Izrada 2D edukativne igre Bookworm u Unity alatu

Makaj, Jasmin

Sveučilište u Rijeci – Odjel za informatiku

Preddiplomski jednopredmetni studij informatike

Jasmin Makaj

Bookworm

Završni rad

Mentorica: izv. prof. dr. sc. Marina Ivašić-Kos

Rijeka, rujan 2019.

Sažetak

U ovom radu prikazan je proces složene 2D edukativne igre u Unity alatu. Igrač u dva moda igranja, Avantura i Utrka s vremenom, mora pronalaziti i sastavljati riječi od zadanih slova. Pritom igrač sakuplja iskustvo te novac koji može razmjeniti za razne napitke koji mu pomažu u igranju. Cilj je završiti Avantura mod igranja i sakupiti sve medalje. Igra se sastoji od tri scene, a to su: MainMenu, BattleScene i TimeAttack. Iz svake od navedenih scena izdvojene su najbitnije stvari te pojašnjene u detalje.

Ključne riječi

Unity, igra, igrač, skripta, neprijatelj, C#, animator, animacije, potez, utrka s vremenom, avantura

Sadržaj

Uvod 1
Unity 2
  Unity sučelje 2
Ideja i opis igre 5
Izrada igre 7
  Igra 7
  Avantura mod 10
  Neprijatelj 15
  Animiranje 20
  Animator 21
  Igrač 25
  Utrka s vremenom 28
  Medalje 33
  Opcije 34
Zaključak 36
Bibliografski navod 37
Popis slika 40
Popis skripti 41
Popis priloga 42

Uvod

Igre su, kao zabavan proizvod, popularne kod svih uzrasta te u današnjem modernom svijetu postanu dio života gotovo svakog djeteta. Gaming industrija kreće od izrade prve igre, nazvane Pong [1], koja je napravljena 1972. godine, a radi se o arkadnoj verziji ping ponga. Od tada, gaming industrija neprekidno raste. Prema procjenama techjury stranice [2], gaming industrija je 2018. godine generirala vrtoglavih 135 milijardi američkih dolara.

Edukativne 2D igre jedne su od rasprostranjenijih žanrova igara na internetu. Poznate su po jednostavnoj grafici i animacijama, zabavnim zvukovima i vizualnim efektima i lako razumljivim sučeljem. Cilj takvih igara je potaknuti igrača na razmišljanje, ali je ujedno i bitno da se igrač zabavlja.

U ovom završnom radu prikazan je detaljniji proces izrade takve vrste 2D igre uz pomoć Unity alata te Adobe Photoshopa i Auditiona. Igra je namjenjena za jednog igrača u kojoj mu je cilj pronalaziti i sastavljati gramatički točne riječi hrvatskog jezika. Primjerice, sastavljanjem riječi „Svijet", igrač uči da se riječ piše sa „ije", a ne sa „je". Igrač u dva moda igranja pokazuje svoje znanje hrvatskog jezika te njima pobjeđuje neprijatelje i sakuplja novac i iskustvo.

Igra je prilagođena za Windows [3], Linux [4] i Mac OS [5] operacijske sustave. Skripte, koje su pisane u C# programskom jeziku [6] u Visual Studio alatu [7], obuhvaćaju sve dijelove igre, kao što su: grafika [8], animacije, zvukovi, te ih povezuju u funkcionalnu cijelinu. Zvukovi, kao i grafika kojom su postignute animacije, preuzeti su sa različitih izvora. Neke grafičke elemente poput loga i ikone igre napravljeni su u Adobe Photoshop alatu.

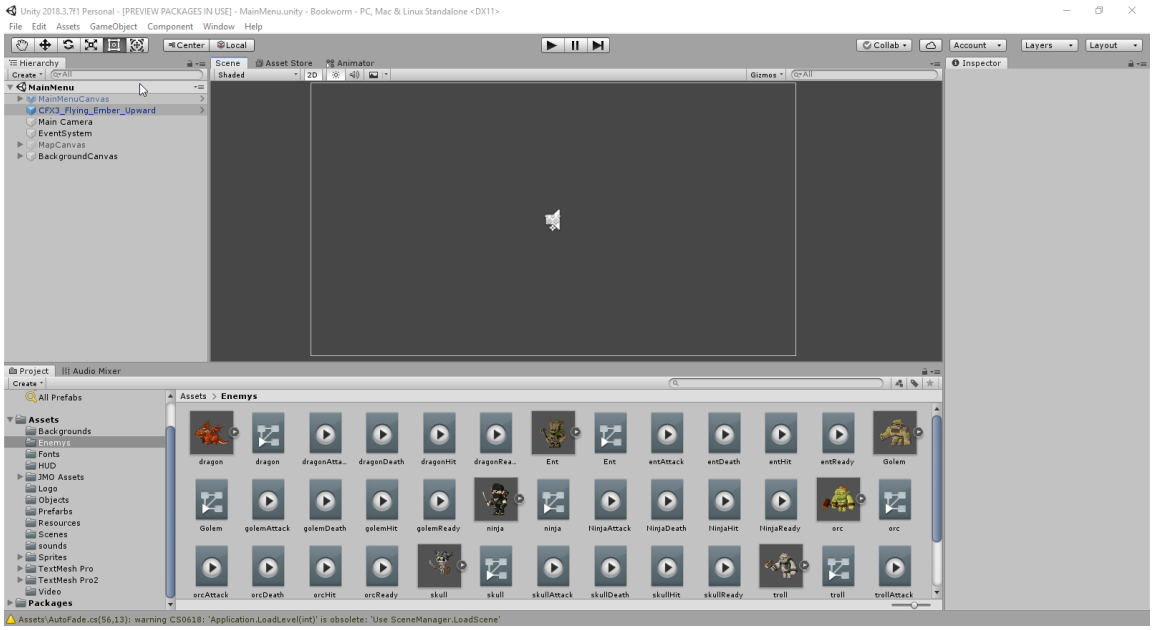
Motiv izrade ove igre proizlazi iz želje za izradom igre iz djetinstva slične tematike, koja se zove Bookworm Delux Adventures [9]. Također je bio cilj proširiti znanje, ne samo iz Unity alata, nego i drugih alata kojima je igra napravljena. Novostečeno znanje i iskustvo možemo primjeniti za buduće slične projekte.

Unity

Unity game engine prvi puta je objavljen 2005. godine na konferenciji Worldwide Developers tvrtke Apple Inc te je tada bio prilagođen samo za izradu igara za platformu OS X [10]. Unity game engine se od tada svakodnevno unaprjeđuje te od 2018. godine podržava izradu igara za više od 25 platformi [11]. Unity korisniku omogućuje pregledniju i organiziraniju izradu igre 2D ili 3D tipa, kao i mogućnost testiranja igre unutar okruženja. U početku se za kodiranje unutar Unity programa koristio programski jezik Boo [12], nakratko i JavaScript [13], ali su oba uklonjena. Nakon toga C# postaje osnovni programski jezik te ga Unity, za uređivanje koda, automatski otvara u Visual Studio alatu. Većina programa za izradu 2D grafike ili 3D modela je također podržana unutar Unity okruženja. Neki od poznatijih programa su Adobe Photoshop te Animate, Blender, Maya, 3DS Max.

Unity sučelje

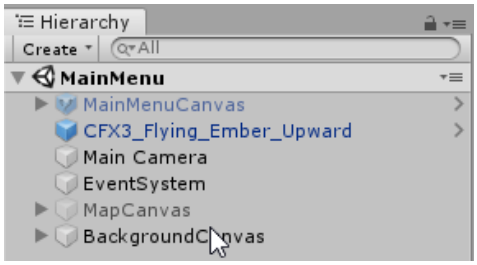
Prilikom prvog pokretanja Unity sučelje je poprilično jednostavno. Slika 1. prikazuje sučelje Unitya. Najbitniji prozor projekt (engl. *Project*), nalazi se na dnu programa te uz pomoć njega lakše pregledavamo sve elemente koje smo dodali u projekt. Elementi projekta mogu biti skripte, zvukovi, animacije, 2D grafika, fontovi i slično.



Slika 1. Unity sučelje

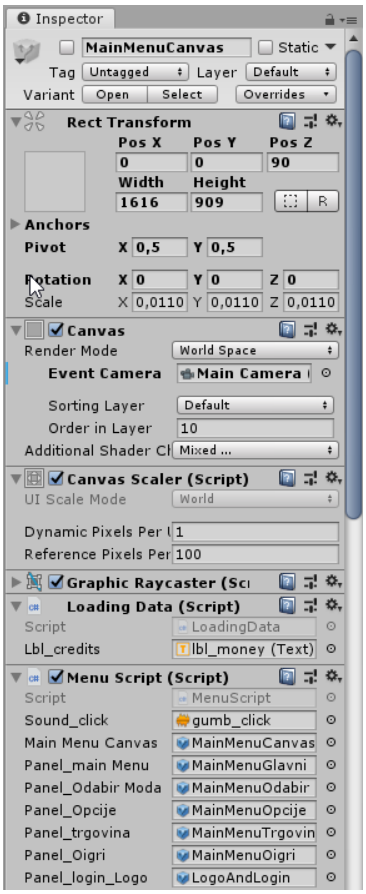
Elemente dodajemo klikom miša na željeni element te povlačenje samog na scenu (engl. *Scene*) [14] ili u prozor hijerarhije (engl. *Hierarchy*) [15] te njih onda nazivamo objektima igre (engl. *GameObject*) [16]. Prozor

hijerarhije olakšava samo navigiranje kroz objekte koji su pridodani sceni. Na Slici 2. je prikazan primjer hijerarhijskog prikaza MainMenu scene.



Slika 2. Hijerarhijski prikaz objekata

Klikom na objekt na sceni, sa desne strane, otvara se prozor inspektora (engl. *Inspector*) [17] koji je prikazan na Slici 3. Na prozoru inspektora vide se sva svojstva (engl. *Properties*) [18] odnosno komponente (engl. *Components*) odabranog objekta. Mogu se i dodavati nove komponente (engl. *Add Components*), uređivati te brisati komponente kao što su skripte, zvukovi, animacije.



Slika 3. Inspector

Iznad prozora scene nalaze se tri gumba (Slika 4), koja služe za pokretanje i testiranje igre.

Slika 4. Play, Pause i Step gumbovi

Pritiskom na gumb Pokreni (engl. *Play*) otvara se novi prozor Igra (engl. *Game*) [19] koji simulira igru i prikazuje developeru kako će izgledati završni projekt. Slika 5. prikazuje prozor nepokrenute igre, odnosno prozor scene. Igru nakon pokretanja možemo i zaustaviti (engl. Pause) te uz pomoć gumba Korak (engl. *Step*) možemo se kroz igru kretati korak po korak (engl. *Frame*).

Slika 5. Game prozor

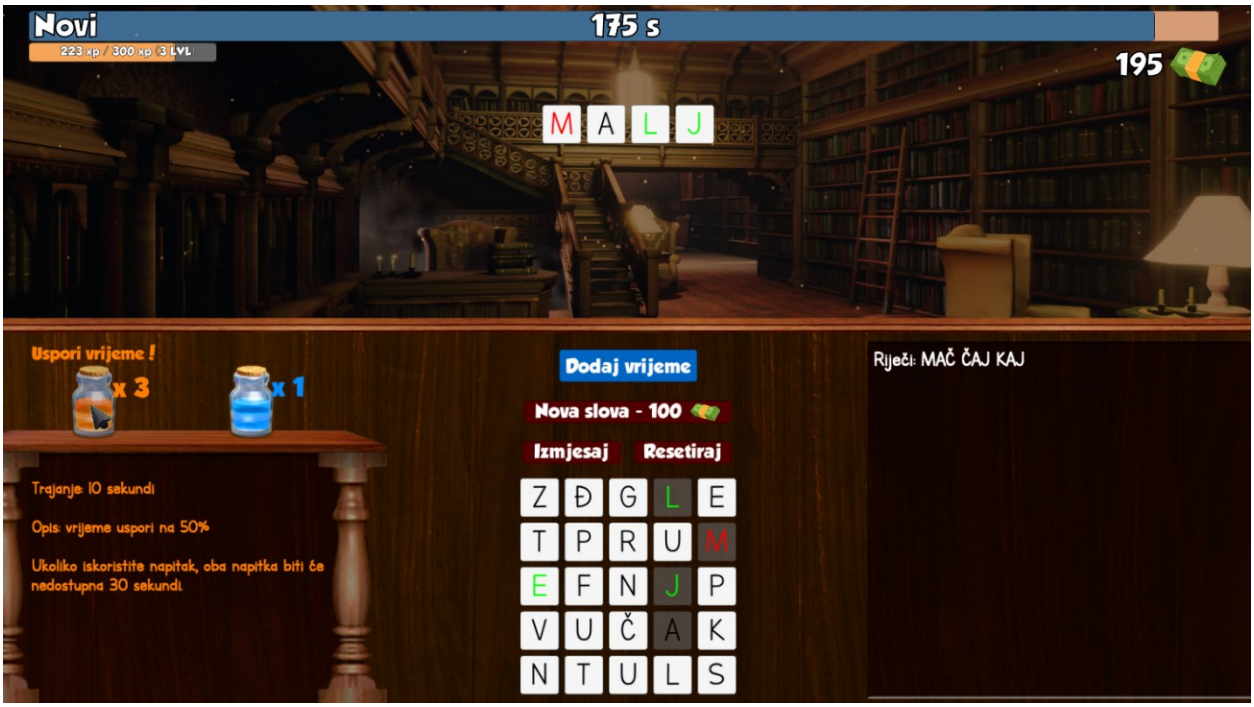
Iznad hijerarhijskog prikaza projekta nalazi se alatna traka s gumbovima za manipuliranje objektima (Slika 6). Njima možemo pomicati, rotirati i skalirati odabrane objekte po sceni, ali i kontrolirati poziciju kamere na sceni.

Slika 6. Alatna traka

Ideja i opis igre

Ideja ovog završnog rada bila je razviti 2D edukativnu igru na hrvatskom jeziku u Unity game enginu. Igra je temeljena na igri developera PopCap Games iz 2006. godine, koja nosi naziv Bookworm Adventures, u kojoj igrač formira riječi engleskog jezika te njima napada neprijatelja. Uzeo sam glavnu funkionalnost te igre i napravio dva moda igranja, Avantura i Utrka s vremenom. U Avantura modu cilj igrača je da kroz 8 razina pobjedi neprijatelja sastavljanjem čim većih riječi. Riječi igrač može sastaviti od 25 automatski generiranih slova, od kojih svako može iskoristiti jednom po potezu. Napadanjem igrač dobiva iskustvo kojim povećata vlastiti nivo (engl. *Level*). Igrač također dobiva i novac koji može potrošiti na napitke koji daju različite stvari poput regeneracije, usporavanje vremena i slično, ili ih može potrošiti na generiranje novih 25 slova. Neprijatelj na isti način napada *playera*. Igrač se, tijekom Avantura moda, može liječiti crvenim napitkom ili ojačavati svoj temeljni napad ispijanjem zelenog napitka. U Utrka s vremenom modu cilj je sakupiti što više različitih riječi koje igraču produžuju vrijeme igranja. Postoji mogućnost iskorištavanja dva različita napitka u tom modu igranja, a to su: narančasti za usporavanje vremena na 10 sekundi i plavi za dodavanje dodatnih 30 sekundi. Igra prati igračev napredak te ga nagrađuje različitim medaljama za određena postignuća. Glavni cilj igrača je završiti Avantura mod te sakupiti sve moguće medalje.

Slika 7. Avantura mod igranja



Slika 8. Mod igranja - Utrka s vremenom

Izrada igre

Igra

Igru čine tri glavne scene: *MainMenu*, *BattleScene* te *TimeAttack*. Prilikom pokretanja igre u pozadini se prikazuje video otvaranja knjige [20] te se po završetku videa ili pritiskom lijeve tipke miša pojavljuje zaslon za prijavu i registraciju.

Pritiskom na gumb prijava skripta *LoadingData* učitava podatke o igraču iz datoteke ime_igrača.txt ili ukoliko ne postoje podaci kreira nove. Ukoliko se izrađuje novi profil, skripta stvara dvije nove tekstualne datoteke o igraču sa zadanim podacima kao što su *username*, *xp*, *level* i slično. Glavna tekstualna datoteka, ime_igrača.txt, prati Avantura mod igrača, otključane razine, trenutno iskustvo igrača, količinu novca, broj napitaka i slično. Kod koji učitava podatke nalazi se u Skripti 1. Funkcija *LoadData,* koja se nalazi u nastavku, pristupa folderu Documents igrača te provjera postoje li datoteke. Ukoliko ne postoje, stvara nove i sprema početne vrijednosti.

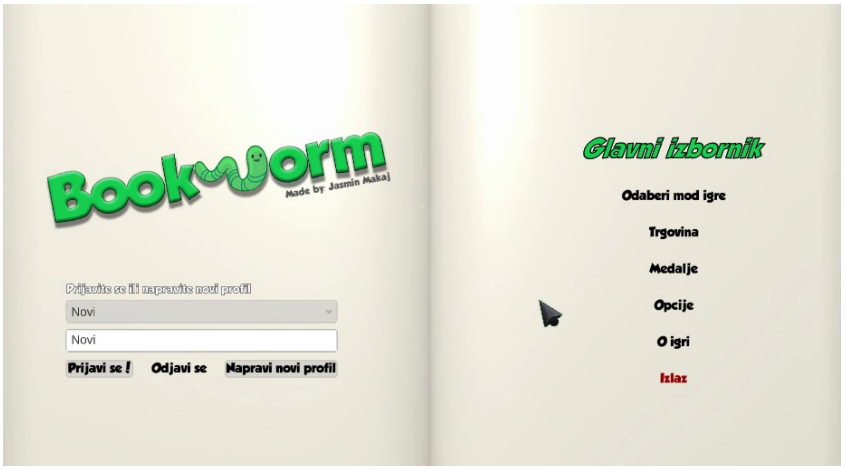
```
    public void LoadData()
    {
        string pathDocuments =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
       public Text lbl_credits;

        PlayerData.username = MenuScript.usernameLogin;
        if (File.Exists(pathDocuments + "\\Bookworm\\" + PlayerData.username + ".txt"))
        {
            var data = File.ReadAllLines(pathDocuments + "\\Bookworm\\" +
PlayerData.username + ".txt");
            var lista = new List<string>(data);
            PlayerData.username = lista[0];
            PlayerData.xp = int.Parse(lista[1]);
            PlayerData.LVL = int.Parse(lista[2]);
            PlayerData.c_health = int.Parse(lista[3]);
            PlayerData.m_health = int.Parse(lista[4]);
            PlayerData.e_c_health = int.Parse(lista[5]);
            PlayerData.e_m_health = int.Parse(lista[6]);
            PlayerData.BaseAttackDmg = int.Parse(lista[7]);
            PlayerData.credits = int.Parse(lista[8]);
            PlayerData.red_potions = int.Parse(lista[9]);
            PlayerData.green_potions = int.Parse(lista[10]);
            PlayerData.orange_potions = int.Parse(lista[11]);
            PlayerData.blue_potions = int.Parse(lista[12]);
            PlayerData.BonusAttack = int.Parse(lista[13]);
            PlayerData.difficulty = int.Parse(lista[14]);
            PlayerData.lvl1 = bool.Parse(lista[15]);
            PlayerData.lvl2 = bool.Parse(lista[16]);
            PlayerData.lvl3 = bool.Parse(lista[17]);
            PlayerData.lvl4 = bool.Parse(lista[18]);
            PlayerData.lvl5 = bool.Parse(lista[19]);
            PlayerData.lvl6 = bool.Parse(lista[20]);
            PlayerData.lvl7 = bool.Parse(lista[21]);
            PlayerData.lvl8 = bool.Parse(lista[22]);
```

```
            lbl_credits.text = PlayerData.credits.ToString();
        }
        else
        {
            PlayerData.username = MenuScript.usernameLogin;
            PlayerData.xp = 0;
            PlayerData.LVL = 1;
            PlayerData.c_health = 100;
            PlayerData.m_health = 100;
            PlayerData.e_c_health = 100;
            PlayerData.e_m_health = 100;
            PlayerData.BaseAttackDmg = 10;
            PlayerData.credits = 100;
            PlayerData.red_potions = 3;
            PlayerData.green_potions = 3;
            PlayerData.orange_potions = 2;
            PlayerData.blue_potions = 1;
            PlayerData.BonusAttack = 0;
            PlayerData.difficulty = 1;
            PlayerData.lvl1 = true;
            PlayerData.lvl2 = false;
            PlayerData.lvl3 = false;
            PlayerData.lvl4 = false;
            PlayerData.lvl5 = false;
            PlayerData.lvl6 = false;
            PlayerData.lvl7 = false;
            PlayerData.lvl8 = false;
            lbl_credits.text = PlayerData.credits.ToString();
            if (!File.Exists(pathDocuments + "\\Bookworm\\" + PlayerData.username +
".txt"))
            {
                using (File.Create(pathDocuments + "\\Bookworm\\" + PlayerData.username +
".txt")) { }
            }
            File.AppendAllText(pathDocuments + "\\Bookworm\\" + PlayerData.username +
".txt", PlayerData.username + "\n" + PlayerData.xp + "\n" + PlayerData.LVL + "\n" +
PlayerData.c_health + "\n" + PlayerData.m_health + "\n" + PlayerData.e_c_health + "\n" +
PlayerData.e_m_health + "\n" +
                PlayerData.BaseAttackDmg + "\n" + PlayerData.credits + "\n" +
PlayerData.red_potions + "\n" + PlayerData.green_potions + "\n" +
PlayerData.orange_potions + "\n" + PlayerData.blue_potions + "\n" +
PlayerData.BonusAttack + "\n" + PlayerData.difficulty + "\n"
                +PlayerData.lvl1 + "\n" + PlayerData.lvl2 + "\n" + PlayerData.lvl3 +
"\n" + PlayerData.lvl4 + "\n" + PlayerData.lvl5 + "\n"
                 + PlayerData.lvl6 + "\n" + PlayerData.lvl7 + "\n" + PlayerData.lvl8);
        }
```

Skripta 1. Kod za učitavanje ili kreiranje podataka igrača

Druga dio te skripte radi gotovo istu stvar, samo što prati postignuća, odnosno zaslužene medalje koje igrač sakuplja tijekom igranja. Program projvjerava postoje li datoteke, ukoliko ne postoje, program stvori datoteku sa zadanim vrijednostima. Postignuća, odnosno medalje, igrač dobiva s obzirom na veličinu riječi, broj pronađenih riječi kroz igranje te broj unesenih riječi u modu igranja Utrka s vremenom i Avantura.

Prijavom program učitava podatke te korisniku prikazuje glavni izbornik koji nudi opcije odabira moda igranja, trgovinu, opcije igre, više o igri te izlaz iz igre. Igraču se također omogućuje pritisak na gumb Odjava. Prikaz glavnog izbornika dan je na Slici 9.

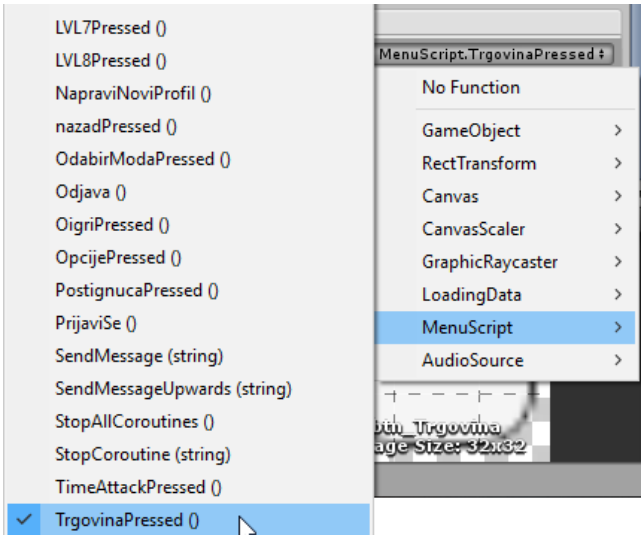


Slika 9. Glavni izbornik

Skripta imena *MenuScript* sadrži sve funkcije koje su potrebne da bi scena ispravno radila. Unutra se nalaze javne fukcije svih gumbova, kao što su odabir moda, kupovanje napitaka, odabir drugog menija. Primjer funkcije gumba za otvaranja trgovine možete vidjeti u Skripti 2. Funkcija sakriva sve nepotrebne prozore te mijenja naslov i pušta zvukovni efekt klika. Navedena funkcija je povezana sa gumbom preko inspektora, kao što je vidljivo na dijelu koda iz MenuScript skripte na Slici 10.

```
public GameObject MainMenuCanvas;
public GameObject panel_mainMenu;
public GameObject panel_OdabirModa;
public GameObject panel_Opcije;
public GameObject panel_trgovina;
public GameObject panel_Oigri;
public GameObject panel_login_Logo;
public GameObject panel_postignuca;
public GameObject panel_mapa;

public void TrgovinaPressed()    {
        this.GetComponent<AudioSource>().Play();
        panel_mainMenu.SetActive(false);
        panel_OdabirModa.SetActive(false);
        panel_Opcije.SetActive(false);
        panel_mapa.SetActive(false);
        panel_trgovina.SetActive(true);
        panel_Oigri.SetActive(false);
        panel_login_Logo.SetActive(false);
        panel_postignuca.SetActive(false);
        lbl_naslov.text = "Trgovina";   }
```

Skripta 2. Pritisak na gumb „Trgovina" u glavnom izborniku



Slika 10. Povezivanje funkcije sa gumbom

Ukoliko igrač odabere mod igranja, iz skripte *AutoFade* [21] poziva se funkcija *LoadLevel* kojom postižemo glatki prijelaz između scena. Funkciju možemo pozvati vrlo jednostavno bilo gdje u igri. Za promjenu scene, funkcija prima ulazne parametre poput naziva scene koju želimo učitati, *fadein* vrijeme, *fadeout* vrijeme te boju pozadine tranzicije. Primjer promjene scene iz *MainMenu* u *BattleScene* uz pomoć skripte možete vidjeti ispod. Funkcija se poziva pritiskom na gumb *levela*.

```
void ScenaAvantura()
    {
        this.GetComponent<AudioSource>().Play();
        AutoFade.LoadLevel("BattleScene",1,1,Color.black);
    }
```

Skripta 3. Funkcija promjene scene

Avantura mod

Ulaskom u avantura mod igranja, inicijalizira se glavna skripta *GameManagment* koja kontrolira sve šta se trenutno događa na sceni. Skripta je vrlo složena i ima puno djelova, stoga ću skratiti i objasniti osnovne dijelove. *GameManagment* skripta kontrolira što se sve na početku treba učitati, što će se dogoditi kada se pritisne gumb za iskoristiti napitak, kada spremati korisničke podatke, kada osvježiti podatke na ekranu te razne male funkcije za izračunavanje napada (*CalculateBaseAttack* funkcija), iskustva, *levela* igrača (*CalculateLVL* funkcija) i prosjeka iskorištenih slova. Također u njoj se nalazi kontrola napada igrača.

Pošto taj mod igranja ima 8 različitih razina u kojima se mijenjaju pozadinske slike, neprijatelji, njihove sposobnosti, život igrača, jačine napada i slično, program mora automatski odrediti takve stvari. Funkciju *Start* [22] možete vidjeti u Skripti 4.

```
void Start()
    {
        LoadBackground();
        LoadEnemy();
        SetHealth();

        gs = GameObject.FindObjectOfType(typeof(GeneratorSlova)) as GeneratorSlova;
        CalculateBaseAttack();
        lbl_name_result.text = PlayerData.username;
        btn_attack.interactable = false;
        gs.GenerirajSlova();
        refreshStats();
        lvl = CalculateLVL(PlayerData.xp);
    }
```

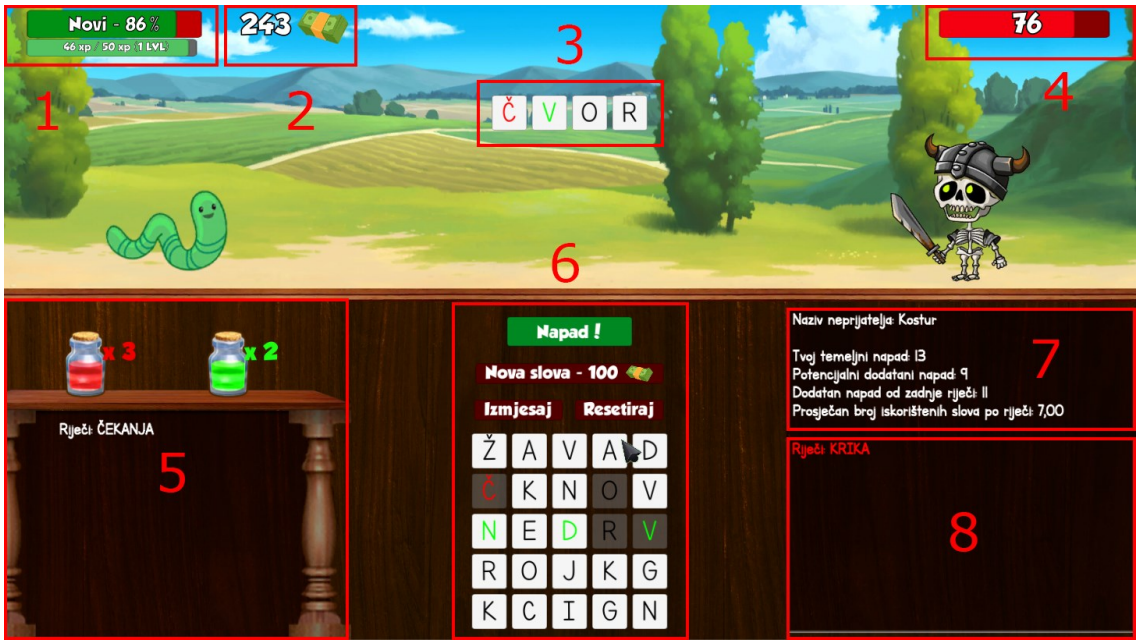
Skripta 4. Funkcija Start u GameManagment skripti

Funkcije koje poziva *Start* funkcija su vrlo jednostavne. One provjeravaju učitane podatke, koji su spremljeni u *MenuScript* skripti te na temelju njih određuju koja pozadina i koji neprijatelj trebaju biti prikazani igraču. U Start funkciji se također izračuna početni napad igrača, koji se temelji na trenutnom *levelu* te se generiraju početna slova. Funkciju *LoadBackground* koju poziva funkcija *Start* možete vidjeti u nastavku.

```
void LoadBackground()
    {
        if (MenuScript.lvl == 1)
            bg.GetComponent<Image>().sprite = bg1;
        else if (MenuScript.lvl == 2)
            bg.GetComponent<Image>().sprite = bg2;
        else if (MenuScript.lvl == 3)
            bg.GetComponent<Image>().sprite = bg3;
        else if (MenuScript.lvl == 4)
            bg.GetComponent<Image>().sprite = bg4;
        else if (MenuScript.lvl == 5)
            bg.GetComponent<Image>().sprite = bg5;
        else if (MenuScript.lvl == 6)
            bg.GetComponent<Image>().sprite = bg6;
        else if (MenuScript.lvl == 7)
            bg.GetComponent<Image>().sprite = bg7;
        else if (MenuScript.lvl == 8)
            bg.GetComponent<Image>().sprite = bg8;
    }
```

Skripta 5. Funkcija za učitavanje prikladne pozadine

Odmah pri učitavanju scene *BattleScene*, igraču se prikazuju sve korisne informacije, kao što su trenutni postotak života, prikupljeno iskustvo i život neprijatelja. Funkcija refreshStats je zaslužna za osvježavanje informacija na ekranu te ćemo kasnije reći malo više o njoj. Neke od tih korisnih informacija smještene su na samom vrhu ekrana te ih možete vidjeti na Slici 11., na oznakama 1, 2 i 4.



Slika 11. Avantura mod

Oznaka 6, označena na Slici 11., prikazuje 25 slova koja se generiraju prilikom pokretanja scene ili pritiskom na gumb Nova slova. Slova se generiraju uz pomoć funkcije *GenerirajSlova* iz skripte *GeneratorSlova*. Funkcija ima ograničenje na generiranje samoglasnika te posebnih, rijetkih slova (varijable *samoglasnici_counter* i *rijetka_slova_counter*), kako bi se izbjeglo generiranje previše istih znakova, odnosno kako bi postigli veći broj ostalih slova. *Random.Range*() funkcijom nasumično generiramo brojeve koji nam pomažu u generiranju različitih slova (*newButton* je gumb koji se generira). Tom funkcijom također nasumično određujemo, koje slovo će biti koje boje, odnosno koje slovo će donositi jači napad. Na kraju funkcije se generiranim gumbovima dodaje što će se dogoditi ako pritisnemo na njega. Kod koji generira slova možete vidjeti u Skripti 6.

```
char[] samoglasnici = { 'A', 'E', 'I', 'O', 'U' };
char[] ostala_slova = { 'B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'P', 'R',
'S', 'T', 'V', 'Z' };
char[] rijetka_slova = { 'Č', 'Ć', 'Đ', 'Š', 'Ž' };

public void GenerirajSlova()
    {
        int samoglasnici_counter = 0;
        int rijetka_slova_counter = 0;
        Game_Managment.slova.Clear();

        for (int i = 0; i < 25; i++)
        {
            Button newButton = Instantiate(button) as Button;
            if (UnityEngine.Random.Range(0, 3) == 0 && samoglasnici_counter < 6)
            {
                samoglasnici_counter++;
```

```
                newButton.GetComponentInChildren<Text>().text = GenerirajSlovo(0);
            }
            else if (UnityEngine.Random.Range(0, 3) == 1 && rijetka_slova_counter < 2)
            {
                rijetka_slova_counter++;
                newButton.GetComponentInChildren<Text>().text = GenerirajSlovo(1);
            }
            else
            {
                newButton.GetComponentInChildren<Text>().text = GenerirajSlovo(2);
            }
            if (UnityEngine.Random.Range(0, 12) == 1)
            {
                newButton.GetComponentInChildren<Text>().color = Color.green;
                newButton.name = "GreenBTN_" +
newButton.GetComponentInChildren<Text>().text + "_" + i;
            }
            else if (UnityEngine.Random.Range(0, 50) == 1)
            {
                newButton.GetComponentInChildren<Text>().color = Color.red;
                newButton.name = "RedBTN_" +
newButton.GetComponentInChildren<Text>().text + "_" + i;
            }
            else
            {
                newButton.name = "BTN_" + newButton.GetComponentInChildren<Text>().text +
"_" + i;
            }
            Game_Managment.slova.Add(newButton.GetComponentInChildren<Text>().text[0]);
            newButton.GetComponent<Button>().onClick.AddListener(delegate {
button_Click(newButton); });
            newButton.transform.SetParent(layoutGeneratedLetters.transform, false);
        }
}
```

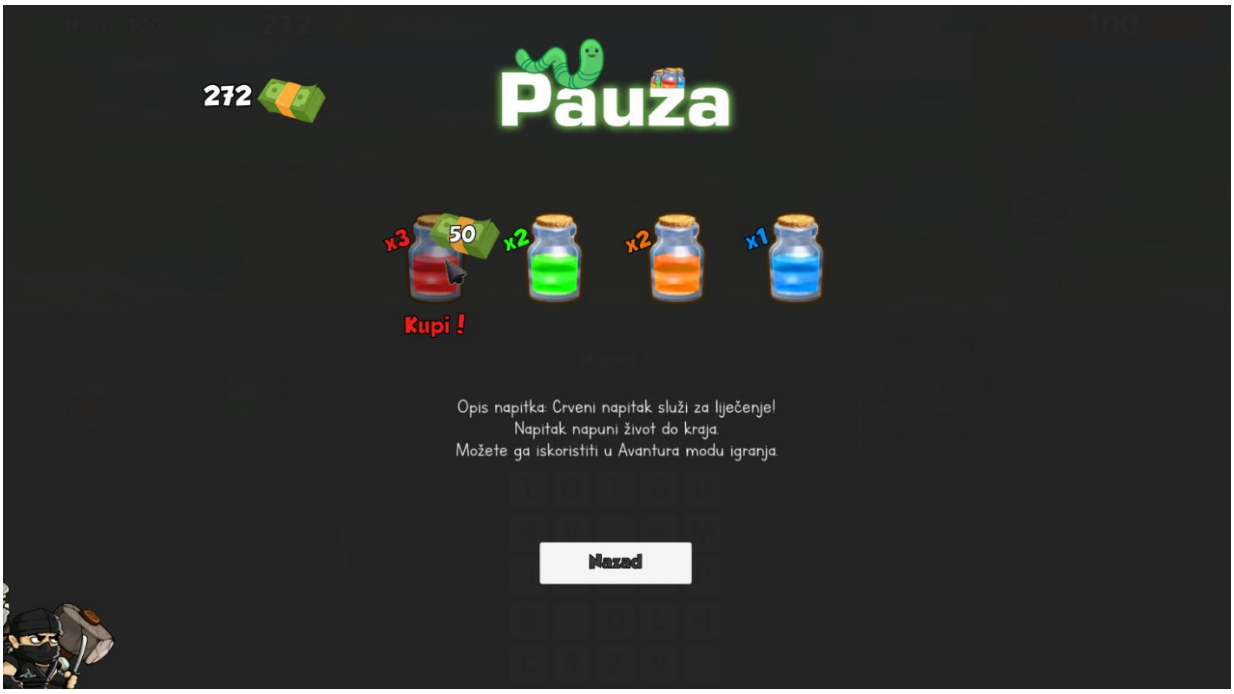
Skripta 6. Funkcija za generiranje nasumičnih slova

Ispod gumba za kupnju novih slova nalaze se gumbovi Izmješaj te Resetiraj. Gumb izmješaj promješa 25 generiranih slova, a gumb Resetiraj briše sva pritisnuta slova. Pritisnuta slova se dupliciraju i prikažu korisniku kao što možete vidjeti na Oznaci 3 (Slika 11). Slova koja su pritisnuta postanu siva, odnosno na njih se ne može više puta pritisnuti. Možete primjetiti da su neka slova zelene i crvene boje. Igraču se uz temeljni napad, računa i dodatni napad. Dodatni napad se računa prema broju slova riječi kojom napadamo, s time da crveno slovo korisniku dodatni napad povećava za 5, zeleno slovo za 2, a obično crno slovo za 1. Primjer sa Slike 11. nam prikazuje riječ ČVOR, gdje su slova Č i V, crvene i zelene boje što nam postavlja potencijalni dodatni napad te riječi na 9. Pri vrhu oznake 6 nalazi se gumb za napad koji se igraču omogućuje tek kada program pronađe unesenu riječ igrača u bazi od više od 300 000 riječi [23] koja se učitava na pokretanu igre. Svaki puta kada igrač pritisne neko slovo program provjerava je li unio postojeću riječ. Dio koda koji je zadužen za provjeru unesenih riječi nalazi se u Skripti 7.

```
void Check_if_word_exists(string word)
    {
        bool postoji = false;
        foreach (string r in Game_Managment.imenice)
        {
            if (r.ToUpper() == word)
            {
                try
                {
                    foreach (string i in Game_Managment.spremljene_rijeci)
                    {
                        if (i == word)
                        {
                            postoji = true;
                            break;
                        }
                        else { postoji = false; }
                    }

                    if (!postoji)
                    {
                        btn_attack.interactable = true;
                    }
                    else { btn_attack.interactable = false; }

                    break;
                }
                catch { }
            }
            else { btn_attack.interactable = false; }
        }
    }
```

Skripta 7. Funkcija za provjeru sastavljenih riječi

Oznaka 5, na Slici 11., sadrži riječi koje je igrač sastavio i dva gumba, odnosno napitka koji igrač može iskoristiti. Crveni napitak služi za lječenje igrača, a zeleni napitak povećava temeljni napad za 10, ali samo na jedan potez. Narančasti i plavi napitak se mogu iskoristiti u drugom modu igranja. Narančasti usporava vrijeme na 50 % na 10 sekundi, a plavi dodaje 30 sekundi dodatnog vremena. Napitci se mogu kupovati ili na glavnom izborniku ili na izborniku za pauzu, koji se otvara pritiskom na tipku ESC. Slika 12 prikazuje trgovinu napitaka u izborniku za pauzu.



Slika 12. Trgovina napitaka

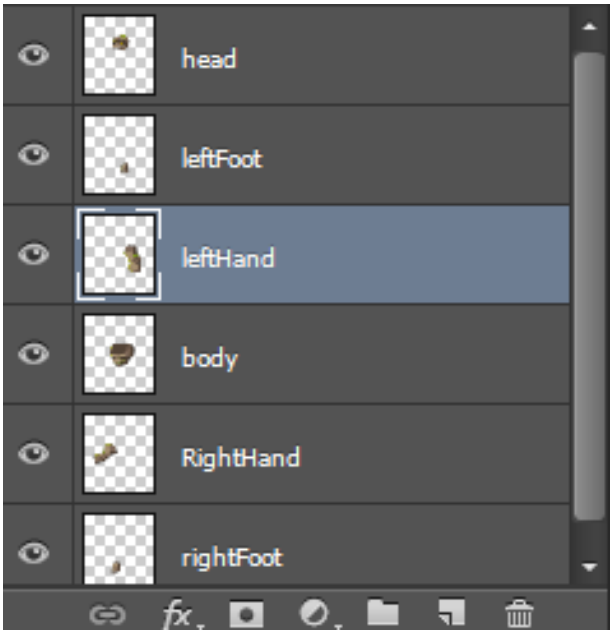
Oznaka 8 (Slika 11.) prikazuje riječi koje je neprijatelj iskoristio, a iznad toga (Oznaka 7) igraču se prikazuju informacije poput: ime neprijatelja, temeljni napad, dodatni napad prethodne riječi i potencijalni dodatni napad.

Neprijatelj

Na svakoj od 8 razina nalazi se 8 različitih neprijatelja. Njihova jačina napada ovisna je o igračevom trenutnom *levelu*. *Spriteovi* [24] neprijatelja su preuzeti u obliku kao što je prikazano na Slici 13.

Slika 13. Sprite neprijatelja

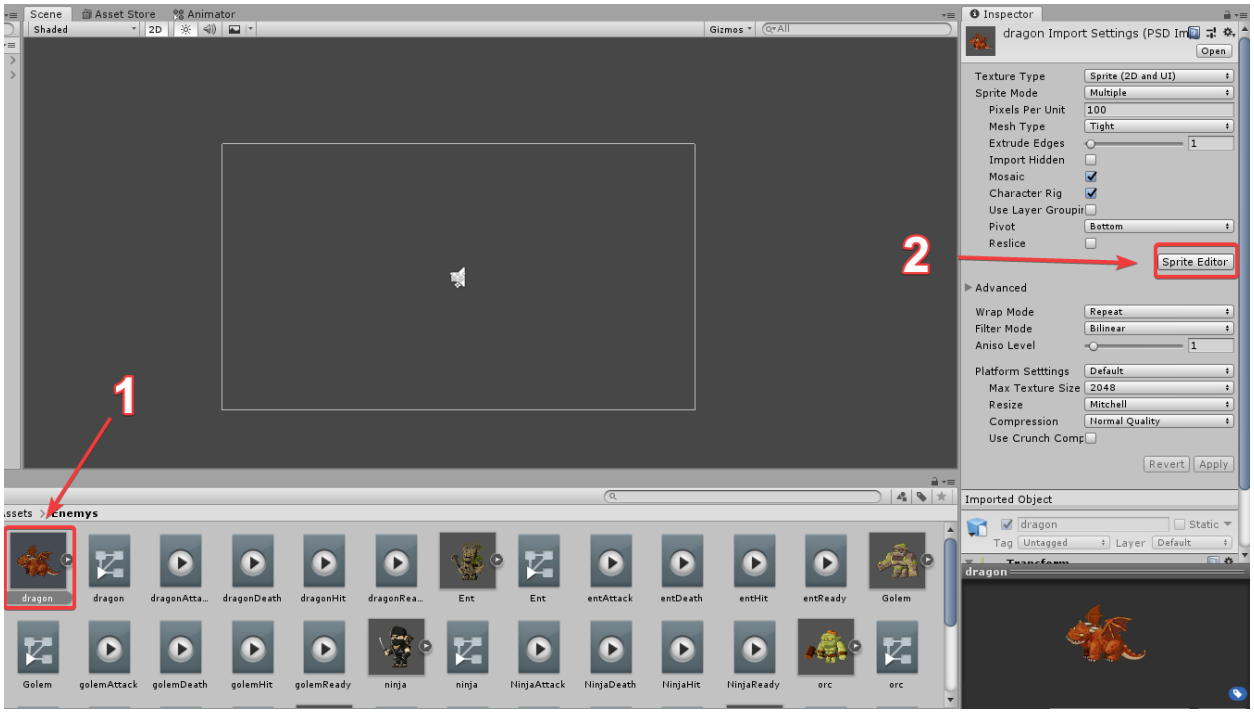
Kako bismo postigli kretanje, napadanje i ostale animacije tog lika, prvo Adobe Photoshop alatom izrežemo nepotrebne dijelove te slike te na *layere* [25] podijelimo sve dijelove tijela. Primjer možemo vidjeti na Slici 14. Sliku spremamo u PSB formatu [26] kojega Unity može prepoznati, a kako bi to postigli, moramo preuzeti i instalirati 2D PSD Importer [27] paket u Unity Package Manageru. Na Slici 15. je prikazan instalirani paket. Također je odmah dobro i preuzeti pakete: 2D Animation [28] i 2D IK [29] koji nam omogućuju animiranje likova.



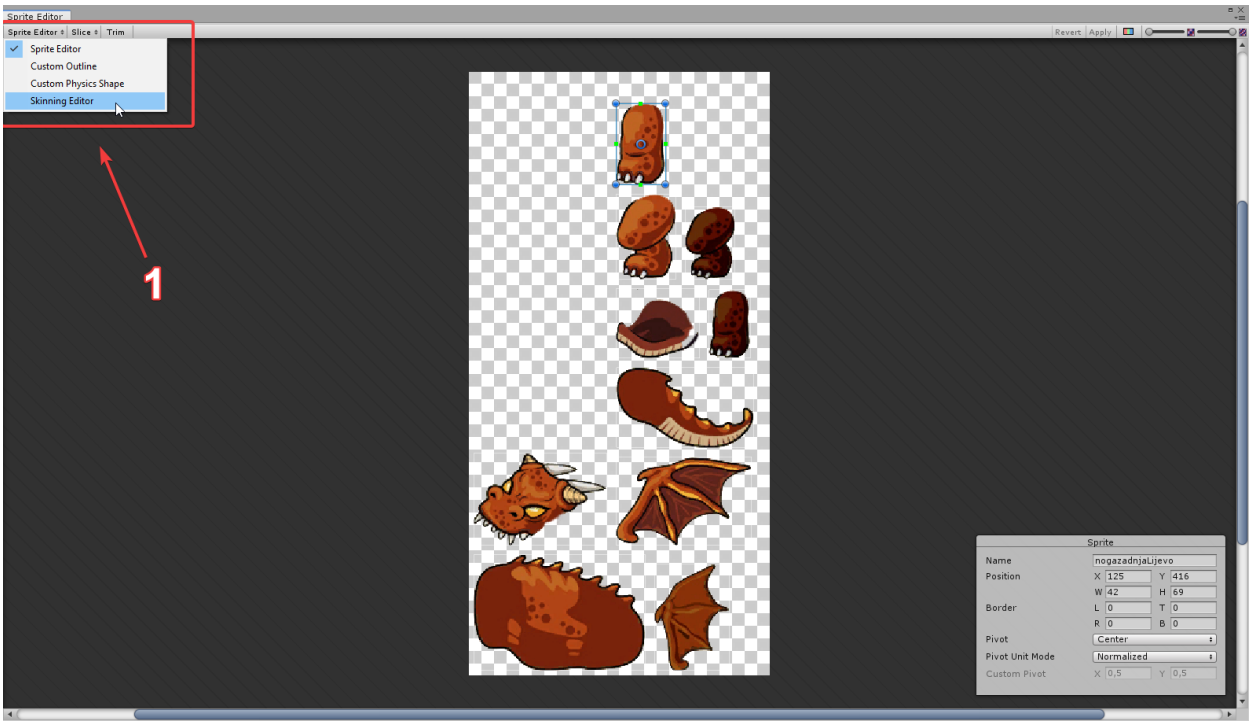
Slika 14. Dijelovi tijela neprijatelja

Slika 15. Instalirani paketi

Nakon dodavanja svih potrebnih paketa, prethodno generirane PSB datoteke su prikazane u projektu (oznaka 1) kao što možete vidjeti na Slici 16., te pritiskom na njih možemo otvoriti Sprite Editor (oznaka 2, Slika 16) [30]. Izgled Sprite Editora prikazan je na Slici 17.



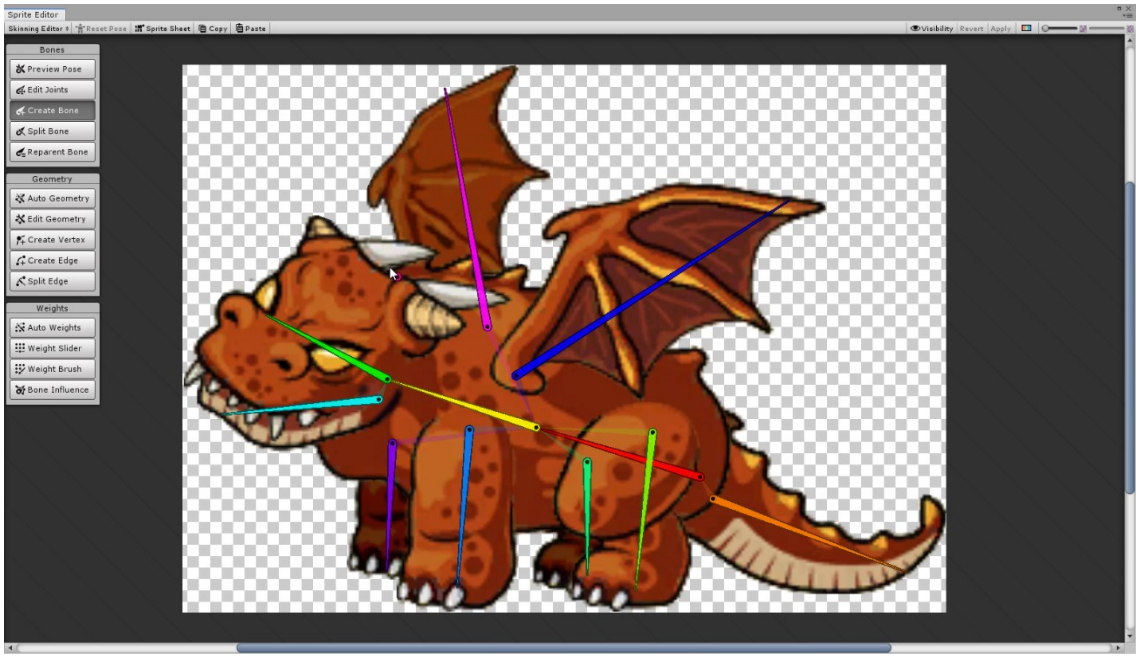
Slika 16. Prikaz PSB datoteke u projektu



Slika 17. Sprite Editor

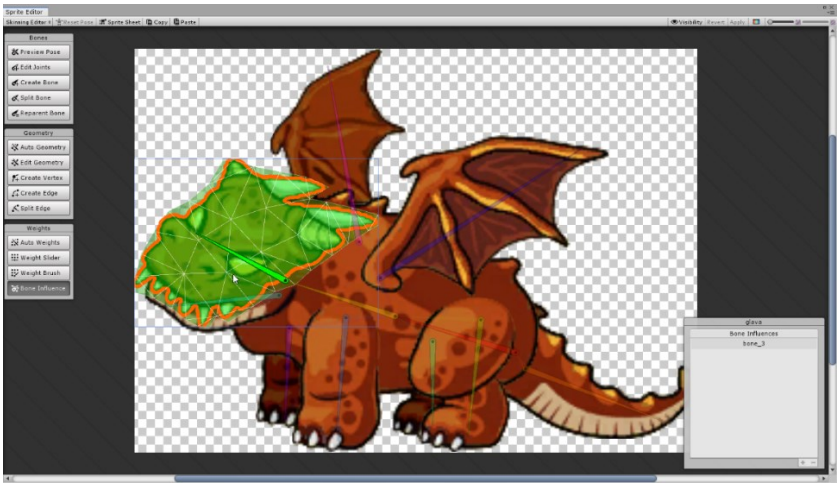
Kako bismo animirali likove, najprije moramo otvoriti Skinning Editor [31] kao što je prikazano na prethodnoj Slici 17 (oznaka 1). U Skinning Editoru želimo definirati kosti i zglobove našeg lika kako bi Unity znao koji dio tijela treba pomicati. Struktura kostiju zmaja prikazana je na Slici 18. Kosti

dodajemo pritiskom na gumb dodaj kost (engl. *Create Bone*) te pritiskanjem na sprite.



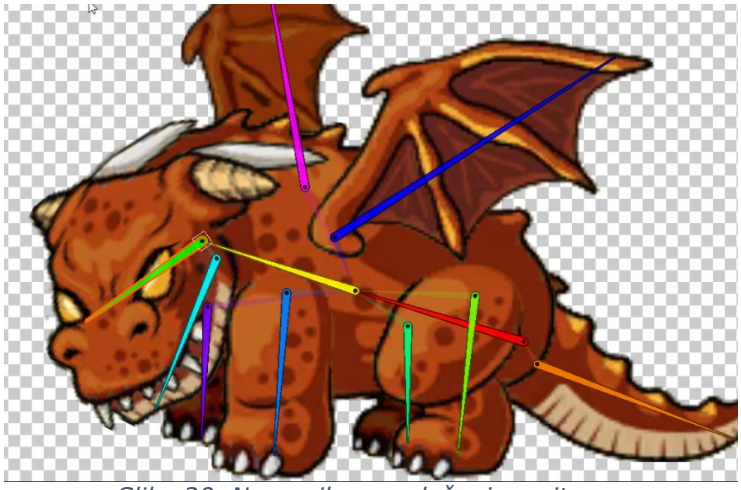
Slika 18. Struktura kostiju zmaja

Po završetku dodavanja kostiju na lika, trebamo još provjeriti na koje sve dijelove tijela djeluju određene kosti. To možemo pregledati pritiskom na gumb Bone Influence na lijevom izborniku koji je prikazan na Slici 16. Dvostrukim klikom na određeni dio tijela, možemo vidjeti koje sve kosti utječu na određeni dio tijela. Primjer možete vidjeti na Slici 19. gdje zelena kost na glavi utječe samo na gornji dio glave. Pažljivo biramo koje kosti imaju utjecaj kako bi se izbjegle distorzije na spriteovima. Na Slici 19., u lijevom donjem kutu, možete vidjeti hijerarhijsku strukturu kosti koje utječu na odabrani dio tijela, odnosno glavu.



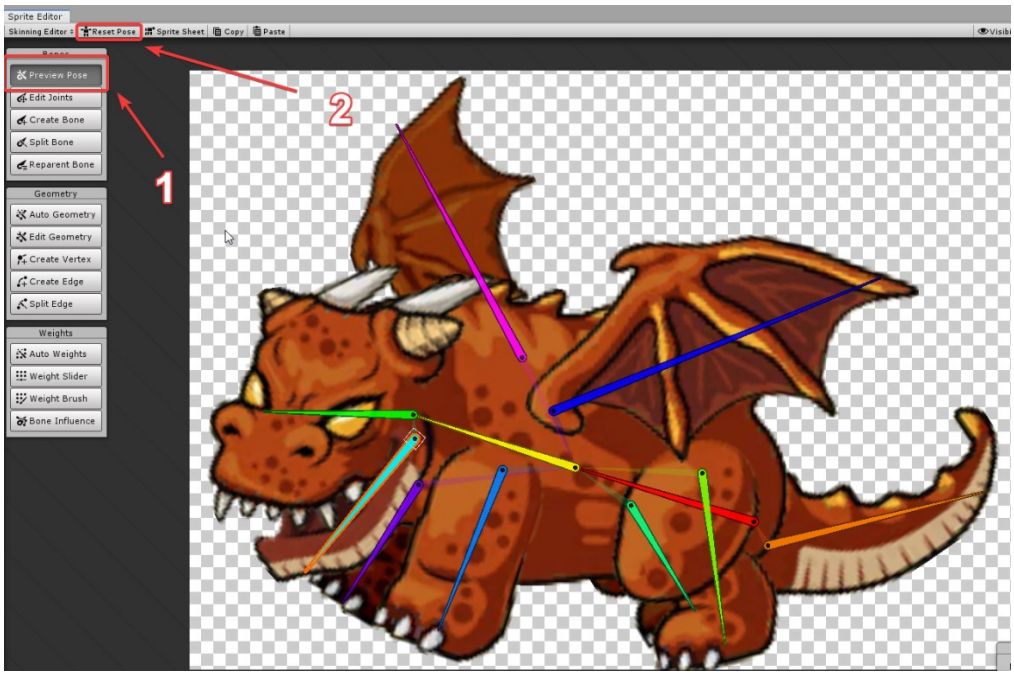
Slika 19. Kosti koje utječu na gornji dio glave

Ukoliko bi, primjerice, dodali još kost od tijela na tu listu, pomicanje glave bi čudno razvuklo sprite glave jer kost tijela ima utjecaj na nju. Primjetite na Slici 20., kako se gornji dio glave neprijatelja razvuče uslijed krivog podešavanja utjecaja kosti.



Slika 20. Nepravilno razvlačenje spriteova

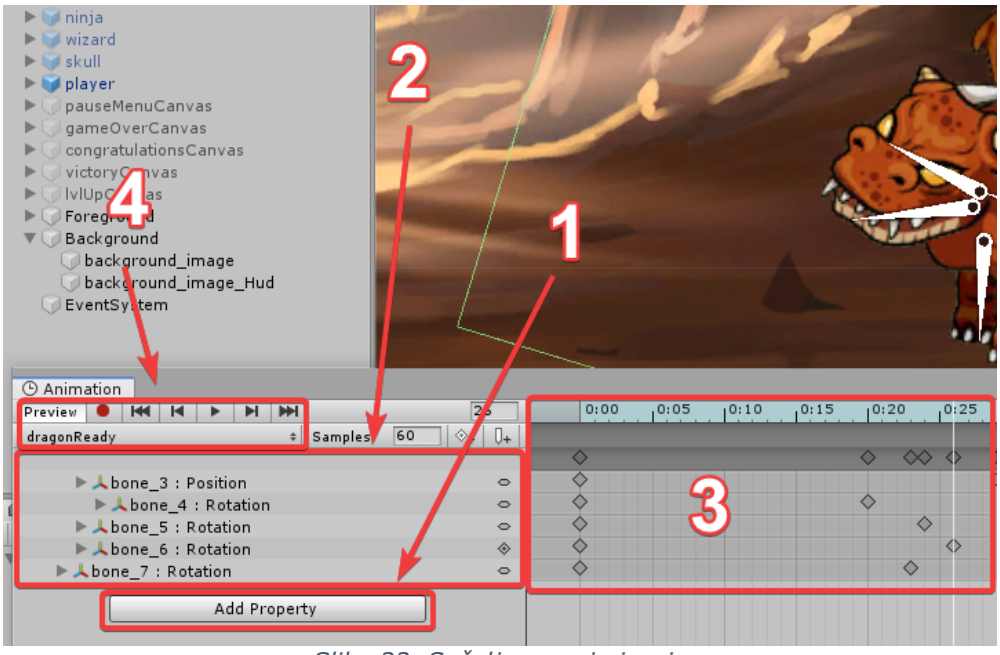
Pritiskom na gumb Preview Pose (oznaka 1), koji se vidi na Slici 21., testiramo jesmo li sve dobro posložili. Pomicanjem kostiju vidimo ima li nekakvih nepravilnosti i gumbom Reset Pose (oznaka 2) na Slici 21. možemo vratiti lika u prirodni položaj.



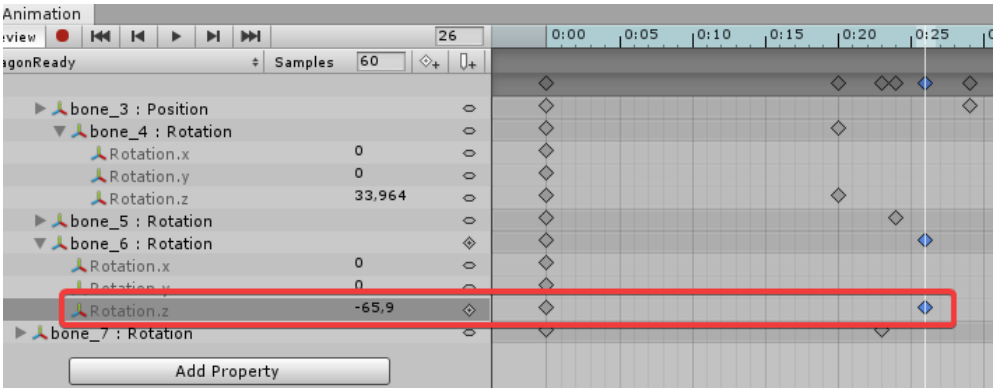
Slika 21. Pomicanje kostiju zmaja

Animiranje

Po završetku izrade kostura možemo krenuti na samo animiranje. Dodajemo lik na scenu, na željenu poziciju i otvorimo prozor za animiranje [32]. Slika 22. prikazuje kako izgleda prozor za animiranje (engl. *Animation*). Pritiskom na gumb Add Property (oznaka 1) na Slici 22., dodajemo kosti koje želimo animirati. Kostima možemo mjenjati poziciju, rotaciju te veličinu. Oznaka 2 na istoj slici prikazuje strukturu tih kostiju te njihova svojstva, odnosno trenutno stanje na x, y te z osi. Te vrijednosti možemo mjenjati na vremenskoj crti (engl. *Timeline*) (oznaka 3) pritiskom na željeni trenutak i promjenom vrijednosti nekog svojstva. Unity sam dodaje keyframe te će postupno mjenjati vrijednost svojstva do tog trenutka. Primjer svojstva možete vidjeti na Slici 23. Oznaka 4, Slika 22., prikazuje gumbove za navigaciju kroz vremensku crtu, pokretanje animacije i slično. Tamo se također dodaju i nove animacije za istog lika pritiskom na ime trenutne animacije. Potom se otvara mali prozor u kojemu nam program nudi mogućnost dodavanja nove animacije.



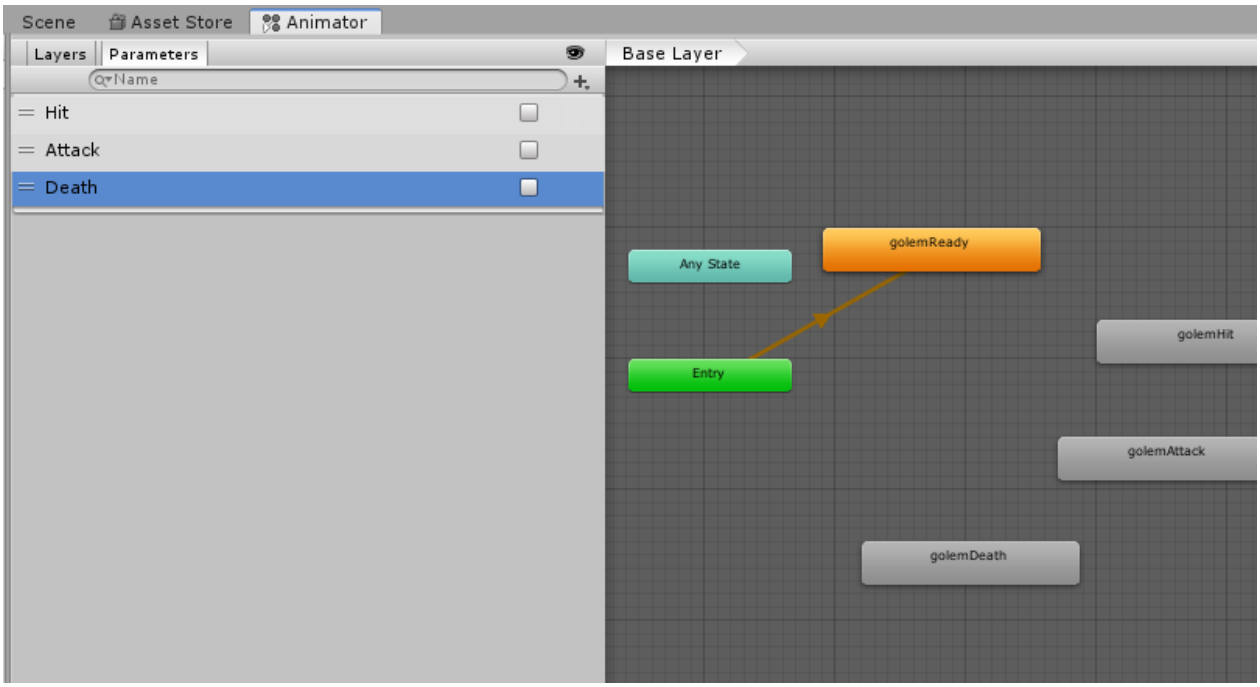
Slika 22. Sučelje za animiranje



Slika 23. Pomicanje zglobova po vremenskoj crti

Animator

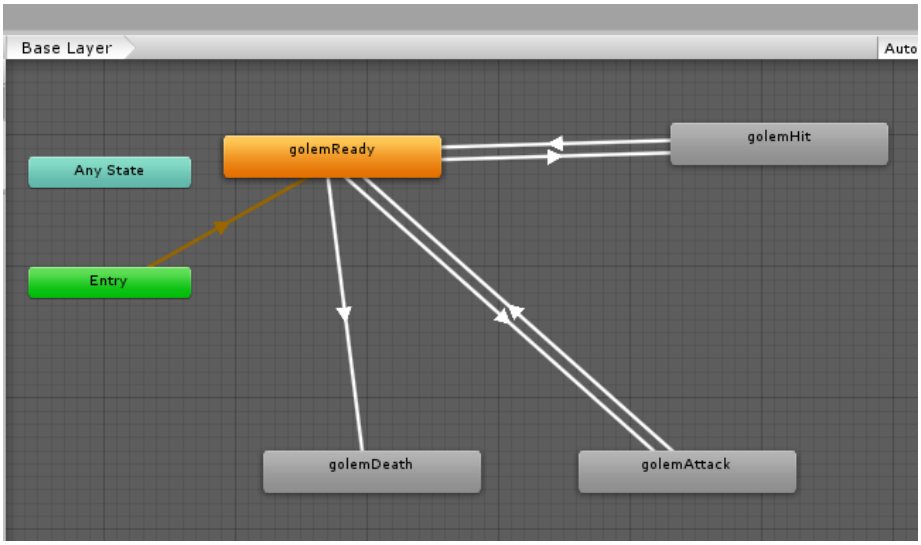
Kada završimo izradu animacija trebamo grupirati te animacije za određeni objekt te urediti prijelaze između njih kako bi tranzicija iz animacije u animaciju izgledala bolje. Kao primjer možemo uzeti prijelaz iz stanja mirovanja u stanje napadanja. Unity ima ugrađeni Animator [33], odnosno Animator Controller koji nam to omogućuje. Primjer kontrolera animacija možete vidjeti na Slici 24. Na lijevoj strani slike možete vidjeti parametre koje koristimo unutar skripta kako bi uspješno napravili tranziciju između animacija. U ovom slučaju imamo ih tri koje smo nazvali *Hit*, *Attack* i *Death*. Desni dio prozora nam vizualno prikazuje animacije koje smo napravili kako bismo ih lakše povezali, odnosno napravili prijelaze između njih.



Slika 24. Animator

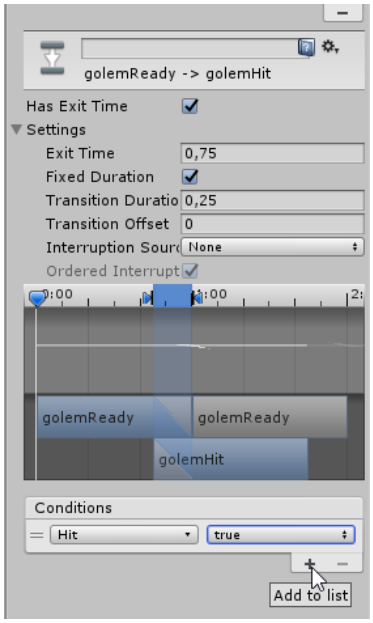
Prijelaze između animacija dodajemo desnim klikom na početnu animaciju, pritiskom na opciju *Make Transition* na izborniku te povlačenjem

bijele strelice na završnu animaciju. Neke animacije se izvrše samo jednom, poput one animacije napadanja, pa isto tako radimo i povratnu tranziciju nazad. Kako bi dodatno spriječili ponavljanje takve animacije pritisnemo na željenu animaciju te maknemo oznaku svojstvo Loop Time. Primjer kontrolera za neprijatelja koji se zove *Golem* možete vidjeti na Slici 25.



Slika 25. Animation controller

Kada napravimo sve prijelaze, moramo samo još definirati kada se koja tranzicija dogodi. Pritiskom na željeni prijelaz, u inspektoru nam se prikazuju sve opcije koje su vezane za prijelaz iz animacije u animaciju (Slika 26). Na dnu možemo dodati parametar s uvjetom kada se događa prijelaz. Unutra također možemo podešavati trajanje tranzicije te njihove *offsetove*.



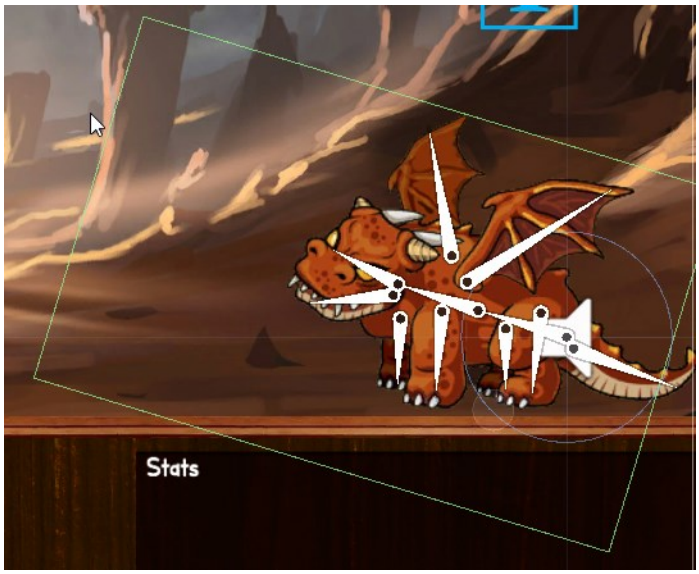
Slika 26. Postavke za prijelaz animacije

Skripta *EnemyAnim*, koja je dodana na svaki od neprijatelja, upravlja prijelazima animacija neprijatelja i zvukovima napadanja. Funkcija koja je najbitnija u toj skripti zove se *OnTriggerEnter2D* i ona prima ulazni parametar *Collider2D*. Funkcija se poziva kada *collider* igrača dotakne *collider* neprijatelja i ona nam služi kao okidač kada će se dogoditi prijelaz iz stanja primanja udarca u stanje napadanja ili u stanje umiranja neprijatelja. Također se u njoj nalaze i zvukovni efekti i animacije efekata udaraca [34] koji se instanciraju na neprijatelju. Funkciju *OnTriggerEnter2D* možete vidjeti u Skripti 8.

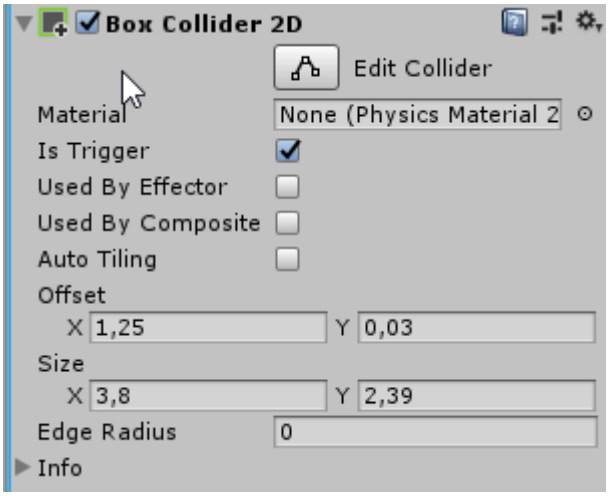
```
void OnTriggerEnter2D(Collider2D collision)
    {
        if (!animator.GetBool("Attack"))
        {
            System.Random r = new System.Random();
            int n = r.Next(1, 2);
            if (n == 1)
                Instantiate(punchEffect1, new Vector3(6f, 1.5f, 0), Quaternion.identity);
            else
                Instantiate(punchEffect2, new Vector3(6f, 1.5f, 0), Quaternion.identity);

            this.GetComponent<AudioSource>().clip = s_punch;
            this.GetComponent<AudioSource>().Play();
        }

        if (!animator.GetBool("Attack") && !animator.GetBool("Hit") && health.value > 0)
        {
            animator.SetBool("Hit", true);
            StartCoroutine(DisableHitAnimation());
            StartCoroutine(DisableAttackAnimation());
        }
        else if(!animator.GetBool("Attack") && !animator.GetBool("Hit") && health.value
<= 0)
        {
            animator.SetBool("Death", true);
            StartCoroutine(DisableDeathAnimation());
            this.GetComponent<AudioSource>().clip = s_punch;
            this.GetComponent<AudioSource>().Play();
        }
    }
```

Skripta 8. Funkcija za provjeru kolizija

Na Slici 27., možete vidjeti *Collider* koji je oznacen zelenom crtom u pravokutnom obliku. Da bi mogao pravilno funkcionirati *Collider* mora biti postavljen kao okidač. To svojstvo se dodaje u inspektoru označavanjem opcije *Is Trigger* kao što je prikazano na Slici 28.



Slika 27. Collider neprijatelja



Slika 28. Postavke collidera

Skripta *GameManagment* također upravlja napadanjem neprijatelja. Jednostavnim algoritmom prolazi kroz više od 300 000 učitanih riječi i prema trenutnom levelu igrača filtrira riječi po broju slova. Tijekom napada neprijatelja, igraču se onemogućuju sve funkcije kao što su: gumbovi generiranih slova, tipke reset, generiranje novih slova. Najmanja riječ koju neprijatelj pronazali sastavljena je od 4 slova, a najveća se računa prema formuli: trenutni *level* igrača + 4. Dio koda koji filtrira riječi prema levelu igrača nalazi se u nastavku. Kako se riječi ne bi ponavljale tijekom igranja, još ih i izmješa.

```
List<string> lista = new List<string>();
        foreach(string rijec in rijeci)
        {
            if(rijec.Length >=4 && rijec.Length <= (CalculateLVL(PlayerData.xp)+4))
            { lista.Add(rijec); }
```

```
        }
```

Skripta 9. Filtriranje riječi

Algoritam prolazi kroz izmješane filtrirane riječi i za svaku provjerava postoje li sva slova potrebna da se sastavi riječ. Ukoliko jedno slovo nedostaje, algoritam odmah preskače na sljedeću riječ i tako sve dok ne pronađe prvu riječ koja odgovara. Algoritam zaslužan za provjeru postoje li za pojedinu riječ sva slova nalazi se u Skripti 10.

```
       string r = "";
        bool pronadeno_slovo = false;
        foreach (string rijec1 in lista)
        {
            string rijec = rijec1.ToUpper();
            if(!tb_rijeci.text.Contains(rijec + " ") &&
!tb_rijeci_enemy.text.Contains(rijec + " "))
            {
                string temp = "";
                for (int i = 0; i < rijec.Length; i++)
                {
                    for (int j = 0; j < slovaTemp.Length; j++)
                    {
                        if (slovaTemp[j] == ime[i])
                        {
                            temp += slovaTemp[j];
                            slovaTemp[j] = ' ';
                            pronadeno_slovo = true;
                            break;
                        }
                        else { pronadeno_slovo = false; }
                    }
                    if(!pronadeno_slovo)
                        slovaTemp = slova.ToArray();

                }
                if(rijec == temp)
                {
                    spremljene_rijeci.Add(rijec);
                    r = rijec;
                    tb_imenice_enemy.text += rijec + " ";
                    break;
                }
            }
        }
```

Skripta 10. Funkcija za provjeru postoje li sva potrebna slova

Igrač

Složena *GameManagment* skripta također prati i trenutno stanje igrača i neprijatelja. Funkcija *RefreshStats* je zaslužna za prikazivanje i osvježavanje informacija koji su prikazani na ekranu. Obično se poziva na kraju gotovo svake funkcije te često ide u paru s funkcijom *SaveProgress* koja sprema napredak igrača. Funkciju *RefreshStats* možete vidjeti u Skripti 11.

```
public void refreshStats()
    {
        lbl_e_health.text = PlayerData.e_c_health.ToString();
        slider_p_health.maxValue = PlayerData.m_health;
        slider_p_health.value = PlayerData.c_health;
        slider_e_health.maxValue = PlayerData.e_m_health;
        slider_e_health.value = PlayerData.e_c_health;

        lbl_num_red_potions.text = "x "+PlayerData.red_potions.ToString();
        lbl_num_green_potions.text = "x "+PlayerData.green_potions.ToString();

        lbl_xp.text = PlayerData.xp + " xp / "+CalculateXP(PlayerData.xp) + " xp
("+CalculateLVL(PlayerData.xp)+ " LVL)";
        slider_xp.maxValue = CalculateXP(PlayerData.xp);
        slider_xp.value = PlayerData.xp;

        lbl_xp_result.text = PlayerData.xp + " xp / " + CalculateXP(PlayerData.xp) + " xp
(" + CalculateLVL(PlayerData.xp) + " LVL)";
        slider_xp_result.maxValue = CalculateXP(PlayerData.xp);
        slider_xp_result.value = PlayerData.xp;

        lbl_money.text = PlayerData.credits.ToString();
        lbl_stats.text = "Naziv neprijatelja: " + naziv_neprijatelja[MenuScript.lvl-1] +
"\n\nTvoj temeljni napad: " + PlayerData.BaseAttackDmg + "\nPotencijalni dodatani napad:
" + gs.bonusAttack + "\nDodatan napad od zadnje riječi: " + bonusAttack + "\nProsječan
broj iskorištenih slova po riječi: " + ProsjekSlova().ToString("F2");

        int healthPercent = (int)Math.Round(slider_p_health.value /
slider_p_health.maxValue * 100f);
        lbl_player_name.text = PlayerData.username + " - " + healthPercent + "%";

        lbl_lvl_up_canvas.text = CalculateLVL(PlayerData.xp).ToString();
        lbl_lvl_up_xp_canvas.text = PlayerData.xp + " xp / " + CalculateXP(PlayerData.xp)
+ " xp";
        slider_xp_LVLup.maxValue = CalculateXP(PlayerData.xp);
        slider_xp_LVLup.value = PlayerData.xp;
    }
```

Skripta 11. Funkcija za osvježavanje prikazanih podataka na ekranu

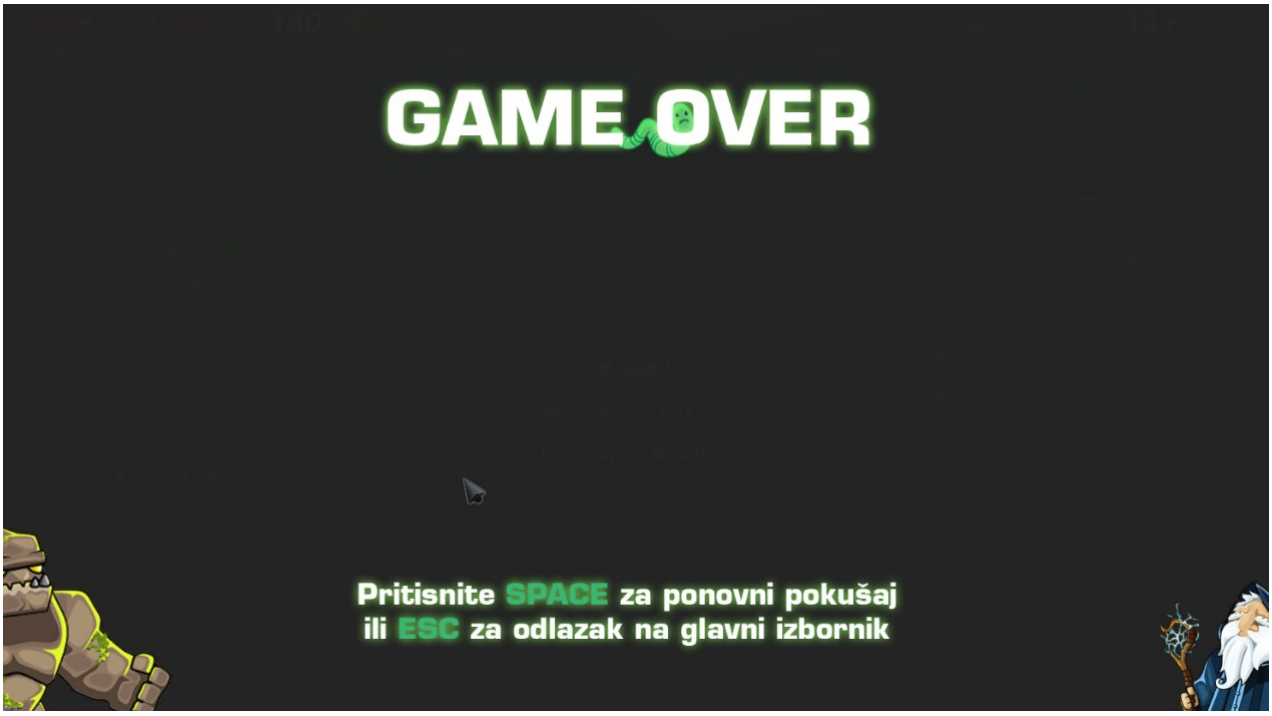
Po završetku svakog *levela* prikazuju se prikladni prozori, na kojima se čestita igraču na izvršenju. Na njima se nalaze informacije poput imena igrača, trenutnog iskustva te novih riječi koje je igrač pronašao. Primjer jednog takvog prozora možete vidjeti na Slici 29. Pritiskom na tipku *Space* zatvara se taj prozor te se učitava sljedeći *level*. Igrač poslije svakog napada dobiva iskustva u iznosu bonus napada, odnosno od riječi koju sastavi. Ukoliko igrač dobije novi *level*, prikazuje mu se zaslon koji mu također prikazuje ime, iskustvo, ali i *level* koji je postigao. Primjer takvog zaslona možete vidjeti na Slici 30.

Slika 29. Prozor za završeni level

Slika 30. Prozor za level up

Igraču se po završetku zadnjeg levela prikazuje zaslon čestitke, kao što je prikazano na Slici 31. Pritiskom na tipku Space igrača se prebacuje na glavni izbornik. Ukoliko igrač izgubi sav život prilikom igranja, prikazuje se zaslon na kojem mu se nudi ponovni pokušaj ili odlazak na glavni izbornik (Slika 32).

Slika 31. Prozor čestitke

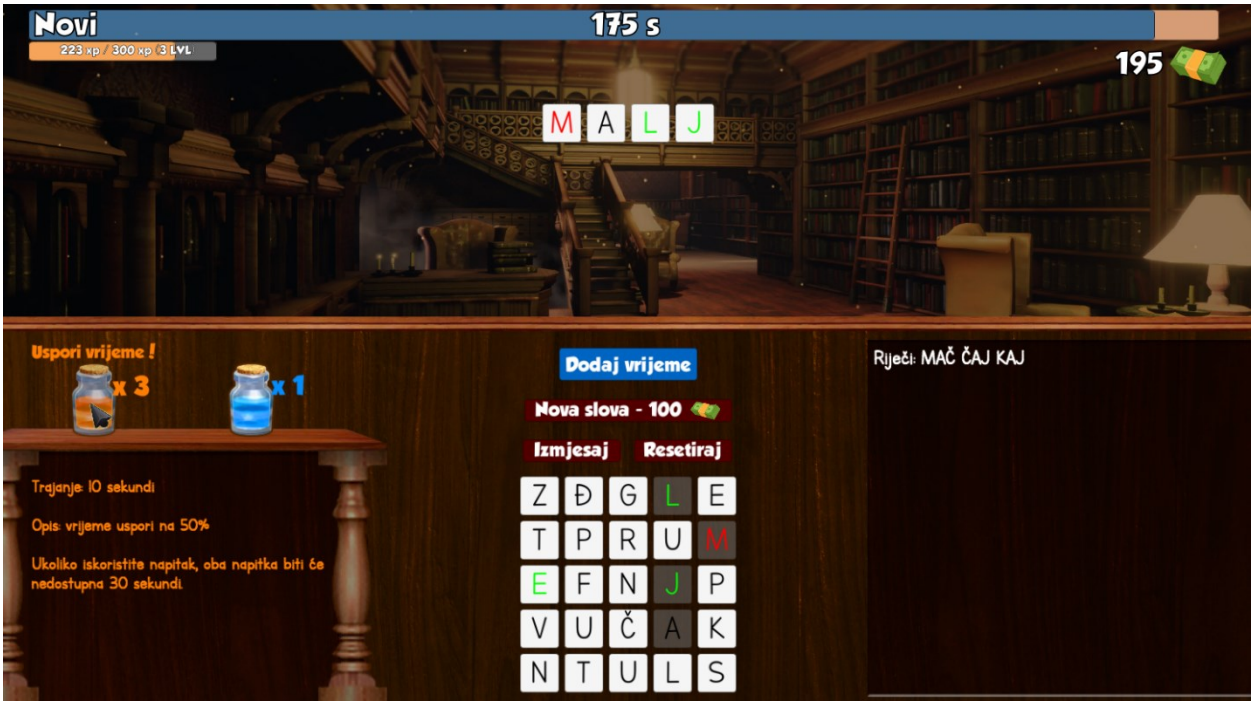


Slika 32. Prozor kad neprijatelj pobjedi

Utrka s vremenom

U modu igranja nazvan Utrka s vremenom, igraču je cilj u zadanom vremeno pronaći što više riječi. Ovaj mod igranja omogućuje igraču sakupljanje novca i iskustva kako bi kupovao napitke i lakše prošao Avantura mod. Na Slici 33., prikazana je scena tog moda. Na samom početku učitavanja

scene generiraju se slova uz pomoć *GenerirajSlova* skripte kao i u Avantura modu. S obzirom da mod Utrka s vremenom nema neprijatelje, iz skripte *GameManagment* iskorištene su samo određene funkcije. To su primjerice funkcije za generiranje novih slova, miješanje slova, resetiranje i izračunavanje levela i iskustva. Funkcija za napad neprijatelja je nepotrebna, a funkcija koja je služila kao napad igrača se morala izmjeniti. U nastavku su spomenuti i objašnjeni samo ključni dijelovi *GameManagment_TimeAttack* skripte.



Slika 33. Utrka s vremenom

Na početku se sve funkcije igraču onemogućuju i pokreće se odbrojavanje od 10 sekundi. U tom kratkom vremenu igrač može vidjeti generirana slova i odlučiti koje riječi može sastaviti odmah na početku. *Start* funkcija unutar *GameManagment_TimeAttack* skripte pokreće odbrojavanje uz pomoć funkcije *StartCoroutine*() [35] i *InvokeRepeating*() [36]. *StartCoroutine* je funkcija koja omogućuje izvršenje nekog koda nakon određenog vremena. *InvokeRepeating* funkciju smo u ovom slučaju iskoristili za potrebe odbrojavanja. Funkcija prima ulazne parametre, a to su ime funkcije, početno vrijeme i korak. Dio *Start* funkcije možete vidjeti u Skripti 12.

```
void Start()
    {
        bg.GetComponent<Image>().sprite = bg1;
        br_rijeci = 0;
        rijeci = File.ReadAllLines(pathDocuments + "\\Bookworm\\imeniceBookWorm.txt");
```

```
        InvokeRepeating("odbrojavanje",0,1);
        StartCoroutine(odbrojavanjePocetka());

        btn_blue_potion.interactable = false;
        btn_orange_potion.interactable = false;
        btn_izmjesaj.interactable = false;
        btn_resetiraj.interactable = false;
        btn_add.gameObject.SetActive(false);
        btn_add.interactable = false;
        btn_nova_slova.interactable = false;
```

Skripta 12. Start Funkcija u GameManagment_TimeAttack skripti

Funkcija koju poziva *InvokeRepeating* u *Start* funkciji zove se odbrojavanje. Ona se poziva svaku sekundu i mijenja tekst odbrojavanja. Ukoliko dođe do nule, ispisuje tekst „Kreni!" i pokreće odbrojavanje od jedne i pol sekunde poslije koje se makne tekst sa ekrana. Funkciju *Odbrojavanje* možete vidjeti u Skripti 13.

```
void odbrojavanje()
    {
        if(odbrojavanje10s > 0)
        {
            lbl_odbrojavanje.text = odbrojavanje10s.ToString();
            odbrojavanje10s -= 1;
        }
        else
        {
            lbl_odbrojavanje.text = "Kreni!";
            StartCoroutine(makniOdbrojavanje());
        }
    }

IEnumerator makniOdbrojavanje()
    {
        yield return new WaitForSeconds(1.5f);
        lbl_odbrojavanje.text = "";
    }
```

Skripta 13. Funkcije za odbrojavanje na početku

Funkciju *odbrojavanjePočetka* koja je pozvana u *Start* funkciji, nakon 10 sekundi počinje sa novom *InvokeRepeating* funkcijom. Njega u ovom slučaju koristimo i za glavno odbrojavanje vremena. Igraču se svaku sekundu na *Slideru* [37] za vrijeme mijenja vrijednost, a početno vrijeme iznosi 180 sekundi. Igraču se također omogućuju sve funkcije poslije početnog odbrojavanja. *Foreach* petlja prolazi kroz sve elemente na sceni koji su tipa gumb (engl. *Button*) i koji u imenu sadrže „BTN" te omogućuje interakciju sa istim. Ti gumbovi su zapravo generirana slova, pa uz pomoć petlje skraćujemo kod. Funkcija *odbrojavanjePocetka* nalazi se u Skripti 14.

```
IEnumerator odbrojavanjePocetka()
    {
        yield return new WaitForSeconds(10);
```

```
        CancelInvoke();
        InvokeRepeating("refreshTime", 1.0f, 1.0f);
        btn_blue_potion.interactable = true;
        btn_orange_potion.interactable = true;
        btn_izmjesaj.interactable = true;
        btn_resetiraj.interactable = true;
        btn_nova_slova.interactable = true;
        btn_add.gameObject.SetActive(true);
        foreach (Button btn1 in UnityEngine.Object.FindObjectsOfType(typeof(Button)))
        {
            if (btn1.name.Contains("BTN"))
            {
                btn1.interactable = true;
            }
        }
    }
```

Skripta 14. Funkcija za glavno odbrojavanje u Utrci s vremenom

Gumb za dodavanje vremena se omogućuje isto kao i gumb za napadanje u Avantura modu. Igra provjerava da li sastavljena riječ postoji u bazi te ukoliko postoji omogućuje se interakcija sa gumbom. Pritiskom na gumb igraču se dodaje vrijeme u iznosu bonus napada sastavljene riječi iz moda Avantura. Kao što smo već napomenuli, crveno slovo daje bonus od pet, zeleno od dvije, a crno od 1 sekunde. Prema tome bi riječ „malj", sa Slike 33., pridodao 10 sekundi. Izračunavanje bonus vremena događa se u *foreach* petlji zbog preglednosti koda. U varijablu *lvl* se sprema trenutni *level* igrača te se na kraju uspoređuje u funkciji *CheckForLVLup* kako bi se prikazao zaslon za *level up*. U varijablu *br_rijeci* se sprema trenutni broj sastavljenih riječi koju igrač vidi na kraju igranja te za provjeru medalja. Funkcija *DodajVrijeme* nalazi se u Skripti 15.

```
public void DodajVrijeme()
    {
       lvl = CalculateLVL(PlayerData.xp);
       br_rijeci++;
       btn_add.interactable = false;
       Obrisi(true);
       gs_TimeAttack.brojac = 0;

       string rijec = ""; bonusVrijeme = 0;

        foreach (Button btn1 in UnityEngine.Object.FindObjectsOfType(typeof(Button)))
               {
                   if (btn1.name.Contains("Copy") && !btn1.name.Contains("Green") &&
       !btn1.name.Contains("Red"))
                   {
                       bonusVrijeme += 1;
                       rijec += btn1.name.Split('_')[2];
                   }
                   else if (btn1.name.Contains("Copy") && btn1.name.Contains("Green"))
                   {
                       bonusVrijeme += 2;
                       rijec += btn1.name.Split('_')[2];
                   }
```

```
                else if (btn1.name.Contains("Copy") && btn1.name.Contains("Red"))
                {
                    bonusVrijeme += 5;
                    rijec += btn1.name.Split('_')[2];
                }
            }


            timeLeft += bonusVrijeme;
            if (timeLeft > slider_timeLeft.maxValue)
            {
                slider_timeLeft.maxValue = timeLeft;
            }
            slider_timeLeft.value = timeLeft;

            lbl_timeLeft.text = timeLeft.ToString() + " s";
            PlayerData.credits += bonusVrijeme;
            PlayerData.xp += bonusVrijeme;
            CheckForAchivement(rijec);
            refreshStats();
            SaveProgress();
            CheckForLVLup(lvl);
        }
```

Skripta 15. Funkcija dodaj vrijeme

Igraču se, osim što dodaje vrijeme, dodaju i novci i iskustvo u iznosu bonus vremena. Potom se provjerava da li je igrač dobio novu medalju ili *level* te se spremaju i osvježavaju podaci. Funkcija za provjeru novoosvojenih medalja kao ulazni parametar prima sastavljenu riječ. Nadalje, funkcija učitava korisničke podatke i provjerava postignuća igrača. Funkcija za provjeru postignuća, koja se zove *CheckForAchivement*, nalazi se u Skripti 16.

```
void CheckForAchivement(string rijec)
    {
        var data = File.ReadAllLines(pathDocuments + "\\Bookworm\\" + PlayerData.username
+ "_rijeci.txt");
        var lista = new List<string>(data);

        if (br_rijeci == 10 && !PlayerData.rijeci10_2)
            PlayerData.rijeci10_2 = true;
        if (br_rijeci == 25 && !PlayerData.rijeci25_2)
            PlayerData.rijeci25_2 = true;
        if (br_rijeci == 50 && !PlayerData.rijeci50_2)
            PlayerData.rijeci50_2 = true;

        if (lista.Count == 10 && !PlayerData.rijeci10)
            PlayerData.rijeci10 = true;
        if (lista.Count == 25 && !PlayerData.rijeci25)
            PlayerData.rijeci25 = true;
        if (lista.Count == 50 && !PlayerData.rijeci50)
            PlayerData.rijeci50 = true;
        if (lista.Count == 100 && !PlayerData.rijeci100)
            PlayerData.rijeci100 = true;
        if (lista.Count == 250 && !PlayerData.rijeci250)
            PlayerData.rijeci250 = true;
        if (lista.Count == 500 && !PlayerData.rijeci500)
```

```
            PlayerData.rijeci500 = true;
        if (lista.Count == 1000 && !PlayerData.rijeci1000)
            PlayerData.rijeci1000 = true;

        if (rijec.Length == 5 && !PlayerData.slova5)
            PlayerData.slova5 = true;
        if (rijec.Length == 6 && !PlayerData.slova6)
            PlayerData.slova6 = true;
        if (rijec.Length == 7 && !PlayerData.slova7)
            PlayerData.slova7 = true;
        if (rijec.Length == 8 && !PlayerData.slova8)
            PlayerData.slova8 = true;
        if (rijec.Length == 9 && !PlayerData.slova9)
            PlayerData.slova9 = true;
        if (rijec.Length >= 10 && !PlayerData.slova10)
            PlayerData.slova10 = true;
    }
```

Skripta 16. Funkcija za provjeru novoosvojenih medalja

Po isteku vremena igraču se prikazuje prigodni zaslon na kojem mu piše ime, iskustvo te broj sastavljenih riječi. Primjer zaslona možete vidjeti na Slici 34.

Slika 34. Zaslon poslije isteka vremena

Medalje

Pritiskom na tipku *Space*, igrač se vraća na početni zaslon, odnosno glavni izbornik. Pritiskom na tipku Medalje učitavaju se korisnički podaci te se otvara novi zaslon u kojemu igrač može pregledati svoja postignuća. Primjer se vidi na Slici 35. Svaka od medalja ima svoj naziv i opis koji igrač može pregledati postavljanjem pokazivača na medalju. Skripta *Postignuca* je

zaslužna za promjenu imena i opisa teksta kod medalji. Ona provjerava na kojem od medalja se nalazi pokazivač te prema tome određuje tekst. Primjer za jednu medalju možete vidjeti u nastavku.

```
public Text tb_opis;

    public void Rijeci_10_Enter()
    {
        tb_opis.text = "Naziv medalje: Novak\n\nOpis: pronađeno 10 različitih riječi.";
    }
```

Skripta 17. Primjer teksta medalje

Slika 35. Medalje

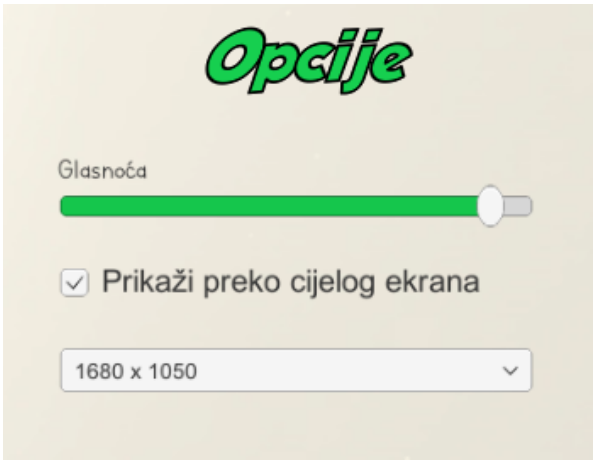
Opcije

U igru su također dodana osnovna svojstva koje igrač može mijenjati. Na glavnom izborniku pritiskom na gumb Opcije otvara se izbornik na kojemu se može promjeniti rezolucija igre, postaviti igra preko cijelog prozora i promjeniti jačina zvuka. Opcijama igre upravlja skripta *GameOptions* koja u svojoj *Start* funkciji učitava prikladne rezolucije za igračev ekran. Ostale funkcije služe za postavljanje promijenjenih vrijednosti. Funkcije možete vidjeti u Skripti 18., a izgled izbornika za opcije prikazan je na Slici 36.

```
    public Slider slider_volume;
    public Toggle toggle_fullscreen;
    public Dropdown dropdown_resolution;
    public AudioMixer am;
    Resolution[] resolutions;

public void setResolution(int resolutionIndex)
```

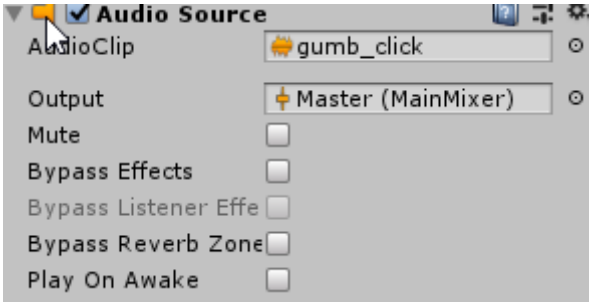
```
    {
        Resolution res = resolutions[resolutionIndex];
        Screen.SetResolution(res.width, res.height,Screen.fullScreen);
    }
public void SetVolume(float volume)
    {  am.SetFloat("volume", volume);  }

public void toggleFullscreen(bool isFullscreen)
    {  Screen.fullScreen = isFullscreen;  }
```

Skripta 18. Funkcije za kontrolu opcija



Slika 36. Opcije

U nastavku je detaljnije pojašnjeno kako je zvuk složen. Pomicanjem *Slidera* za zvuk, funkcija postavlja jačinu zvuka na *AudioMixeru* [38] (u kodu nazvan „am"), točnije na kanalu koji se zove *MainMixer*, *Master*. *MainMixer* kontrolira jačinu zvukova u igri te njega dodajemo objektima na *AudioSource* [39] komponentu preko inspektora, Slika 37. Ukoliko želimo da zvuk odmah krene po učitavanju objekta, označujemo opciju *Play On Awake*. Ta opcija nam je korisna ukoliko želimo dodati pozadinsku pjesmu na scenu.



Slika 37. Audio Source

Jednostavnim kodom mijenjamo i puštamo zvukove po potrebi. Primjer koda nalazi se u nastavku. Postavljamo zvuk *s_victory* kao zvuk koji želimo pustiti te *Play* funkcijom pokrećemo zvuk. Zvukovni efekti, kao i muzika, preuzeti su sa interneta.

```
this.GetComponent<AudioSource>().clip = s_victory;
this.GetComponent<AudioSource>().Play();
```

Skripta 19. Primjer promjene i puštanja zvuka

Zaključak

U ovom radu detaljno je predstavljen razvoj 2D edukativne igre u Unity razvojnom alatu u C# programskom jeziku. Grafika, odnosno spriteovi, preuzeti su s interneta, dodatno uređeni u Adobe Photoshop alatu i naposljetku animirani u Unityu. Video sa početka te zvukovi i muzika također su preuzeti sa interneta. Neki od zvukova dodatno su i uređeni u Adobe Audition alatu.

Ovaj rad detaljnije opisuje proces izrade 2D edukativne igre, sa ciljem da se nauči izrada složenijih vrsta 2D igara.

Na početku je objašnjeno sučelje Unity alata, njegove funkcije i prozori, sa ciljem kako bi alat bio jasniji i kako bi se lakše koristio. Potom su objašnjeni svi glavni elementi igre, odnosno scene, objekti te scene te skripte. Pronalaženje spriteova za različite likove i animiranje istih bio je najzahtjevniji dio te je taj proces trajao relativno dugo, stoga je animiranje detaljno pojašnjeno. Igra je, iako složena, napravljena na način da se jednostavnije nadograđuje i dodaju nove funkcije, što je zahtjevalo dodatno napora.

Bibliografski navod

[1] 4 rujan 2019. [Mrežno]. Available: https://en.wikipedia.org/wiki/Pong.

[2] [Mrežno]. Available: https://techjury.net/stats-about/gaming-industry-worth/. [Pokušaj pristupa 4 rujan 2019].

[3] [Mrežno]. Available: https://hr.wikipedia.org/wiki/Microsoft_Windows. [Pokušaj pristupa 4 rujan 2019].

[4] [Mrežno]. Available: https://hr.wikipedia.org/wiki/Linux. [Pokušaj pristupa 4 rujan 2019].

[5] [Mrežno]. Available: https://hr.wikipedia.org/wiki/MacOS. [Pokušaj pristupa 4 rujan 2019].

[6] [Mrežno]. Available: https://hr.wikipedia.org/wiki/C_sharp. [Pokušaj pristupa 4 rujan 2019].

[7] [Mrežno]. Available: https://visualstudio.microsoft.com. [Pokušaj pristupa 4 rujan 2019].

[8] [Mrežno]. Available: https://www.google.hr/search?biw=1638&bih=933&tbm=isch&sa=1&ei=e3RmXe-ZFvGmrgTXxp3IAQ&q=2d+wizard&oq=2d+wizard&gs_l=img.3..35i39j0i19l3j0i8i30i19.28027.28932..29020...0.0..0.175.721.0j5.....0....1..gws-wiz-img.tR93_5b4iXU&ved=0ahUKEwivw_22yaXkAhVxk4sKHVd. [Pokušaj pristupa 4 rujan 2019].

[9] [Mrežno]. Available: https://en.wikipedia.org/wiki/Bookworm_Adventures. [Pokušaj pristupa 4 rujan 2019].

[10] [Mrežno]. Available: https://en.wikipedia.org/wiki/Unity_(game_engine). [Pokušaj pristupa 4 rujan 2019].

[11] [Mrežno]. Available: https://unity3d.com/unity/features/multiplatform. [Pokušaj pristupa 4 rujan 2019].

[12] [Mrežno]. Available: https://en.wikipedia.org/wiki/Boo_(programming_language). [Pokušaj pristupa 4 rujan 2019].

[13] [Mrežno]. Available: https://hr.wikipedia.org/wiki/JavaScript. [Pokušaj pristupa 4 rujan 2019].

[14] [Mrežno]. Available: https://docs.unity3d.com/Manual/CreatingScenes.html. [Pokušaj pristupa 4 rujan 2019].

[15] [Mrežno]. Available: https://docs.unity3d.com/Manual/Hierarchy.html. [Pokušaj pristupa 4 rujan 2019].

[16] [Mrežno]. Available: https://docs.unity3d.com/Manual/class-GameObject.html. [Pokušaj pristupa 4 rujan 2019].

[17] [Mrežno]. Available: https://docs.unity3d.com/Manual/UsingTheInspector.html. [Pokušaj pristupa 4 rujan 2019].

[18] [Mrežno]. Available: https://learn.unity.com/tutorial/creating-properties. [Pokušaj pristupa 4 rujan 2019].

[19] [Mrežno]. Available: https://docs.unity3d.com/Manual/GameView.html. [Pokušaj pristupa 4 rujan 2019].

[20] [Mrežno]. Available: https://www.youtube.com/watch?v=SBsXEYJngzU. [Pokušaj pristupa 4 rujan 2019].

[21] [Mrežno]. Available: https://forum.unity.com/threads/autofade-doesnt-work-after-building-project.251674/. [Pokušaj pristupa 4 rujan 2019].

[22] [Mrežno]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html. [Pokušaj pristupa 4 rujan 2019].

[23] [Mrežno]. Available: https://docs.unity3d.com/ScriptReference/AudioSource.html. [Pokušaj pristupa 4 rujan 2019].

[24] [Mrežno]. Available: https://en.wikipedia.org/wiki/Sprite_(computer_graphics). [Pokušaj pristupa 4 rujan 2019].

[25] [Mrežno]. Available: https://helpx.adobe.com/uk/photoshop/using/layer-basics.html. [Pokušaj pristupa 4 rujan 2019].

[26] [Mrežno]. Available: https://www.quora.com/Adobe-Photoshop-What-are-the-differences-between-a-PSD-and-a-PSB-file. [Pokušaj pristupa 4 rujan 2019].

[27] [Mrežno]. Available: https://docs.unity3d.com/Packages/com.unity.2d.psdimporter@1.2/manual/index.html. [Pokušaj pristupa 4 rujan 2019].

[28] [Mrežno]. Available: https://blogs.unity3d.com/2018/11/09/getting-started-with-unitys-2d-animation-package/. [Pokušaj pristupa 4 rujan 2019].

[29] [Mrežno]. Available: https://blogs.unity3d.com/2018/12/04/getting-started-with-2d-inverse-kinematics/. [Pokušaj pristupa 4 rujan 2019].

[30] [Mrežno]. Available: https://docs.unity3d.com/Manual/SpriteEditor.html. [Pokušaj pristupa 4 rujan 2019].

[31] [Mrežno]. Available: https://docs.unity3d.com/Packages/com.unity.2d.animation@2.2/manual/SkinningEditor.html. [Pokušaj pristupa 4 rujan 2019].

[32] [Mrežno]. Available: https://docs.unity3d.com/ScriptReference/Animation.html. [Pokušaj pristupa 4 rujan 2019].

[33] [Mrežno]. Available: https://docs.unity3d.com/ScriptReference/Animator.html. [Pokušaj pristupa 4 rujan 2019].

[34] [Mrežno]. Available: https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-free-109565. [Pokušaj pristupa 4 rujan 2019].

[35] [Mrežno]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html. [Pokušaj pristupa 4 rujan 2019].

[36] [Mrežno]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.InvokeRepeating.html. [Pokušaj pristupa 4 rujan 2019].

[37] [Mrežno]. Available: https://docs.unity3d.com/Manual/script-Slider.html. [Pokušaj pristupa 4 rujan 2019].

[38] [Mrežno]. Available: https://docs.unity3d.com/Manual/AudioMixer.html. [Pokušaj pristupa 4 rujan 2019].

[39] [Mrežno]. Available: https://docs.unity3d.com/ScriptReference/AudioSource.html. [Pokušaj pristupa 4 rujan 2019].

Popis slika

Slika 1. Unity sučelje 2
Slika 2. Hijerarhijski prikaz objekata 3
Slika 3. Inspector 3
Slika 4. Play, Pause i Step gumbovi 4
Slika 5. Game prozor 4
Slika 6. Alatna traka 4
Slika 7. Avantura mod igranja 5
Slika 8. Mod igranja - Utrka s vremenom 6
Slika 9. Glavni izbornik 9
Slika 10. Povezivanje funkcije sa gumbom 10
Slika 11. Avantura mod 12
Slika 12. Trgovina napitaka 15
Slika 13. Sprite neprijatelja 15
Slika 14. Dijelovi tijela neprijatelja 16
Slika 15. Instalirani paketi 16
Slika 16. Prikaz PSB datoteke u projektu 17
Slika 17. Sprite Editor 17
Slika 18. Struktura kostiju zmaja 18
Slika 19. Kosti koje utječu na gornji dio glave 18
Slika 20. Nepravilno razvlačenje spriteova 19
Slika 21. Pomicanje kostiju zmaja 19
Slika 22. Sučelje za animiranje 20
Slika 23. Pomicanje zglobova po vremenskoj crti 21
Slika 24. Animator 21
Slika 25. Animation controller 22
Slika 26. Postavke za prijelaz animacije 22
Slika 27. Collider neprijatelja 24
Slika 28. Postavke collidera 24
Slika 29. Prozor za završeni level 27
Slika 30. Prozor za level up 27
Slika 31. Prozor čestitke 28
Slika 32. Prozor kad neprijatelj pobjedi 28
Slika 33. Utrka s vremenom 29
Slika 34. Zaslon poslije isteka vremena 33
Slika 35. Medalje 34
Slika 36. Opcije 35
Slika 37. Audio Source 35

Popis skripti

Skripta 1. Kod za učitavanje ili kreiranje podataka igrača 8
Skripta 2. Pritisak na gumb „Trgovina" u glavnom izborniku.....9
Skripta 3. Funkcija promjene scene 10
Skripta 4. Funkcija Start u GameManagment skripti.....11
Skripta 5. Funkcija za učitavanje prikladne pozadine.....11
Skripta 6. Funkcija za generiranje nasumičnih slova.....13
Skripta 7. Funkcija za provjeru sastavljenih riječi 14
Skripta 8. Funkcija za provjeru kolizija 23
Skripta 9. Filtriranje riječi 25
Skripta 10. Funkcija za provjeru postoje li sva potrebna slova.....25
Skripta 11. Funkcija za osvježavanje prikazanih podataka na ekranu 26
Skripta 12. Start Funkcija u GameManagment_TimeAttack skripti.....30
Skripta 13. Funkcije za odbrojavanje na početku 30
Skripta 14. Funkcija za glavno odbrojavanje u Utrci s vremenom.....31
Skripta 15. Funkcija dodaj vrijeme 32
Skripta 16. Funkcija za provjeru novoosvojenih medalja.....33
Skripta 17. Primjer teksta medalje.....34
Skripta 18. Funkcije za kontrolu opcija 35
Skripta 19. Primjer promjene i puštanja zvuka 36

Popis priloga

Uz ovaj rad priložen je CD na kojemu se nalazi ovaj rad, igra te sve komponente koje su korištene za izradu igre.