool, some of its functionalities. The game consists of 5 scenes. The first scene shows the main menu. The first option shows a short video showing the games and controls used. The next option is to play game, where opponents attacks player that you have to shoot to survive. The following are options to view the results the player has achieved. The paper describes in detail the scripts with which the game was created.

Keywords: Unity Engine, Arena Battle, C #

Sadržaj

1. Uvod

Industrija video igara zadnji niz godina zabilježila je nagli razvoj računalnih programa za kreiranje video igara. Igre su oplemenjene zavidnom grafikom. Igranje video igara je popularno u svim uzrastima današnjeg doba. Industrija video igara je ozbiljna grana informacijskih tehnologija, koja zapošljava mnogo stručnjaka i generira više stotina milijardi dolara godišnje u cijelom svijetu. Sve je više kompanija koji izrađuju igre, pa tako su se i u Hrvatskoj pojavile svjetske poznate igre kompanija kao što su CroTeam, Nanobit i Gamepires.

Unity alat je jedan od najpopularnijih alata za kreiranje igara, te po riječima mnogih developera najbolji alat za 2D igre. Unity omogućava stvaranje 2D i 3D video igara za brojne platforme. Tijekom godina Unity je razvijao nove funkcionalnosti i inovacije unutar svog alata kako bi omogućio što lakše korištenje.

Cjelokupan završni rad sastoji se 5 poglavlja uključujući uvod i zaključak. Svrha rada je opisati korištenje Unity alata kroz razvoj vlastite 2D igre. Smisao kreirane 2D igre pod nazivom Arena Battle je preživjeti napade protivnika, te tako skupiti što više bodova. Igra je kreirana za operativni sustav Windows.

Drugo poglavlje odnosi se na opis korištenog alata, njegovu povijest i definiranje svih verzija Unity alata koje su razvijene od osnutka tvrtke pa sve do danas. Glavni motiv za razvoj ovog alata je bio taj što su njegovi tvorci uvidjeli mali broj funkcionalnosti u tadašnjim alatima. Upravo iz tog razloga svaka nova verzija nosila je sa sobom niz novih funkcionalnosti. Unutar ovog poglavlja opisan je i sam izgled Unity alata, odnosno njegovo sučelje, funkcije i komponente.

Sljedeće poglavlje opisuje izradu odabrane 2D igre Arena Battle. Za izradu je korištena verzija 2018.4.11f1 Unity alata dok je za pisanje skripti korišten Visual Studio, a Photoshop za modificiranje modela.

Četvrto poglavlje fokusira se na samu implementaciju. Opisuje sve korištene komponente i skripte za izradu igre.
Posljednje poglavlje donosi zaključak odabranog rada, te zatim slijedi popis korištene literature i popis prikazanih slika.

2. Unity okruženje

Unity je računalni program pomoću kojeg se razvijaju 2D i 3D video igre. Video igre izgrađuju se za raznolike platforme kao što su Linux, Mac, Playstation 4, PC i ostale. Razvijen je 2005. godine od strane Unity Technologies. Njegovi tvorci su Nicolas Francis, David Helgason i Joachim Ante koji su tvrtku Unity Technologies osnovali godinu ranije u Danskoj. Stvarajući svoju igricu GooBall uvidjeli su nedostatke u razvojnim okvirima te odlučili stvoriti vlastiti koji će biti pristupačan za korištenje prosječnim ljudima. Cilj im je bio omogućiti jednostavno i intuitivno razvojno okruženje te stvoriti razvojni okvir s mnoštvom značajnih funkcionalnosti. Vrlo brzo Unity alat je stekao popularnost zbog mogućnosti korištenje na različitim uređajima. Cijelo razvojno okruženje napisao je u C i C++ programskom jeziku. Na jezgri sustava stvoren je omotač koji predstavlja sloj za pristup .NET jezika što olakšava razvoj igrica korisnicima. Unity zbog mnoštva funkcionalnosti koji pruža korisniku na izbor samo klikom miša olakšava cjelokupni proces izrade igrice. Takav izbor funkcionalnosti skraćuje i vrijeme razvoja, jer za neke od ponuđenih funkcionalnosti bi implementacija mogla trajati i do nekoliko tjedana [1].

2.1. Verzija

Prva verzija koja je donijela nove funkcionalnosti nakon verzije 1.0. bila je verzija 2.0. koja je stvorena 2007. godine.
Nove funkcionalnosti su bile:

- Alati za rad s terenom
- Sustav za kreiranje korisničkog sučelja
- Osjenčavanje u stvarnom vremenu [2]

Funkcionalnosti koje su došle s izradom Unity 3.0. verzije 2010. godine su:

- Mapiranje svijetla
- Umetanje zvučnih efekata
- Podrška za C#

- Podrška za implementaciju igre za android operacijske sustave [3]

U 2012. godini izašla je nova 4.0. verzija koja je donijela nove funkcionalnosti:

- Podrška za DirectX
- Novi sustav za animacije
- Osjenčavanje u stvarnom vremenu za mobilne uređaje [4]

Verzija 5.0. koja je izašla 2015. godine imala je nove funkcionalnosti kao što su:

- Novi audio mixer
- Unaprjeđenje procesa animacije
- Osvjetljenje u realnom vremenu
- Podršku za implementaciju igara na gotovo svim platformama [5]

U prosincu 2016 godine mijenjaju nazive verzija pa se iduća verzija naziva Unity 2017 koja je imala nove funkcionalnosti:

- Grafičko prikazivanje u stvarnom vremenu
- Ocjenjivanje boje i izgradnja svijeta
- Timeline za animacije
- 2017.2 integrira 3DS i Mayu [6]

Sljedeća verzija je Unity 2018 s novim funkcionalnostima:

- Novi način prikazivanja grafike
- Bolja grafika za virtualnu, produženu i mješovitu stvarnost
- Alati za strojno učenje [7]

Zadnja verzija je Unity 2019 s novim funkcionalnostima:

- Poboljšanje grafike i osvjetljenja
- Poboljšanje performansi
- Poboljšanja fizike
- Podrška H.265 videa [8]

2.2. Sučelje

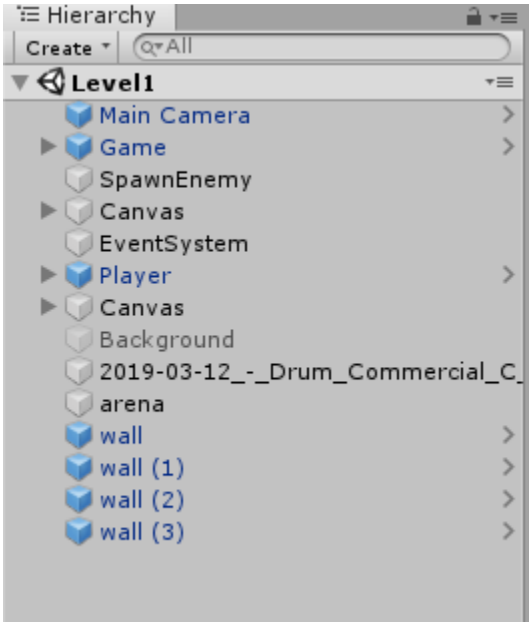
Sučelje alata Unity, prikazano na slici 1, se sastoji od komponenata: Alatna traka (eng. The ToolBar), Hijerarhije (eng. The Hierarchy window), Prikaza igre (eng. The Game view), Prikaza scene (eng. The Scene view), Inspektora (eng. The Inspector window) i Projekta (eng. The Project window) [9].

Slika 1. Prikaz Unity sučelja (izvor:autor)

Na slici 2 je prikazana Alatna traka koja s lijeve strane sadrži osnovne alate za manipulaciju objekta. U sredini se nalazi dugme za pauziranje i pokretanje igre. S desne strane se nalazi dugme za pristup računu i servisima, te vidljivost slojeva i rasporeda [9].

Slika 2. Prikaz Alatne trake (izvor:autor)

Na slici 3 je prikazan Prozor hijerarhija koji omogućuje prikaz svih objekata koji se nalaze na sceni. Otkriva strukturu kako su objekti povezani [9].

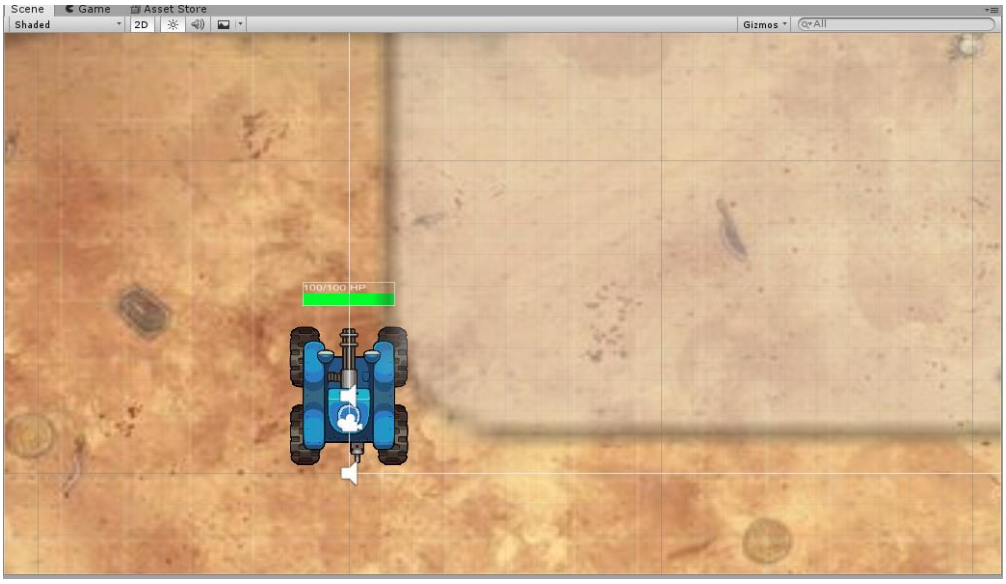


Slika 3. Prozor hijearhija (izvor:autor)

Na slici 4 je prikazan Prikaz igre koji nam prikazuje konačni prikaz igre koji se vidi kamerom. Igra se pokreće na pritisak dugma [9].

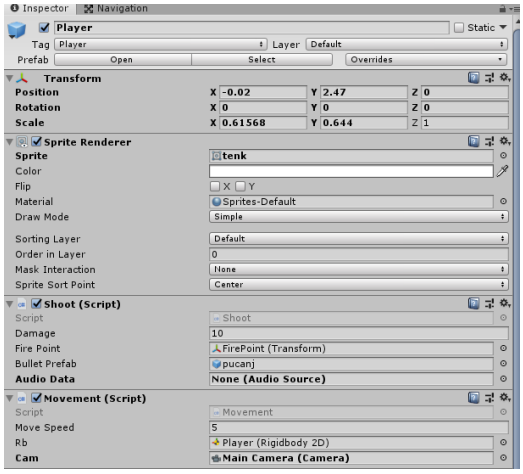
Slika 4. Prikaz igre (izvor:autor)

Na slici 5 je prikazana Scena koja omogućuje uređivanje i dodavanje objekata u igru. Podržava 2D i 3D perspektivu, što je u ovom slučaju 2D [9].



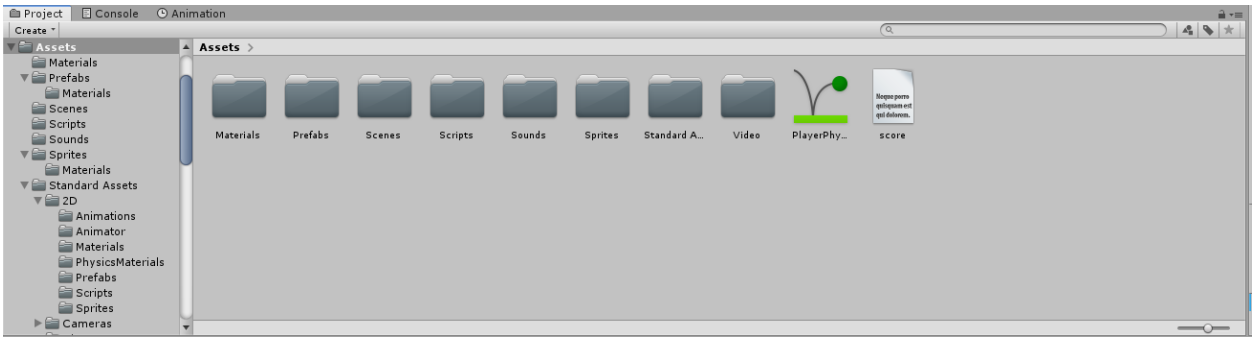
Slika 5. Scena (izvor:autor)

Na slici 6 je prikazan Prozor inspektora koji omogućuje pregled i uređivanje svojstava odabranog objekta. Kada se pokrene igra, postoji mogućnost mijenjanja svojstva nekog objekta [9].



Slika 6. Prozor inspektora (izvor:autor)

Na slici 7 je prikazan Prozor projekta koji prikazuje sve datoteke koji se nalaze u projektu, koje se mogu koristiti za izgradnju igre [9].



Slika 7. Prozor projekta (izvor:autor)

2.3. Funkcija

Unity sadrži ugrađene funkcije koje imaju zadanu ulogu. U nastavku su opisane korištene funkcije za izradi igre. Funkcija Početak (eng. Start) omogućuje izvršenje zadanog koda kada se pokrene skripta. Funkcija Osvjesti (eng. Awake) se poziva kada je objekt skripte inicijaliziran, neovisno je li skripta omogućena [10] .

Funkcija Ažuriraj (eng. Update) omogućuje prikaz svakog okvira u igri. Koristi se da bi se prikazale neke promjene koje se događaju na objektima. Slična funkcija je FixedUpdate, no ona prikazuje promjene svakih nekoliko okvira u igri. Funkcija OnCollisionEnter2D omogućuje detekciju kontakta između objekata. Kada se kontakt dogodi funkcija detektira, te se izvršava zadana akcija [11].

2.4. Komponenta

Unity sadrži mnoge komponente koje olakšavaju izgradnju igre. Najbitniji dio je Objekt (eng. GameObjects). Svaki objekt ima komponentu Transform, koja daje objektu položaj i orijentaciju, te se ne može ukloniti [12].

RigidBody2D omogućuje utjecaj fizike na zadani objekt. Služi za kreiranje uvjerljivog kretanja ili sudar s drugim objektom, bez korištenja kodiranja. Komponenta se sastoji od mnogo varijabli s kojima možemo postići željeno ponašanje objekta [13].

BoxCollider2D je nevidljivi pravokutni okvir koji se stavlja na objekte kako bi se objekti mogli detektirati međusobno. Moguće je koristiti više komponenti na jednom objektu, te proizvoljno odrediti veličinu.

Izvor zvuka (eng. Audio Source) omogućuje korištenje zvuka u igri. Omogućuje korištenje raznih funkcija i efekata koji omogućuju manipuliranje zvukom [15].

Video Player omogućuje dodavanja videa u igri. Također sadrži razne funkcije,te efekte koji omogućuju manipuliranje videom [16].

Prefab je objekt koji se sprema u projekt sa svim komponentama i svojstvima. Služi za ponovno korištenje objekta u projektu. Omogućuje automatsko sinkroniziranje svih kopija koje postoje [17].

3. Razvoj igre Arena Battle

Izrada igre je prikazana pomoću igre Arena Battle, prikazana na slici 8. Igru predstavlja igrač koji se može upravljati uz pomoć tipki na tipkovnici, te rotirati uz pomoć pokazivača miša. Igrača napadaju protivnici koje igrač može uništiti tako da ispali dovoljan broj metaka. Protivnik se napadom uništavaju, te smanjuje igraču energiju. Uništenjem svakog protivnika igrač osvaja 10 bodova,te se na kraju igre bodovi spremaju i prikazuju. Igrač tijekom igre može pokupiti pomoć kako bi duže preživio u igri.

Slika 8. Igra Arena Battle (izvor:autor)

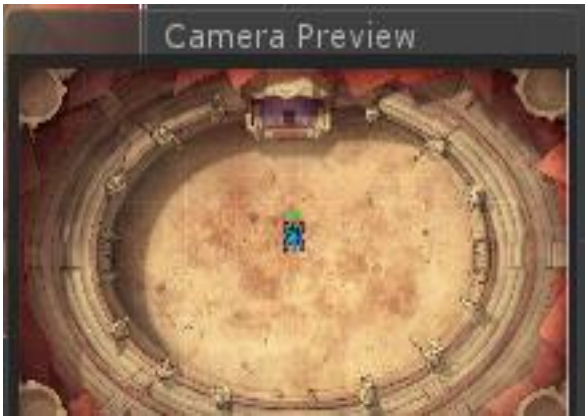
Igra je kreirana uz pomoć Unity alata verzije 2018.4.11f1, te je korišten Visual Studio za pisanje skripti i Photoshop za dorađivanje modela.

4. Implementacija

U ovom odjeljku će biti opisane skripte i komponente s kojima je kreirana igra u Unity alatu.

4.1. Kamera

Unity objekt Kamera (eng. Main Camera) daje prikaz koji igrač vidi, prikazan na slici 9. Kamera je statična te prikazuje scenu s pozadinom. U svim scenama kamera je s ne promijenjenim svojstvima, osim scene koja nam prikazuje igru gdje kamera prikazuje igru.



Slika 9. Prikaz kamere u Unity (izvor:autor)

4.2. Igrač

Na slici 10 je prikazan igrač kojeg možemo kontrolirati i ispaljivati metke u igri. Koristi Rigidbody2D komponentu koja omogućuje utjecaj fizike na objekt. BoxCollider2D komponenta omogućuje i detektira interakciju između više objekata. Koristi Izvor zvuka (eng. Audio Source) komponentu koja omogućuje puštanje zadanog zvuka.

Slika 10. Prikaz igrača [17]

Programski kod 1 prikazuje dijelove skripte Player. Sadrži PlayerStats klasu koja sadrži varijable u kojima se nalazi maksimalna energija koju igrač može imati i trenutna energija koja se smanjuje kada ga protivnik napadne. Funkcijom Init trenutna energiju postaje maksimalna.

```
public class PlayerStats
{
    public int maxHealth = 100;

    private int _curHealth;

    public int curHealth
    {
        get { return _curHealth; }
        set { _curHealth = Mathf.Clamp(value, 0, maxHealth); }
    }

    public void Init()
    {
        curHealth = maxHealth;
    }
}
```

Programski kod 1. Svojstva igrača (izvor:autor)

Programski kod 2 prikazuje promjenu energije na indikatoru. Objektom stats instanciramo novi objekt klase PlayerStats kako bi pristupio varijablama. Funkcija Start pokreće funkciju stats.Init, te inicijalizira vrijednost indikatora kao maksimalnu energiju koju igrač može imati. Funkcija Update detektira promjenu svakog okvira, pa tako dođe do promjene na indikatoru energije kada protivnik napadne igrača.

```
private Status statusIndicator;


void Start()
{
    stats.Init();
    if (statusIndicator == null)
    {
        Debug.LogError("NO status indicator on player");
    }
    else
    {
        statusIndicator.SetHealth(stats.curHealth, stats.maxHealth);
    }
}

void Update()
{
    statusIndicator.SetHealth(stats.curHealth, stats.maxHealth);
}
```

Programski kod 2. Promjena energije na indikatoru (izvor:autor)

Programskim kodom 3 prikazano je ranjavanje igrače. Funkcija DamagePlayer detektira napade protivnika, te se trenutna energija smanjuje za iznos varijable damage. Ako vrijednost energije dođe na 0 aktivira se skripta GameMaster čija funkcija KillPlayer ubija protivnika.

```
public void DamagePlayer (int damage)
{
    stats.curHealth -= damage;
    if (stats.curHealth<= 0)
    {
        GameMaster.KillPlayer(this);
    }
    statusIndicator.SetHealth(stats.curHealth, stats.maxHealth);
}
```

Programski kod 3. Ranjavanje igrača (izvor:autor)

Sudar igrača s okolinom je prikazan Programskim kodom 4. Funkcija OnCollisionEnter2D detektira dodir igrača i okoline. Ako igrač dodirne objekt s nazivom health(Clone) trenutna energija postaje maksimalna. Ako igrač dodirne objekt s nazivom ammo(Clone) vrijednost varijable snaga pucnja se povećava za 10. Ako igrač dodirne objekt s nazivom life(Clone) broj života igrača se povećava za 1.

```
void OnCollisionEnter2D(Collision2D collision)
{

    if (collision.gameObject.tag != "Player")
    {
        if (collision.gameObject.name == "health(Clone)")
        {
            stats.curHealth = stats.maxHealth;
            Debug.Log(stats.curHealth);
            Destroy(collision.gameObject);
        }
        if (collision.gameObject.name == "ammo(Clone)")
        {
            Bullet.damage += 10;
            Debug.Log(Bullet.damage);
            Destroy(collision.gameObject);
        }
        if (collision.gameObject.name == "life(Clone)")
        {
            GameOver.lifes += 1;
            Destroy(collision.gameObject);
        }
    }
}
```

Programski kod 4. Sudar igrača sa okolinom (izvor:autor)

Programski kod 5 prikazuje mogućnost ispaljivanja metka. Igrač koristi i skriptu Shoot. Skripta služi za ispaljivanje metaka. Funkcijom se inicijalizira zvuk koji će se čuti za vrijeme pucnja. Funkcijom Shooot se inicijalizira objekt koji će izletjeti za vrijeme pucnjave. Rigidbody2D komponentom se izbacuje objekt u zadanom smjeru. Pokreće se zvuk, te objekt nestaje za 1 sekundu. Na pritisak lijevog klika miša se aktivira funkcija Shoot, te se ispaljuje metak.

```
public class Shoot : MonoBehaviour
{
    public int Damage = 10;
    public Transform firePoint;
    public GameObject bulletPrefab;
    public static float bulletForce = 100f;
    public AudioSource audioData;

void Start()
    {
        audioData = GetComponent<AudioSource>();
    }
    0 references
    void Update()
    {
        if (Input.GetButtonDown("Fire1"))
        {
            Shooot();

        }
    }

    void Shooot()
    {
        GameObject bullet = Instantiate(bulletPrefab, firePoint.position, firePoint.rotation);
        Rigidbody2D rb = bullet.GetComponent<Rigidbody2D>();
        rb.AddForce(firePoint.up * bulletForce, ForceMode2D.Impulse);
        audioData.Play();
        Destroy(bullet, 1);
    }
}
```

Programski kod 5. Ispaljivanje metka (izvor:autor)

Programski kod 6 opisuje sudar metka s okolinom. Skripta Bullet se nalazi na objektu metka. Uz pomoć funkcije OnTriggerEnter2D detektira kontakt s ostalim objektima koje metak pogodi. Metak ignorira dodir s igračem i pomoći koje se stvaraju u igri, te smanjuje protivniku energiju za vrijednost varijable damage, te se uništava.

```
public class Bullet : MonoBehaviour
{
    public static int damage = 10;
    GameObject target;

    void OnTriggerEnter2D(Collider2D collision)
    {
        Debug.Log(collision.name);
        target = GameObject.FindWithTag("Player");
        if (collision.gameObject.tag != target.tag && collision.gameObject.tag != "Help")
        {
            Enemy enemy = collision.GetComponent<Enemy>();
            if (enemy != null)
            {
                enemy.DamageEnemy(damage);
            }
            Destroy(gameObject);
        }

    }
}
```

Programski kod 6. Sudar metka sa okolinom (izvor:autor)

Programski kod 7 donosi mogućnost kretanja igrača. Uz pomoć funkcije Update se mijenjaju vrijednosti pozicije igrača pritiskom tipke na tipkovnici kako bi se kretao po x i y osi, te vrijednost pozicije pokazivača miša koji omogućuje okretanje igrača. Korištenjem FixedUpdate uz pomoć komponente Rigidbody2D se mijenja položaj igrača, te se uz pomoć matematičkih funkcija igrač rotira pomicanjem pokazivača miša.

```
public class Movement : MonoBehaviour
{
    public float moveSpeed = 5f;
    public Rigidbody2D rb;
    public Camera cam;
    Vector2 movement;
    Vector2 mousePos;

    void Update()
    {
        movement.x = Input.GetAxisRaw("Horizontal");
        movement.y = Input.GetAxisRaw("Vertical");
        mousePos = cam.ScreenToWorldPoint(Input.mousePosition);


    }

    void Awake()
    {
        if (cam == null)
        {
            cam = FindObjectOfType<Camera>();
        }
    }

    void FixedUpdate()
    {
        rb.MovePosition(rb.position + movement * moveSpeed * Time.fixedDeltaTime * 2);

        Vector2 lookDir = mousePos - rb.position;

        float angle = Mathf.Atan2(lookDir.y,lookDir.x)* Mathf.Rad2Deg - 90f;
        rb.rotation = angle;
    }
}
```

Programski kod 7. Kretanje igrača (izvor:autor)

Programski kod 8 donosi mogućnost postavljanja indikatora energije. U inspektoru se dodaje izgled indikatora, te vrijednost energije. Funkcijom SetHealth je prikazano koliko je trenutna energija od maksimalne moguće kod igrača i protivnika.

```
public class Status : MonoBehaviour
{
    [SerializeField]
    private RectTransform health;
    [SerializeField]
    private Text healthText;


    void Start()
    {
        if (health == null)
        {
            Debug.LogError("No health bar");
        }
        if (healthText == null)
        {
            Debug.LogError("No health bar");
        }
    }

    public void SetHealth(int _cur, int _max)
    {
        float _value = (float)_cur / _max;
        health.localScale = new Vector3(_value, health.localScale.y, health.localScale.z);
        healthText.text = _cur + "/" + _max + "HP";
    }

}
```

Programski kod 8. Postavljanje indikatora energije (izvor:autor)

4.3. Protivnik

Na slici 11,12 i 13 su prikazani protivnici koji se stvaraju i napadaju igrača. Napadom na igrača protivnici se uništavaju, te oduzimaju energiju igraču. Koristi Rigidbody2D i BoxCollider2D kao kod igrača.

Slika 11. Protivnik 1 [18]

Slika 12. Protivnik 2 [19]

Slika 13. Protivnik 3 [20]

Programski kod 9 donosi svojstva protivnika. Sadrži EnemyStats klasu koja sadrži varijable u kojima se nalazi maksimalna energija koju protivnik može imati i trenutna energija koja se smanjuje kada ga igrač napadne. Varijabla damage sadrži podatak koliko će energije uzeti igraču napad protivnika. Funkcijom Start se inicijalizira Rigidbody2D komponenta koja je potrebna za kretanje protivnika, te se inicijalizira vrijednost indikatora kao kod igrača.

```
public class EnemyStats
{
    public int maxHealth = 100;

    private int _curHealth;

    public int curHealth
    {
        get { return _curHealth; }
        set { _curHealth = Mathf.Clamp(value, 0, maxHealth); }
    }
    public int damage = 20;

    public void Init()
    {
        curHealth = maxHealth;
    }
}
public EnemyStats stats = new EnemyStats();
```

Programski kod 9. Svojstva protivnika (izvor:autor)

Programski kod 10 donosi mogućnost kretnje protivnika prema igraču. Funkcijom Update se traži igrač uz pomoć varijable search ako igrač nije inicijaliziran. Uz pomoć matematičkih funkcija se traži putanja po kojoj bi se protivnik trebao kretati prema igraču. Uz pomoć funkcije FixedUpdate se pokreće funkcija moveCharacter s kojom se protivnik kreće prema igraču.

```
void Update()
{

      if (player == null)
    {
        search = GameObject.FindGameObjectWithTag("Player");
        if (search != null)
        {
            player = search.transform;
        }
    }
    Vector3 direction = player.position - transform.position;
    float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;
    rb.rotation = angle;
    direction.Normalize();
    movement = direction;
}

private void FixedUpdate()
{
    moveCharacter(movement);
}

void moveCharacter(Vector2 direction)
{
    rb.MovePosition((Vector2)transform.position + (direction * moveSpeed * Time.deltaTime));
}
```

Programski kod 10. Kretnja protivnika prema igraču (izvor:autor)

Programski kod 11 donosi mogućnost ranjavanja protivnika. Funkcija DamageEnemy smanjuje vrijednost energije protivniku. Ako vrijednost energije dođe do 0, aktivira se skripta GameMaster čija funkcija KillEnemy ubija protivnika.

```
public void DamageEnemy(int damage)
{
    stats.curHealth -= damage;
    if (stats.curHealth <= 0)
    {
        GameMaster.KillEnemy(this);
    }
    if (statusIndicator != null)
    {
        statusIndicator.SetHealth(stats.curHealth, stats.maxHealth);
    }
}
```

Programski kod 11. Ranjavanje protivnika (izvor:autor)

Programski kod 12 donosi mogućnost sudara protivnika i okoline.Funkcija OnCollisionEnter2D omogućuje detektiranje protivnika sa okolinom. Ako protivnik dodirne igrača, igraču se smanjuje energija, te se protivnik uništi. Ako protivnik dodirne objekt s oznakom Help uništava taj objekt. Ako protivnik dodirne objekt s oznakom Wall ignorira njegovu komponentu Collider2D kako bi prošao kroz njega.

```
void OnCollisionEnter2D(Collision2D collision)
{
    Player _player = collision.collider.GetComponent<Player>();


    if (_player != null)
    {
        _player.DamagePlayer(stats.damage);
        DamageEnemy(1000);

    }
    if (collision.gameObject.tag== "Help")
    {
        Destroy(collision.gameObject);
    }
    if (collision.gameObject.tag == "Wall")
    {
        Physics2D.IgnoreCollision(collision.gameObject.GetComponent<Collider2D>(), GetComponent<Collider2D>());
    }

}
```

Programski kod 12. Sudar protivnika sa okolinom (izvor:autor)

4.4. Stvaranje igrača

Programski kod 13 donosi mogućnost ponovnog kreiranja igrača. Nakon što igrač izgubi svu energiju stvara se ponovno uz pomoć skripte Game Master. Funkcijom RespawnPlayer se pokreće ponovno scenu Levela gdje se inicijalizira igrač na mjesto koje smo odredili varijablom spawnPoint. Pošto se funkcija aktivira kada igrač izgubi svu energiju smanjujemo broj života za 1 i ponovno definiramo snagu metka.

```
public void RespawnPlayer()
{
    Application.LoadLevel(Application.loadedLevel);
    Instantiate(playerPrefab, spawnPoint.position, spawnPoint.rotation);
    GameOver.lifes -= 1;
    Bullet.damage = 10;

}
```

Programski kod 13. Ponovno kreiranje igrača (izvor:autor)

Programski kod 14 donosi mogućnost umiranja igrača i protivnika. Funkcijom KillPlayer se uništava objekt igrača, te se aktivira funkciju za ponovno stvaranje igrača. Funkcijom KillEnemy uništavamo objekt protivnika, te povećavamo vrijednost bodova za 10.

```
public static void KillPlayer (Player player)
{
    Destroy(player.gameObject);
    gm.RespawnPlayer();
}

public static void KillEnemy(Enemy enemy)
{
    Destroy(enemy.gameObject);
    Score.scoreValue += 10;
}
```

Programski kod 14. Umiranje igrača i protivnika (izvor:autor)

Programski kod 15 donosi mogućnost nastavka igre. Kako se ne bi odmah igra nastavila, pokreće se odbrojavanje 3 sekunde za nastavak igre uz pomoć IEnumeratora CountDownStart. Time.timeScale=0f zaustavlja igru, a Time.timeScale=1f nastavlja igru. Uz pomoć petlje na ekranu se prikazuje odbrojavanje 3 sekunde, te se isključuje platno (eng. Canvas) uz pomoć funkcije SetActive(false). Pomoću funkcije Start se pokreće IEnumerator uz pomoć funkcije StartCoroutine(CountDownStart()).

```
void Start()
{
    StartCoroutine(CountDownStart());
    if (gm == null)
    {
        gm = GameObject.FindGameObjectWithTag("GM").GetComponent<GameMaster>();
    }
}

void Update()
{
    if(playerPrefab == null)
    {
        playerPrefab = GameObject.FindGameObjectWithTag("Player").transform;
    }
}

 IEnumerator CountDownStart()
{
    Time.timeScale = 0f;
    while (count > 0)
    {

        countText.text = count.ToString();
        yield return new WaitForSecondsRealtime(1f);
        count--;


    }
    countText.text = "Go";

    yield return new WaitForSecondsRealtime(1f);

    canvas.SetActive(false);
    Time.timeScale = 1f;

}
```

Programski kod 15. Nastavak igre (izvor:autor)

4.5. Stvaranje protivnika

Programski kod 16 donosi mogućnost stvaranja protivnika i pomoći na mapi. Novi protivnici se stvaraju uz pomoć skripte Spawn. Polje enemy sadrži 3 objekta protivnika koji se dodaju u inspektoru, te u polje helpItems 3 objekta pomoći. Funkcijom Update u varijablu random se sprema nasumični broj, te u Vector2 objekt se sprema lokacija gdje će se protivnik, te pomoć stvoriti na mapi. Uz pomoć Time.time se stvaraju novi protivnici za svaku vrijednos spawnRate, te se vrijednost smanjuje za 0.5 sekundi svaki puta kada se stvore novi protivnici, dok ne dođe do vrijednosti 2.5 sekundi. Funkcijom Instantiate se stvara protivnik, te pomoć na zadanoj lokaciji na mapi.

```
void Update()
{

    random = Random.Range(0, 3);
    Vector2 whereToSpawn1 = new Vector2(Random.Range(-46.3f, 46.2f), 35f);
    Vector2 whereToSpawn2 = new Vector2(Random.Range(-46.3f, 46.2f), -32f);
    Vector2 whereToSpawn3 = new Vector2(-54f, Random.Range(34f, -27.2f));
    Vector2 whereToSpawn4 = new Vector2(Random.Range(-42.7f, 40.7f), Random.Range(25.9f, -20.7f));
    if (Time.time > nextSpawn)
    {

        nextSpawn = Time.time + spawnRate;
        spawnRate -= 0.5f;
        Instantiate(enemy[random], whereToSpawn1, Quaternion.identity);
        Instantiate(enemy[random], whereToSpawn2, Quaternion.identity);
        Instantiate(enemy[random], whereToSpawn3, Quaternion.identity);
        if(spawnRate == 2)
        {
            spawnRate += 0.5f;

        }
    }
    if (Time.time > nextHelp)
    {
        nextHelp = Time.time + spawnHelp;
        Instantiate(helpItems[random], whereToSpawn4, Quaternion.identity);
    }
}
```

Programski kod 16. Stvaranje protivnika i pomoći (izvor:autor)

4.6. Rezultat

Programski kod 17 omogućuje spremanje bodova u datoteku. Uz pomoć skripte GameOver se spremaju bodovi koje je igrač skupio u datoteku score.txt na disku.U funkciji Save se definira putanja gdje će se spremiti datoteka u koju spremamo podatke. Uz pomoć funkcija WriteAllText i AppendAllText se spremaju bodovi koje je igrač skupio. Funkcijom Update provjerava ako je broj života jednak 0, onda se pokreće funkcija Save kako bi se podaci spremili.

```
void Update()
{
    life.text = "Life: " + lifes;
    bulletDmg.text = "Bullet damage: " + Bullet.damage;


    if (lifes == 0)
    {
        Save();
        lifes = 3;
        Score.scoreValue = 0;
        SceneManager.LoadScene("Credits");
    }
}

public void Save()
{
    string path = Application.dataPath + "/score.txt";
    if (!File.Exists(path))
    {
        File.WriteAllText(path, "");
    }
    string content = "Score :" + Score.scoreValue.ToString() + "\n";
    File.AppendAllText(path, content);
    Debug.Log(Application.dataPath);
}
```

Programski kod 17. Spremanje bodova u datoteku (izvor:autor)

Programski kod 18 donosi mogućnost prikazivanja bodova iz datoteke. Skriptom SaveScore prikazujemo bodove iz datoteke. U inspektoru se dodaje Text objekt koji će prikazivati podatke iz datoteke u varijabli score. Funcijom Load se pročitaju podatci s putanje na kojem se nalazi datoteka gdje su spremljeni bodovi igrača, te ih uz pomoć petlje dodaju u varijablu score. Funkcijom Start se inicijalizira putanju gdje se nalazi datoteka, te e pokreće funkcija Load.

```
public class SaveScore : MonoBehaviour
{
    string[] data;
    string path;
    public Text score;

    void Start()
    {
        path = Application.dataPath + "/score.txt";
        Load();
    }

    public  void Load()
    {
        data = File.ReadAllLines(path);
        foreach (string line in data)
        {
            score.text += line +"\n";
        }
    }
}
```

Programski kod 18. Prikazivanje bodova iz datoteke (izvor:autor)

4.7. Scena

Programski kod 19 donosi mogućnost pristupanja scenama. Scenama se upravlja uz pomoć skripte MainMenu. Uz pomoć funkcije Update se dodaje mogućnost igraču izlazak u glavni meni uz pomoć ESC tipke na tipkovnici. Funkcija PlayGame otvara scenu Level1 koja pokreće igru. Funkcija Menu vodi igrača na scenu Menu, koja prikazuje glavni izbornik u igri. Funkcija HowToPlay vodi igrača do scene HowToPlay gdje je prikazan video s kontrolama i što sve očekuje igrača u igri.
Funkcija Credits vodi igrača na scenu Credits, gdje su prikazani podaci o autoru igre. Funkcija Exit omogućuje izlazak iz igre.

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        SceneManager.LoadScene("Menu");
    }
}

public void PlayGame()
{
    SceneManager.LoadScene("Level1");
}

public void Menu()
{
    SceneManager.LoadScene("Menu");
}

public void HowToPlay()
{
    SceneManager.LoadScene("HowToPlay");
}

public void Credits()
{
    SceneManager.LoadScene("Credits");
}

public void Score()
{
    SceneManager.LoadScene("Scores");
}

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

Programski kod 19. Pristupanje scenama (izvor:autor)

Programski kod 20 donosi mogućnost pauziranje igre. Skripta PauseGame omogućuje pauziranje igre i otvaranje izbornika gdje se odabire nastavak igre, izlazak u glavni izbornik ili izlazak iz igre. Funkcija Resume omogućuje nastavak igre, dok funkcija Pause pauzira igru. ESC tipka omogućuje navedenu funkcionalnost.

```
void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        if (GameIsPaused)
        {
            Resume();

        }
        else
        {
            Pause();
        }
    }
}

public void Resume()
{
    pauseMenuUI.SetActive(false);
    Time.timeScale = 1f;
    GameIsPaused = false;
}

void Pause()
{
    pauseMenuUI.SetActive(true);
    Time.timeScale = 0f;
    GameIsPaused = true;
}
```

Programski kod 20. Pauziranje igre (izvor:autor)

5. Zaključak

U ovom je radu opisan način izrade 2D igre u alatu Unity. Unity je jako moćan alat s kojim se mogu kreirati igre za mnoge platforme, što smanjuje vrijeme izrade igre ako želimo igru izdati na više platformi. Spomenuti alat omogućuje korištenje skripti u programskim jezicima C# i JavaScript. Mogućnosti alata možemo istražiti kroz službenu dokumentaciju, koja je opsežna i dobro dokumentirana. Osim dokumentacije postoji velik broj video uradaka vezanih za korištenje alata koji pomažu da se svladavaju prepreke koje se pojave u procesu izrade igre.

Unity alat nije jedini besplatni alat za kreiranje igara, no dovoljno je jednostavan i sadrži dovoljno materijala kako bi početnici mogli kreirati svoju prvu igru, kao što je prikazano u ovom radu. Napretkom tehnologija kao što su virtualna, proširena i produžena stvarnost, igre će postati kreativnije i zanimljivije za igrače, stoga je poželjno znati koristiti alat koji omogućuje jednostavniju implementaciju takvih tehnologija, kao što je alat Unity.

Iako je alat jednostavan za korištenje, postupak izrade igre zahtjeva puno vremena i truda, kako bi kreirali funkcionalnosti koje želimo u igri. Izbor funkcionalnosti unutar Unity alata je velik jer se svakom novom verzijom broj funkcionalnosti povećavao. Tvorcima ovog alata je upravo to bio prvenstveni cilj, stvoriti alat s mnoštvom funkcionalnosti koje ostali alati ne pružaju. Kako bi kreirali modele i okolinu u igri potrebno je koristiti alate za obradu slike kao što su Photoshop i Gimp.

Korištenje alata Unity za kreiranje igre je jako zanimljivo i zabavno, no ujedno i zahtjevno jer osim programerskih vještina zahtjeva dizajnerske vještine za kreiranje modela i okoliša. Kreirana igra je funkcionalna, no zahtjeva još puno vremena kako bi se implementirale sve zamišljene funkcionalnosti. Nadogradnja ove igre bi se trebala temeljiti na novim modelima, mapama, animacijama. Nove razine i funkcionalnosti kao što su bolja naoružanja, te različite težine igranja čime bi igra postala zanimljivija igračima.

6. Literatura

1. Peckham, E., How Unity built the worlds most popular game engine, [website], 2019., https://techcrunch.com/2019/10/17/how-unity-built-the-worlds-most-popular-game-engine/?guccounter=1 (pristupljeno 13.7.2020.)
2. Cohen, P., Unity 2.0. game engine now available, [website], 2007., https://www.macworld.com/article/1060484/unity.html (pristupljeno 15.7.2020.)
3. Girard, D., Unity 3 brings very expensive dev tools at a very low price, [website], 2010., https://arstechnica.com/information-technology/2010/09/unity-3-brings-very-expensive-dev-tools-at-a-very-low-price/?comments=1 (pristupljeno 18.7.2020.)
4. Tach, D., unity 4.0. available for download today with DX 11 support and Linux preview, [website], 2012., https://www.polygon.com/2012/11/14/3645122/unity-4-0-available-download (pristupljeno 18.7.2020.)
5. Kumparak, G., Unity 5 Announced With Better Lighting, Better Audio, And “Early“ Support For Plugin-Free BrowservGames, [website], 2014., https://consent.yahoo.com/v2/collectConsent?sessionId=3_cc-session_b27aef01-0003-4038-be71-ec8cb0f18e69 (pristupljeno 22.07.2020.)
6. Batchelor, J., Unity dropping major updates in favour of date-based model, [website], 2016. https://www.gamesindustry.biz/articles/2016-12-14-unity-dropping-major-updates-in-favour-of-date-based-model (pristupljeno 28.7.2020.)
7. Batchelor, J., Unity 2018 detalied in GDC keynote, [website], 2018. https://www.gamesindustry.biz/articles/2018-03-20-unity-2018-detailed-in-gdc-keynote (pristupljeno 28.07.2020.)
8. Krogh-Jacobsen, T., Introducing Unity 2019.1, [website], 2019., https://blogs.unity3d.com/2019/04/16/introducing-unity-2019-1/?fbclid=IwAR2v-RMQv_wc3jWrS8_zcqKW6UujvFPbFW5bYgYI_6UP0H-hkfOXaGTWKyw (pristupljeno 29.7.2020.)
9. Unity Technologies, Unity's interface, [website], 2020., https://docs.unity3d.com/Manual/UsingTheEditor.html (pristupljeno 2.8.2020.)

10. Unity Technologies, MonoBehaviour.Start(), [website], 2020., https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html (pristupljeno 5.8.2020.)
11. Unity Technologies, Event Functions, [website], 2020., https://docs.unity3d.com/Manual/EventFunctions.html (pristupljeno 14.8.2020.)
12. Unity Technologies, GameObjects, [website], 2020., https://docs.unity3d.com/Manual/GameObjects.html (pristupljeno 18.8.2020.)
13. Unity Technologies, Rigidbody 2D, [website], 2020., https://docs.unity3d.com/Manual/class-Rigidbody2D.html (pristupljeno 20.8.2020.)
14. Unity Technologies, Box Collider 2D, [website], 2020., https://docs.unity3d.com/Manual/class-BoxCollider2D.html (pristupljeno 2.9.2020.)
15. Unity Technologies, Audio Source, [website], 2020., https://docs.unity3d.com/Manual/class-AudioSource.html (pristupljeno 3.9.2020.)
16. Unity Technologies, Prefabs, [website], 2020., https://docs.unity3d.com/Manual/Prefabs.html (pristupljeno 3.9.2020.)
17. Pngfind [online fotografija], https://www.pngfind.com/pngs/m/52-523806_preview-top-down-tank-png-transparent-png.png (pristupljeno: 14.9.2020)
18. Pikpng [online fotografija], https://www.pikpng.com/pngl/m/215-2154656_nuclear-bomb-clipart-png-transparent-png.png (pristupljeno: 14.9.2020)
19. K30 [online fotografija], https://k30.kn3.net/taringa/F/2/9/1/4/C/Shioo4Play/83A.png (pristupljeno: 14.9.2020)
20. Kissclipart [online fotografija], https://library.kissclipart.com/20181123/rae/kissclipart-cherry-bomb-explosion-clipart-cherry-bomb-0eb45b8a456f0e43.jpg (pristupljeno: 14.9.2020)

Popis slika

Programski kod