2D računalna igra

Belić, Stefan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni prijediplomski studij Računarstva

2D RAČUNALNA IGRA

Završni rad

Stefan Belić

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Stefan Belić
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R4464, 27.07.2020.
JMBAG:	0165085400
Mentor:	izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	2D računalna igra
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	[Rezervirano: Stefan Belić] Potrebno je izraditi računalnu igru koja će omogućiti igraču izbjegavanje raznih prepreka na koje nailazi (preskakanjem, saginjanjem), te pri tome i skupljanje određenih bodova s obzirom na način kako je prepreka izbgjegnuta. Igra bi trebala s vremenom povećavati težinu na način da se prepreke kreću brže i/ili da su teže za izbjeći. Cilj igre je izdržati što duže i pri tome skupiti što više bodova.

Datum prijedloga ocjene završnog rada od strane mentora:	16.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	25.09.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	26.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 26.09.2024.

Ime i prezime Pristupnika:	Stefan Belić
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4464, 27.07.2020.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: 2D računalna igra**

izrađen pod vodstvom mentora izv. prof. dr. sc. Alfonzo Baumgartner

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

Razvoj računalnih igara danas je vrlo rasprostranjena grana programskog inženjerstva. Industrija igara često se povezuje s kompleksnim 3D naslovima i izuzetnim mehanikama te visokom rezolucijom, jednostavnije 2D igre također pružaju korisnicima užitak zbog svoje pristupačnosti, zabavnosti i izazovima koje pružaju igračima. Ovaj završni rad opisuje razvoj jedne takve 2D računalne igre gdje je glavni lik kvadrat koji preskače nadolazeće prepreke i s vremenom postaje zahtjevnije. Karakteristika igre je skupiti što veći broj bodova. Igra je razvijena korištenjem Unity razvojne okoline i programskog jezika C#, pripada kategoriji igara beskonačnog trčanja i naziv je Jumpy Jumper. Cilj ovog rada bio je istražiti i primijeniti temeljne koncepte razvoja igara kroz kreiranje funkcionalne i zabavne računalne igre. U radu su obrađeni svi ključni aspekti razvoja, uključujući konceptualizaciju igre, dizajn likova te implementacija fizike i kolizija. Grafika je izrađena u web aplikaciji Piskel, zvučni efekti [1] i melodija [2] su preuzeti sa stranica slobodnog sadržaja. Zvučni efekti su dodatno uređeni u AudioMass web aplikaciji.

Druga točka rada navodi postojeća rješenja u kojoj se obrađuju popularne igre koje dijele slične mehanike s razvijenom igrom. Sličnost kod navedenih rješenja i razvijene igre je način stvaranja prepreka, kretanje prepreka prema igraču i povećanje težine prolaskom vremena te trajanje igre sve dok se igrač ne sudari sa preprekom. Treća točka opisuje tehnologije koje su korištene pri izradi ovog projekta. Četvrta točka detaljno opisuje razvoj igre, proces od ideje, dizajna glavnog lika, prepreka i bonusa, pa sve do pozadinskog dizajna. Posebna pažnja posvećena je sustavu praćenja i spremanja rezultata, kao i dizajnu korisničkog sučelja koje omogućuje intuitivnu interakciju s igrom. Na kraju, zaključak sažima cjelokupni proces i doprinos korištenih tehnologija u izradi igre.

1.1. Zadatak završnog rada

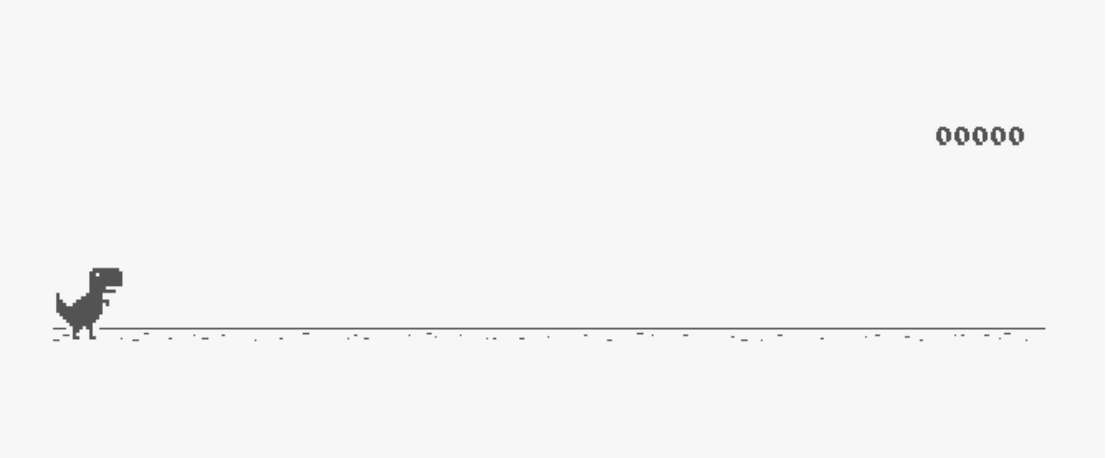
Potrebno je izraditi računalnu igru koja će omogućiti igraču izbjegavanje raznih prepreka na koje nailazi (preskakanjem, saginjanjem), te pri tome i skupljanje određenih bodova s obzirom na način kako je prepreka izbjegnuta. Igra bi trebala s vremenom povećavati težinu na način da se prepreke kreću brže i/ili da su teže za izbjeći. Cilj igre je izdržati što duže i pri tome skupiti što više bodova.

2. POSTOJEĆA RJEŠENJA

Prve 2D računalne igre su se pojavile već u ranim 1970-im godinama kada su tadašnji programeri imali ograničene tehnologije i znanja. Godinama se to znanje širilo i tehnologija ubrzano poboljšavala. Videoigre su postajale popularnije među ljudima stoga se više ljudi počinjalo baviti i razvojem računalnih igara. Od 1970-te pa sve do danas se pojavilo neopisivo mnogo videoigara sa različitim mehanikama i pričama. Prema tome postoji i mnogo beskonačno trčećih računalnih igara koje su svojom jedinstvenom logikom beskonačnog igranja pružile zabavu korisnicima. Analizirana su online rješenja pronađena na web stranici Poki [3].

2.1. Dinosaur Game

Prva analizirana računalna igra koja radi na principu beskonačnosti je *Dinosaur Game* [4]. Slika 2.1. prikazuje izgled videoigre



Sl. 2.1. Izgled videoigre Dinosaur Game

Ova igra je ugrađena u Google Chrome preglednik i automatski se pokreće kada korisnik nema pristup internetu. Korisnik kontrolira malog dinosaura (T-Rex) koji trči kroz pustinju. Cilj igre je preskakati kaktuse i druge prepreke koje se pojavljuju na putu, kako bi se postigao što veći rezultat. Igra je minimalističkog dizajna, s crno-bijelom grafikom, i postala je prepoznatljiv simbol za korisnike interneta kada dođe do prekida mreže.

2.2. Jetpack Joyride

Sljedeća videoigra koja nudi princip beskonačnog trčanja je *Jetpack Joyride* [5]. Slika 2.2. prikazuje scenu iz tijeka videoigre.

Sl. 2.2. Scena iz videoigre Jetpack Joyride

Korisnik kontrolira lika po imenu Barry Steakfries, koji koristi razne vrste mlaznih ranaca kako bi izbjegavao prepreke i prikupljao novčiće uz mogućnosti kretanja gore i dolje. Cilj igre je postići što veći rezultat prije nego što igrač naleti na prepreku, poput električnih barijera ili projektila. Igra ima živopisnu grafiku, brzo kretanje i razne vizualne efekte, što doprinosi dinamičnosti.

2.3. Temple Run

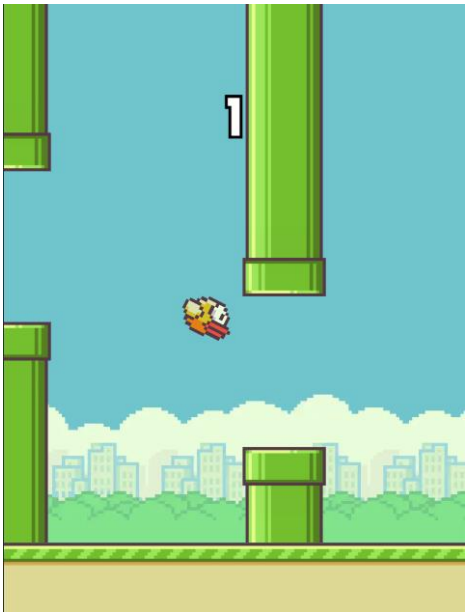
Videoigra *Temple Run 2* [6] je drugačija od prijašnjih jer igrač ima mogućnost skretanja lijevo i desno uz mogućnosti saginjanja i skakanja. Slika 2.3. prikazuje scenu iz videoigre.

Sl. 2.3. Scena iz videoigre Temple Run

U ovoj igri igrač preuzima ulogu istraživača koji bježi kroz drevni hram, izbjegavajući različite prepreke, skreće u pravom trenutku, preskače i kliže ispod prepreka te sakuplja novčiće. Igra je beskonačna, što znači da traje dok god igrač može izbjegavati prepreke, a cilj je postići što veći rezultat prije nego što igra završi. Temple Run 2 je jedna od igara koja je popularizirala žanr beskonačnog trčanja na mobilnim uređajima.

2.4. Flappy Bird

Najpopularnija od beskonačno trčećih videoigara je *Flappy Bird* [7]. Slika 2.4. prikazuje scenu iz igre *Flappy Bird*, jednostavne, ali izuzetno popularne mobilne igre koja je postala globalna atrakcija brzo nakon pojavljivanja. Igrač kontrolira lika (Flappy Bird) koji mora proći kroz uske prolaze između zelenih prepreka bez dodirivanja. Prepreke se generiraju jedna za drugom sa slučajno generiranim visinama prolaza. Ova videoigra je poznata po svojoj visokoj razini težine, vrlo je teško postići veliki rezultat zbog preciznosti potrebne za kontrolu leta ptice što je kod igrača izazivalo frustracije te dodatno doprinosilo popularnosti ove igre.



Sl. 2.4. Izgled videoigre Flappy Bird

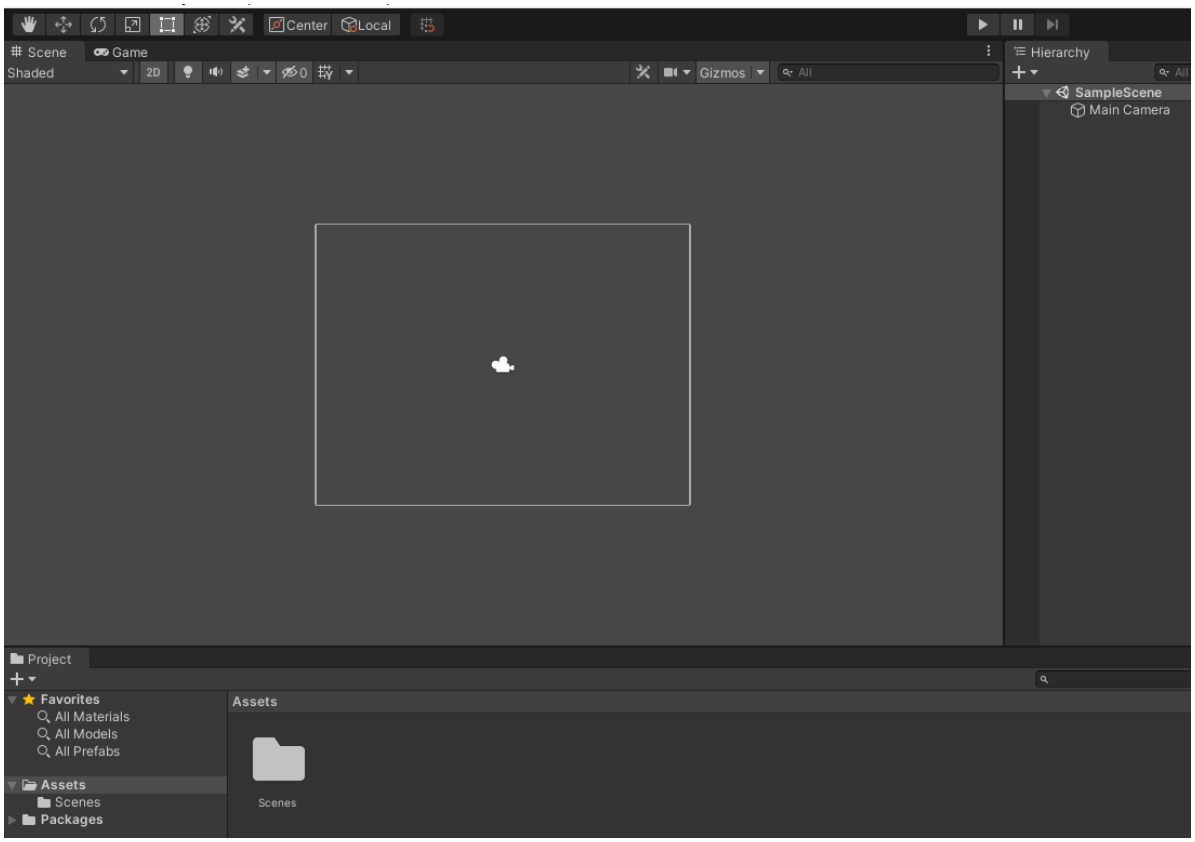
Cilj igre je postići što veći broj bodova, prolazeći kroz što više prepreka. Igra se završava čim ptica dotakne jednu od prepreka ili padne na tlo.

3. KORIŠTENE TEHNOLOGIJE

Tehnologije korištene u razvoju igre uključuju Unity Engine, gdje je sama igra kreirana, te C# programski jezik za pisanje skripti. Za razvoj su također korišteni alati poput Visual Studio Code za uređivanje koda, Piskel za izradu likova te AudioMass za uređivanje zvuka.

3.1. Unity

Unity je trenutno jedan od najpopularnijih i najkorisnijih alata koji služi za razvoj 2D ili 3D videoigara, aplikacija i simulacija. Pojavio se 2004. godine od strane tvrtke *Unity Technologies* [8] te je evoluirao od jednostavnog alata za razvoj igara do platforme koja obuhvaća razne industrije, uključujući arhitekturu, medicinu, obrazovanje, automobilski dizajn i filmskim industrijama. U industriji videoigara, koristi se za razvoj igara svih žanrova. U arhitekturi, Unity se koristi za stvaranje interaktivnih 3D prikaza zgrada i interijera, omogućujući klijentima da se postave kroz buduće prostore. U automobilskoj industriji, koristi se za simulacije i vizualizacije novih modela automobila. U obrazovanju, *Unity* omogućuje stvaranje edukativnih igara i simulacija koje pomažu učenicima da lakše nauče gradivo kroz interaktivno iskustvo. Filmska industrija koristi *Unity* kako bi u animiranim filmovima stvorila pozadinske slike. Ključna značajka Unityja je njegova jednostavnost za korištenje, čak i za početnike koji se prvi puta susreću sa razvojem igara. Korisničko sučelje omogućuje programerima i dizajnerima da lako upravljaju objektima, kamerama i svjetlom, dok razvijeni sustav za skriptiranje, temeljen na programskom jeziku C#, omogućava visoku fleksibilnost u implementaciji logike igre i drugih interaktivnih funkcionalnosti. *Unity* podržava više od 25 različitih operacijskih sustava uključujući Windows, macOS, iOS, Android, PlayStation i ostale. Poznat po svojim naprednim grafičkim mogućnostima, omogućuje obradu visoke kvalitete, uključujući napredne efekte osvjetljenja, sjenčanja i post-processinga, što omogućava kreiranje vizualno interesantnijih, zanimljivijih igara i aplikacija. Također, *Unity* uključuje ugrađeni fizikalni engine koji simulira zakone fizike, što doprinosi stvaranju autentičnih interakcija između objekata u 3D okruženju. Velika prednost Unityja je njegova zajednica korisnika i razvijen ekosustav. *Unity Asset Store* [9] omogućuje pristup nekoliko tisuća resursa, uključujući modele, teksture, skripte i alate, što poprilično ubrzava proces razvoja. Aktivna zajednica također pruža podršku kroz forume, komentare, instrukcije i online tečajeve, što predstavlja *Unity* kao idealni alat za učenje i profesionalni razvoj u području dizajna i razvoja igara. Slika 3.1. prikazuje sučelje koje sadrži prozore za obavljanje različitih poslova u izradi igre.

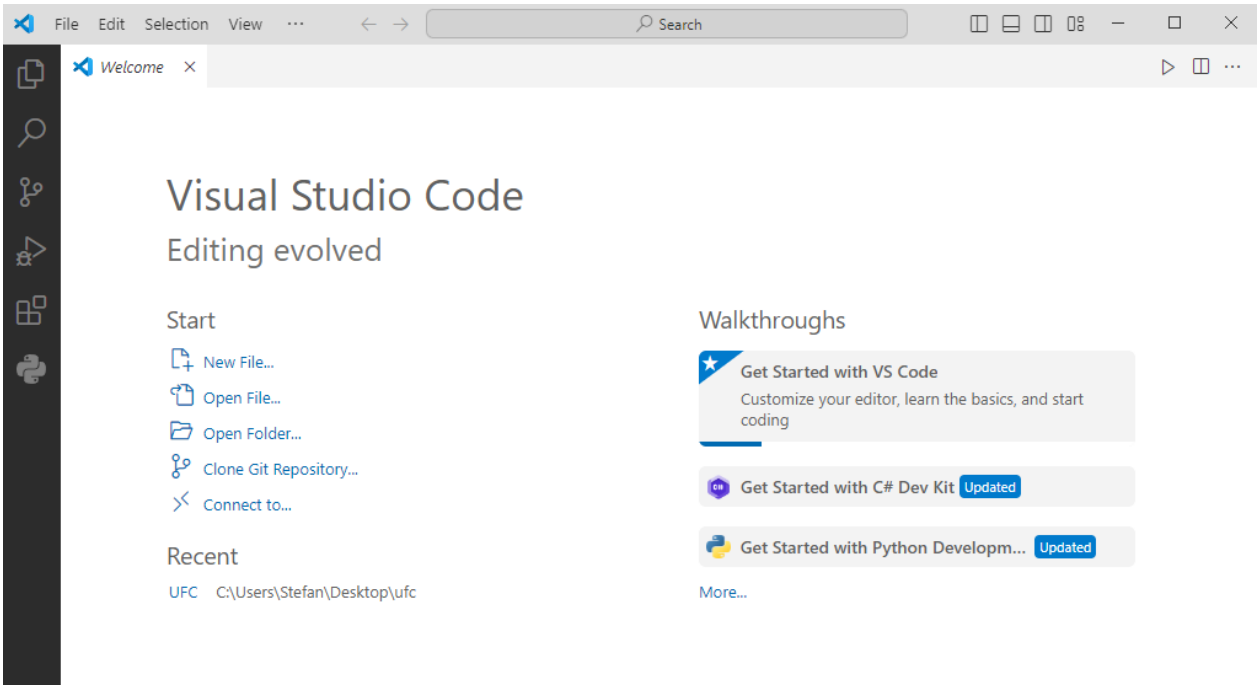


Sl. 3.1. Prikaz sučelja iz programa

Glavni prozor je scena u kojem se postavljaju objekti te se kreira videoigra. Na sceni se nalazi ono što vidimo kada pokrenemo videoigru. Većinom slučajeva igra se sastoji od više scena, npr. za svaku razinu jedna scena. Objekti su osnovne cjeline od kojih se izrađuje igra te ih možemo vidjeti u prozoru hijerarhija. Objektima možemo dodati grafike, zvukove, skripte te ponašanja.

3.2. Visual Studio Code

Visual Studio Code je snažni, besplatni uređivač koda koji je razvio *Microsoft* 2015. godine. Ubrzo je postao jedan od najpopularnijih razvojnih alata na svijetu zbog podrške za različite programske jezike i tehnologije. *Visual Studio Code* omogućuje brzo otvaranje datoteka, pretragu kroz projekte i pokretanje jednostavnih zadataka bez potrebe za pokretanjem cijelog integriranog razvojnog okruženja. *Visual Studio Code* dolazi s ugrađenom podrškom za JavaScript, TypeScript i Node.js, ali se može proširiti na stotine drugih jezika poput Python, C++, Java, C#, PHP i mnogih drugih, putem proširenja dostupnih u *Visual Studio Marketplaceu* [10]. Dolazi s integriranim terminalom, što omogućuje programerima da pokreću naredbe izravno iz uređivača bez potrebe za prelaskom na vanjski terminal. Slika 3.2. predstavlja početnu stranicu razvojnog okruženja Visual studio.

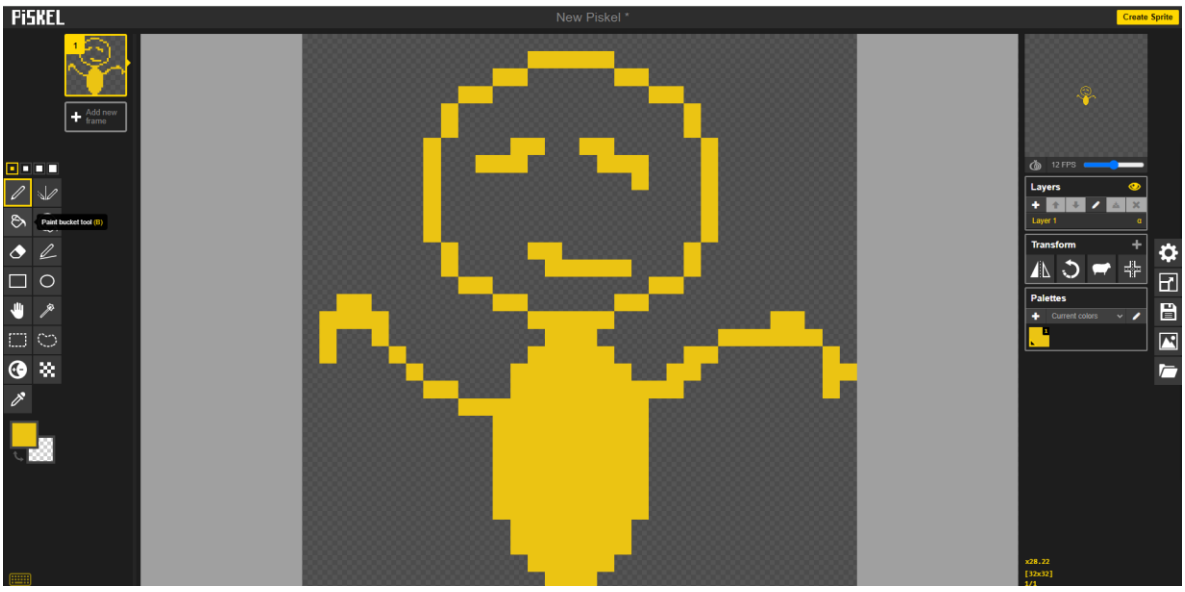


Sl. 3.2. Razvojno okruženje *Visual Studio Code*

Prikazuje se naslov *Visual Studio Code* s podnaslovom *Editing evolved*, što naglašava njegovu svrhu kao naprednog uređivača koda. Unutar izbornika se nalaze opcije za kreiranje nove datoteke, otvaranje već postojeće datoteke, otvaranje uređivača i kloniranje git repozitorija. Poveži se s...; omogućuje povezivanje na udaljene servere ili servise. "Recent" sekcija ispod opcija za početak rada, koja prikazuje nedavno otvorene projekte ili datoteke. U ovom slučaju, prikazuje nedavno otvoreni projekt pod nazivom UFC. Projekt je rađen u C# programskom jeziku. C# je moderan, objektno orijentirani programski jezik razvijen od strane Microsofta kao dio .NET platforme. C# je od kada se pojavio 2000. godine postao jedan od najraširenijih jezika u industriji, posebno za izradu web aplikacija, mrežnih usluga i programske podrške na razini poduzeća. Vrlo slični jezici su Java, C, C++. C# nudi razne značajke koje znatno olakšavaju izradu aplikacije i korištenje memorije. Prednost programskog jezika C# je ta da programski jezik sam raspodjeljuje memoriju i oslobađa od nekorištenih objekata. C# omogućuje polimorfizam, nasljeđivanje i enkapsulaciju koji su temelji objektno orijentiranog programiranja.

3.3. Piskel

Piskel je web aplikacija dizajnirana da bude jednostavna za korištenje, omogućujući korisnicima da se usredotoče na kreiranje umjetničkih djela bez potrebe za učenjem kompliciranih alata. Alat omogućava jednostavno kreiranje animacija ili jedinstvenih slika. Svaka slika se može uređivati pojedinačno, a alat nudi mogućnosti pregleda animacije u stvarnom vremenu. Piskel je najčešće korišten u razvoju indie igara, gdje je pikselski stil popularan zbog svoje nostalgične vrijednosti i jednostavnosti. Korisnici mogu sa lakoćom kreirati *spriteove* za likove, objekte, pozadine i druge elemente igara. Alat je također koristan za izradu jednostavnih animacija koje se mogu koristiti na web stranicama, u mobilnim aplikacijama ili u digitalnim umjetničkim projektima. Slika 3.3. prikazuje radni prostor web aplikacije Piskel.



Sl. 3.3. Radni prostor Piskela

Na vrhu se nalazi naziv datoteke, u ovom slučaju standardni naziv *New Piskel*, a desno opciju za kreiranje novog *spritea*. U sredini je prostor za crtanje sa izbornikom potrebnih alata koji se nalazi lijevo od radnog prostora. Alati koji su dostupni su olovka, gumica, alat za ispunjavanje boja, alat za crtanje linija, krugova i pravokutnika, alat za selekciju i pomicanje, pipeta za odabir boje i alat za zrcaljenje slike. Iznad izbornika za alate nalazi se opcija dodavanja nove slike koje se automatski povežu u .gif datoteku prilikom dodavanja. Ova opcija služi za kreiranje animacija, pokrete likova i slično. Ova web aplikacija koristila je za kreiranje spriteova za ovaj projekt. Desno od radnog prostora postoji opcija za podešavanje prikaza broja slika u sekundi. Prilikom završetka postoji mogućnost spremanja slike, odnosno animacije u različitim formatima.

3.4. AudioMass

AudioMass je besplatni, online alat za uređivanje zvuka koji je otvorenog koda. Dizajniran je za jednostavno korištenje i radi izravno u web pregledniku, bez potrebe za instaliranjem bilo kakvog zahtjevnijeg programa. Namijenjen je korisnicima koji trebaju brzo urediti audio datoteke bez potrebe za kompleksnim alatima ili aplikacijama. Podržava uvoz i izvoz zvuka u raznim formatima, uključujući WAV i MP3 te je zbog toga fleksibilan za različite projekte. AudioMass omogućuje korisnicima uređivanje zvuka u stvarnom vremenu, s trenutnim pregledom svih promjena koje se primjenjuju na audio datoteku. Alat uključuje osnovne audio efekte kao što su normalizacija, promjena tempa i filtriranje visokih i niskih frekvencija. AudioMass je dostupan na bilo kojem uređaju s modernim preglednikom, uključujući računala, tablete i pametne telefone. Kao projekt otvorenog koda, AudioMass je dostupan zajednici za prilagodbu i poboljšanje, a korisnici mogu slobodno pridonijeti njegovom razvoju. Sučelje AudioMass web aplikacije za uređivanje zvuka prikazano je na slici 3.4.

Sl. 3.4. Sučelje AudioMassa

Na sučelju web aplikacije može se vidjeti prikaz zvučnog zapisa u formi valnog oblika (engl. *waveform*) i osnovne kontrole za uređivanje audio datoteke. Unutar središnjeg dijela sučelja prikazuje se valni oblik zvučnog zapisa koji se uređuje, u ovom slučaju pozadinska melodija unutar ovog projekta. Valni oblik prikazuje amplitudu zvuka tijekom vremena za oba kanala (lijevi i desni kanal), što je tipično za stereo zapise. Gornji valni oblik prikazuje lijevi kanal, dok donji prikazuje desni kanal. Iznad valnog oblika nalaze se osnovne kontrole za upravljanje reprodukcijom zvuka kao što su gumb za pokretanje ili zaustavljanje reprodukcije zvuka, stop koji zaustavlja reprodukciju i vraća na početak. Gumb koji služi za brzo premotavanje naprijed ili natrag, gumb za snimanje, koji služi korisnicima da dodaju novi zvuk u postojeći zapis i funkcije za poništavanje ili vraćanje posljednjih akcija. Također ima mogućnost označivanja bilo kojeg dijela zvuka uz pomoć miša i pritiskom tipke za brisanje na tipkovnici, automatsko brisanje označenog dijela zvuka. Ispod kontrola za reprodukciju nalazi se traka gdje se prikazuje trenutni položaj u zvučnom zapisu. Traka omogućuje korisnicima da skoče na određeno vrijeme u zapisu klikom na odgovarajuću poziciju. Na vrhu sučelja nalaze se izbornici koji pružaju dodatne funkcije, poput spremanja datoteke, primjene efekata na zvuk, ili pomoć za korisnike. Na desnoj strani prikazana su polja za unos točnog vremena početka, kraja i trajanja odabranog dijela zvuka, što pomaže korisniku da precizno uređuje zvuk.

4. RAZVOJ RAČUNALNE IGRE

U ovoj točki detaljno se opisuje proces razvoja računalne igre od početne ideje, izgleda, tehničkih detalja i spremanja rezultata te prikaza korisničkog sučelja.

4.1. Zamisao

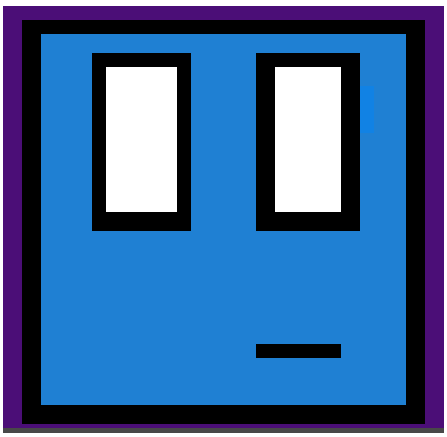
Ideja projekta bila je stvoriti funkcionalnu i zabavnu 2D računalnu igru koja će biti beskonačna, odnosno sve dok igrač ne izbjegne određenu prepreku. Glavni izvor ideja potiče iz starijih i jednostavnijih videoigara koje su koristile slične algoritme za stvaranje prepreka i povećanje težine uz veći vremenski interval. Glavni lik je plavi kvadrat koji se kreće po tlu i ima zadatak izbjeći što je više moguće prepreka i postići visok krajnji rezultat, prikupljajući uz to zlatne novčiće koji dodavaju bonus bodova ukoliko su pokupljeni. Igra omogućuje kratko skakanje, dugačko skakanje i saginjanje. Igraču je zadatak dobro procijeniti kada treba iskoristiti skok i koliko visoko smije skočiti kako ne bi pao na sljedeću prepreku. Unutar igre ne postoje određeni neprijatelji već samo različite vrste prepreka, od onih koje se nalaze na podu do onih koje su u zraku, malih i velikih. Igra se sastoji od početnog zaslona, izbornika unutar kojega je opcija za početak igranja. Prilikom stiska na početak, igra započinje i rezultat počinje da se povisuje. Nakon završetka igre pojavljuje se drugi izbornik koji daje opcije za ponovno igranje i opciju za odlazak na glavni početni izbornik.

4.2. Izgled i tehnički detalji

Opisuje se izgled i funkcionalnost glavnog lika, prepreka i promjena boje pozadine.

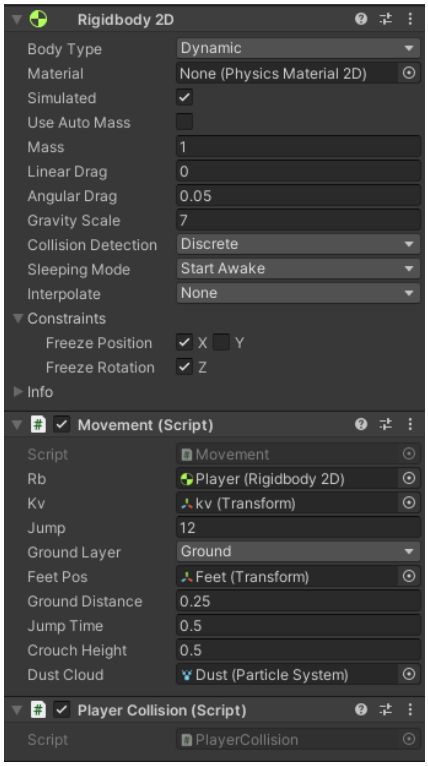
4.2.1. Glavni lik

Glavni lik ove računalne igre je plavi kvadrat prikazan na slici 4.1. Korisnik računalne igre ga kontrolira s mogućnostima skoka i saginjanja.



Sl. 4.1. Prikaz glavnog lika

Na slici 4.2. se vidi kako je na glavnog lika primijenjena komponenta *Rigidbody 2D* i dvije prilagođene skripte: *Movement* (slika 4.3.), koja je odgovorna za mogućnosti skakanja i saginjanja i *Player Collision* (slika 4.4.) koja uništava objekt igrača te određuje kraj igre.



Sl. 4.2. Scena unutar objekta glavnog lika

RigidBody 2D je standardna komponenta u Unity-u koja omogućava da objekt bude pod utjecajem fizike u 2D svijetu (poput gravitacije i sudara). *Body Type* je postavljen na Dynamic, što znači da se ovaj objekt aktivno simulira pomoću Unity-jevog fizičkog sustava (kretanje, gravitacija itd.) Nema dodijeljenog fizičkog materijala (*None*), što znači da se koriste standardne postavke bez dodatnih efekata poput trenja ili odskoka. *Simulated* opcija je označena, što govori da je fizička simulacija aktivna za ovaj objekt. *Use Auto Mass* opcija nije označena, što znači da se masa ne određuje automatski, već je ručno postavljena. Masa objekta je postavljena na 1, što utječe na to kako se objekt ponaša pod utjecajem fizike, uključujući i gravitaciju. Koja je postavljena na 7, što znači da gravitacija koja djeluje na ovaj objekt ima 7 puta veću snagu od standardne gravitacije u igri. *Collision Detection* je postavljeno na *Discrete*, što je standardna metoda detekcije sudara koja je pogodna za objekte koji se kreću sporije. Objekt odmah aktivan i reagira na fizičke sile čim igra počne.

Opcije za zaključavanje rotacije su označene, govori da je objekt zaključan po X osi pa se ne može kretati u tim smjerovima. Također je zaključana rotacija objekta na Z osi. Skripta za kretanje pod nazivom *Movement* (eng. Kretanje) sadrži sva potrebna polja koja omogućuju funkcionalan rad računalne igre. *Rb* referencira na igračevu *Rigidbody 2D* komponentu, kako bi skripta mogla manipulirati fizičkim karakteristikama objekta igrača. *Kv* opisuje transformaciju objekta, koristi se za smanjivanje slike igrača prilikom čučnja. Vrijednost skoka postavljena je na 12, što definira snagu skoka, odnosno koliko visoko lik može skočiti prilikom pritiska gumba za skok. Na *Ground Layer* (eng.) postavljeno je *Ground* (eng.), što označava sloj koji se koristi za detekciju tla, kako bi skripta znala kada je igrač na zemlji. *Feet Pos* (eng.) je referenca na poziciju stopala igrača (predstavlja točku koja se koristi za provjeru je li igrač na tlu). Postavljena vrijednost vremena za skok je 0,5, što određuje koliko dugo igrač može držati gumb za skok kako bi produžio skok. Visina saginjanja je postavljena na 0,5, igrač se smanji prilikom stiskanja tipke za saginjanje Također postoji *Dust Cloud* (eng.). Ovo referencira *Dust Particle System* (slika 4.5.), koji se koristi za stvaranje vizualnog efekta prašine prilikom skoka igrača.

```
private void Update(){

    isGrounded = Physics2D.OverlapCircle(feetPos.position, groundDistance,groundLayer);
    #region JUMP

    Debug.Log("Is Grounded: " + isGrounded);

    if(isGrounded && Input.GetButtonDown("Jump")){
        audioManager.PlaySFX(audioManager.jump);
        isJumping=true;
        rb.velocity = Vector2.up*jump;
        CreateDust();
    }
    if(isJumping && Input.GetButton("Jump")){
        if(jumpTimer<jumpTime){
            rb.velocity = Vector2.up* jump;
            jumpTimer += Time.deltaTime;
        } else{
            isJumping=false;
        }
    }
    if(Input.GetButtonUp("Jump")){
        isJumping=false;
        jumpTimer=0;
    }
    #endregion


    #region CROUCH
    if(isGrounded && Input.GetButtonDown("Crouch")) {
        audioManager.PlaySFX(audioManager.crouch);
        kv.localScale= new Vector3(kv.localScale.x, crouchHeight, kv.localScale.z);
    }
    if(isJumping && Input.GetButton("Crouch")){
        kv.localScale= new Vector3(kv.localScale.x, 1f, kv.localScale.z);
    }
    if(Input.GetButtonUp("Crouch")){
        kv.localScale= new Vector3(kv.localScale.x, 1f, kv.localScale.z);
    }
```

Sl. 4.3. Metoda *Update()* unutar *Movement* klase

Pomoću ove metode se vrši upravljanje skakanjem i saginjanjem. Igrač reagira na pritisak tipki za skok i saginjanje, a ujedno provjerava je li igrač na tlu kako bi omogućio pravilno funkcioniranje ovih akcija. Koristi se *Physics2D.OverlapCircle* da provjeri je li igračeva pozicija stopala u kontaktu s tlom. Kružnica određene veličine provjerava preklapanje s bilo kojim objektima na sloju tla. Ako se preklapanje dogodi, postavlja se *isGrounded* na *true*, inače je *false*. Omogućuje se skakanje igrača samo kada je na tlu. Kod skakanja se provjerava je li igrač u kontaktu sa tlom i je li pritisnuta tipka za skakanje. Ukoliko su oba uvjeta ispunjena reproducira se zvuk, i omogućuje se skok igraču, povećava brzina skoka i stvara efekt prašine. Ukoliko igrač drži tipku za skok i još uvijek je u skoku, nastavlja se kretanje prema gore sve dok trajanje skoka nije premašilo maksimalno dopušteno. Nakon isteka dozvoljenog vremena skok se prekida. Ukoliko igrač pusti tipku za skok u zraku, skok se prekida. Prilikom čučnja provjerava se je li igrač u dodiru sa tlom i je li pritisnuta tipka zadužena za čučanj. Ukoliko su uvjeti zadovoljeni reproducira se zvuk čučnja i postavlja se visina igrača na definiranu vrijednost koliko će igrač biti niži. Čučanj traje sve dok je tipka pritisnuta. Prilikom puštanja tipke igrač se vraća na normalnu visinu.

```
    0 references
    private void OnCollisionEnter2D (Collision2D other){
        if(other.transform.tag=="Obstacle"){
            audioManager.PlaySFX(audioManager.death);
            gameObject.SetActive(false);
            GameManager.Instance.GameOver();
        }
    }
}
```

Sl. 4.4. Metoda za sudar sa preprekama

Unutar *PlayerCollision* klase se nalazi logika prilikom sudara s objektom koji je označen kao prepreka. Skripta pokreće zvučni efekt za uništavanje lika. Zvučni efekt označuje kraj igre. Nakon što se zvučni efekt pokrene, igračev objekt postaje neaktivan, što znači da igračev objekt nestaje iz igre. Igrač više ne može upravljati tim objektom niti će biti vidljiv na ekranu. Time se jasno stavlja do znanja da je igra gotova ili da je igrač izgubio. Nakon što igrač izgubi pojavljuje se ekran sa prikazom rezultata, najveći postignutim rezultatom tijekom prijašnjih pokušaja i opcija za ponovno pokretanje igre ili odlazak na glavni izbornik.

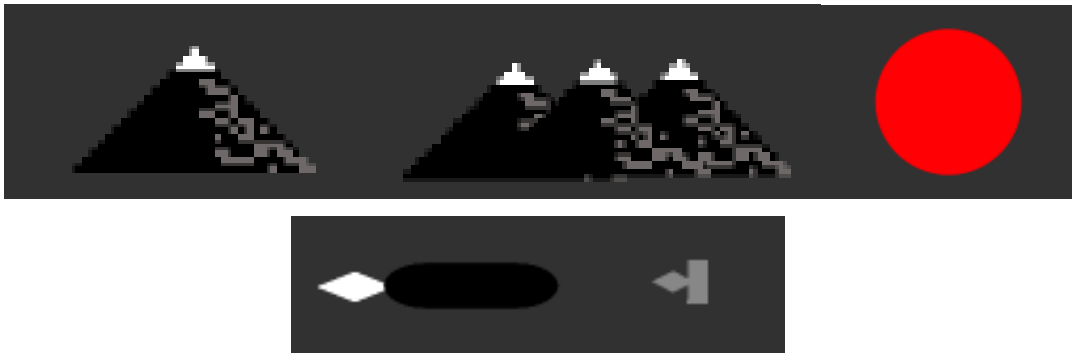
Prilikom skoka kreira se oblačić prašine koji je napravljen pomoću *Unity Particle System* koji olakšava kreiranje čestica za vizualno poboljšanje i stvaranje dodatnog ugođaja računalnih igara. Slika 4.5. prikazuje postavke za stvaranje prašine.

Sl. 4.5. Postavke sistema za kreiranje čestica

Trajanje je postavljeno na 10 sekundi i ponavljanje je uključeno. *Prewarm* je uključen jer se efekt aktivira odmah pri pokretanju. Vrijeme trajanja svake čestice je u intervalu od 0,5 sekundi do 1 sekunde. Početna brzina je postavljena na 3, a veličina na 0,2. Uključen je sistem za dodavanje gravitacije kako bi čestice lagano padale prema tlu. Po vremenskoj jedinici emitira se 50 čestica. Boja prašine je #C5C0C0 (197,192,192).

4.2.2. Prepreke

Prepreke u igri imaju ključnu ulogu u stvaranju izazova i određivanju krajnjeg cilja za igrača. To su objekti koji se stalno kreću prema igraču, a sudar s njima dovodi do završetka igre. Svaka prepreka nosi potencijalnu opasnost jer, ako igrač ne uspije izbjeći sudar, dolazi do trenutnog uništenja igrača. Neke prepreke se mogu samo preskočiti dok za druge je potrebno koristiti saginjanje. Prepreke su postavljene tako da se dinamično pojavljuju i kreću prema glavnom liku, stvarajući stalnu prijetnju. Kako igra napreduje, njihova brzina se povećava i vrijeme stvaranja smanjuje, čineći izbjegavanje zahtjevnijim. Igrač mora pažljivo procijeniti svoje pokrete, skačući preko prepreka ili izbjegavajući ih na vrijeme kako bi ostao što duže u igri. Na slici 4.6. prikazuju se sve prepreke koje se nalaze unutar igre.



Sl. 4.6. Prepreke

Prve dvije prepreke sa slike 4.6. su šiljci koji se uvijek nalaze na podu, potrebno ih je preskočiti. Crvena prepreka predstavlja metak koji putuje zrakom, moguće ga je zaobići preskakanjem ili saginjanjem. Koplje putuje zrakom, no nije moguće sagnuti se, već se mora preskočiti. Zadnja prepreka je raketa koja se izbjegava saginjanjem. Sve prepreke pripadaju sloju prepreka pa kontakt igrača sa njima dovodi do završetka igre.

Stvaranje i kretanje prepreka opisano je u klasi *Spawner* koja sadrži metodu *Update()* (slika 4.7.)

```
 private void Update()
{
if (GameManager.Instance.isPlaying)
{
    timeAlive += Time.deltaTime;
    timeSinceLastCoinSpawn += Time.deltaTime;
    CalculateFactors();
    SpawnLoop();
}
}
private void SpawnLoop(){
    timeUntilObstacleSpawn+= Time.deltaTime;
    if(timeUntilObstacleSpawn>= _obstacleSpawnTime){
        Spawn();
        timeUntilObstacleSpawn=0f;
    }
}
private void ClearObstacles(){
    foreach(Transform child in obstacleParent){
        Destroy(child.gameObject);
    }
}
private void CalculateFactors(){
     _obstacleSpawnTime = obstacleSpawnTime / Mathf.Pow(timeAlive, obstacleSpawnTimeFactor);
     _obstacleSpeed = obstacleSpeed * Mathf.Pow(timeAlive, obstacleSpeedFactor);

}

private void ResetFactors(){
    timeAlive = 1f;
    _obstacleSpawnTime = obstacleSpawnTime;
    _obstacleSpeed= obstacleSpeed;
}

private void Spawn()
{

GameObject obstacleToSpawn = obstaclePrefabs[Random.Range(0, obstaclePrefabs.Length)];
GameObject spawnedObstacle = Instantiate(obstacleToSpawn, transform.position, Quaternion.identity);
spawnedObstacle.transform.parent = obstacleParent;

Rigidbody2D obstacleRB = spawnedObstacle.GetComponent<Rigidbody2D>();
obstacleRB.velocity = Vector2.left * _obstacleSpeed;
```

Sl. 4.7. Algoritmi za stvaranje i ubrzavanje prepreka

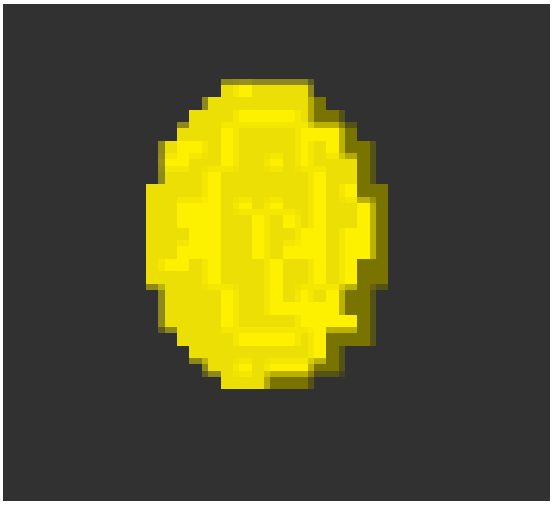
Tijekom svakog trenutka, dok je igra aktivna, bilježi se koliko je vremena prošlo od početka igre i koliko vremena je prošlo od posljednje generirane prepreke. Prepreke se pojavljuju u pravilnim intervalima, a brzina njihovog pojavljivanja i kretanja prilagođava se prema tome koliko dugo igra traje, čime se igra postupno ubrzava i postaje izazovnija. Osigurava se da se prepreke ne generiraju prečesto, a interval između prepreka se računa pomoću faktora koji uzima u obzir proteklo vrijeme igre. Također se uništavaju sve postojeće prepreke kako bi se očistila scena prilikom završetka igre kako bi se omogućilo ponovno kreiranje novih prepreka.

Kada se prepreka generira, nasumično se bira jedan objekt iz skupa definiranih prepreka. Taj se objekt tada kreira na sceni i postavlja kao dijete objekta koji služi kao roditelj za sve prepreke. Zatim se toj novo generiranoj prepreci dodjeljuje brzina prema kojoj će se kretati u lijevo (što simulira efekt dolaska prepreka prema igraču).

Ugrađeni su algoritmi koji računaju koliko će se vremenom brzina nadolazećih prepreka povećavati, brzina se dobiva eksponencijalno množenjem početne brzine sa proteklim vremenom koji je na potenciji definiranog faktora, u ovom slučaju 0,2. Smanjene vremena između stvaranja sljedećih prepreka dobiva se na isti način samo što se koristi operacija dijeljenja umjesto množenja.

4.2.3. Bonus

Računalna igra sadrži novčić (slika 4.8.) koji se pojavljuje na isti način kao prepreke. Vremenom češće i kreće se brže. Pojavljuje se uvijek visoko u zraku pa je potrebno skočiti što je više moguće kako bi se dohvatio.



Sl. 4.8. Zlatni novčić

Prikupljanjem ovog novčića povećavamo trenutni rezultat za 10 bodova. Na slici 4.9. prikazan je kod koji omogućuje ispravan rad prilikom dodira igrača sa novčićem.

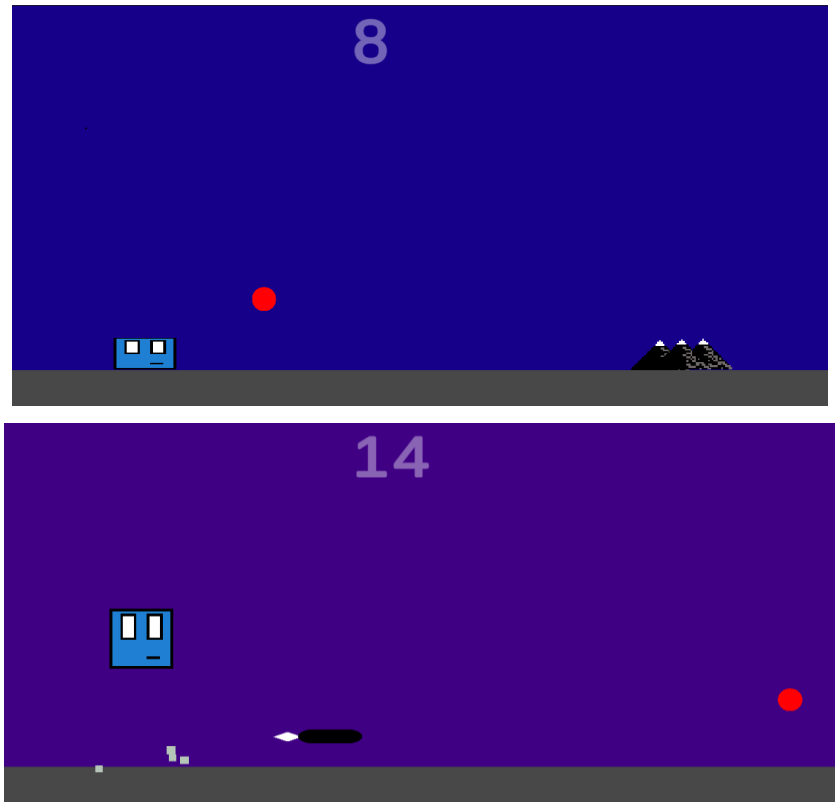
```
private void Awake(){
    audioManager = GameObject.FindGameObjectWithTag("Audio").GetComponent<AudioManager>();
}
private void OnTriggerEnter2D(Collider2D other)
{
    if (other.CompareTag("Player"))
    {
        audioManager.PlaySFX(audioManager.coin);
        GameManager.Instance.CollectCoin();
        Destroy(gameObject);
    }
}
```

Sl. 4.9. Metoda za dodir sa novčićem

Metoda *Awake()* omogućuje korištenje zvučnih funkcionalnosti, konkretno za reproduciranje zvučnih efekata. Druga metoda se pokreće svaki put kada neki drugi objekt u igri uđe u zonu novčić objekta. Provjerava je li objekt koji je ušao u zonu igrač (provjera oznake "*Player*"). Ukoliko je uvjet ispunjen, odnosno ako igrač pokupi novčić reproducira se zvuk skupljanja novčića. Poziva se metoda *CollectCoin()* za prikupljanje novčića koja dodjeljuje dodatnih 10 bodova u trenutnom rezultatu. Nakon što je novčić prikupljen, objekt se uništava (nestaje iz igre).

4.2.4. Pozadina

Pozadina se sastoji od izmjene triju različitih boja. Izabrane su tamnije boje zbog bolje vizualizacije igre. Slika 4.10. prikazuje izgled računalne igre tijekom igranja na kojoj se može vidjeti promjena boje.



Sl. 4.10. Mijenjanje pozadine

Na slici su prikazana dva dijela tijekom igranja istog pokušaja. Gornja slika sadrži tamno plavu boju, a donja tamno ljubičastu. Izmjena boja vrši se postupno uz pomoć koda koji je prikazan na slici 4.11.

```
private Color[] colors = new Color[] { new Color(0.29f, 0.00f, 0.51f), new Color(0.33f, 0.42f, 0.18f), new Color(0.00f, 0.00f, 0.55f) };
private int currentColorIndex = 0;
private int nextColorIndex = 1;
private float t = 0.0f;

void Start()
{
    if (cam == null)
    {
        cam = Camera.main;
    }
    cam.backgroundColor = colors[currentColorIndex];
}

void Update()
{
    cam.backgroundColor = Color.Lerp(colors[currentColorIndex], colors[nextColorIndex], t);
    t += Time.deltaTime * colorChangeSpeed;

    if (t >= 1.0f)
    {
        t = 0.0f;
        currentColorIndex = nextColorIndex;
        nextColorIndex = (nextColorIndex + 1) % colors.Length;
    }
}
```

Sl. 4.11. Kod za mijenjanje pozadine

Pomoću ove metode mijenja se boja pozadine kamere postupno između tri unaprijed definirane boje. Na početku se postavlja inicijalna boja, a zatim se u svakoj iteraciji igre pozadina kamere glatko interpolira između trenutne i sljedeće boje. Brzina promjene boje kontrolira se pomoću varijable koja se povećava s proteklim vremenom. Kada se završi prijelaz između dvije boje, kod ažurira indekse boja i počinje prelazak na sljedeću boju, stvarajući beskonačni ciklus promjena boja.

4.3. Rezultat i spremanje rezultata

Rezultat ima ključnu ulogu unutar ove igre. Motivira igrače i prati napredak tijekom igranja. Rezultat se povećava kako igrač prelazi prepreke i prikupljanjem novčića za bonus. Igra se sastoji od praćenja trenutnog rezultata tijekom istog pokušaja igranja i najvećeg rezultata koji je ikada postignut. Na slici 4.12. prikazuju se metode zadužene za praćenje trenutnog rezultata.

```
private void Update (){
    if(isPlaying){
        currentScore += Time.deltaTime;
    }

}

public void StartGame(){

    onPlay.Invoke();
    isPlaying = true;
    currentScore = 0;
}
```

Sl. 4.12. Trenutni rezultat

Prvom metodom se provjerava je li igrač pokrenuo igru, ukoliko je taj uvjet ispunjen trenutnom rezultatu se dodaje vrijeme provedeno igrajući. Druga metoda pokreće igru i postavlja trenutni rezultat na 0. Spremanje rezultata i postavljanje najvećeg postignutog rezultata se prikazuje na slici 4.13.

```
public static void Save(string fileName, string dataToSave){
    if(!Directory.Exists(SAVE_FOLDER)){
        Directory.CreateDirectory(SAVE_FOLDER);
    }

    File.WriteAllText(SAVE_FOLDER + fileName + FILE_EXT, dataToSave);
}

public static string Load(string fileName){
    if(File.Exists(SAVE_FOLDER+ fileName + FILE_EXT)){
        string loadedData = File.ReadAllText (SAVE_FOLDER+ fileName + FILE_EXT);

        return loadedData;
    } else{
        return null;
    }
}
```

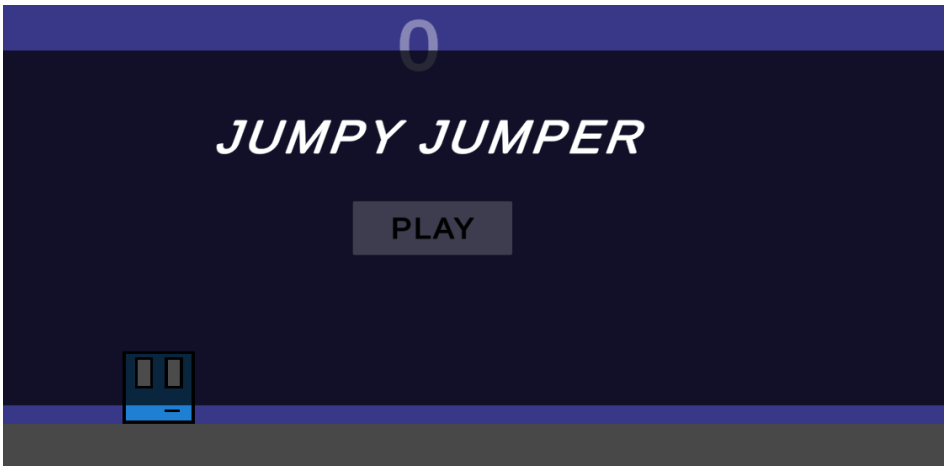
```
public void GameOver (){

    if(data.highscore < currentScore){
        data.highscore = currentScore;
        string saveString = JsonUtility.ToJson(data);
        SaveSystem.Save ("save", saveString);
    }

    isPlaying = false;
```

Sl. 4.13. Spremanje rezultata

Prvo se provjerava postoji li direktorij u kojem će se podaci spremiti. Ukoliko direktorij ne postoji, metoda ga kreira. Ime datoteke sastoji se od mape za spremanje, imena datoteke i proširenja datoteke. Metoda prima dva argumenta: ime datoteke i podatke koje treba spremiti (u obliku stringa). Druga statička metoda učitava spremljene podatke iz datoteke. Prvo provjerava postoji li tražena datoteka na određenoj lokaciji. Ako datoteka postoji, učitava njen sadržaj, a ako datoteka ne postoji, metoda vraća null kako bi označila da podaci nisu pronađeni. Metoda *GameOver()* upravlja završetkom igre i provjerava je li postignut novi najbolji rezultat. Ako je trenutni rezultat igrača veći od dosadašnjeg najboljeg rezultata, trenutni rezultat postavlja se kao novi najveći rezultat. Nakon ažuriranja najboljeg rezultata, podaci se pretvaraju u JSON format i spremaju u datoteku. Osigurava se da se novi najbolji rezultat ažurira i sprema svaki put kada igrač završi igru, ako je postigao veći rezultat nego prije.

4.4. Korisničko sučelje

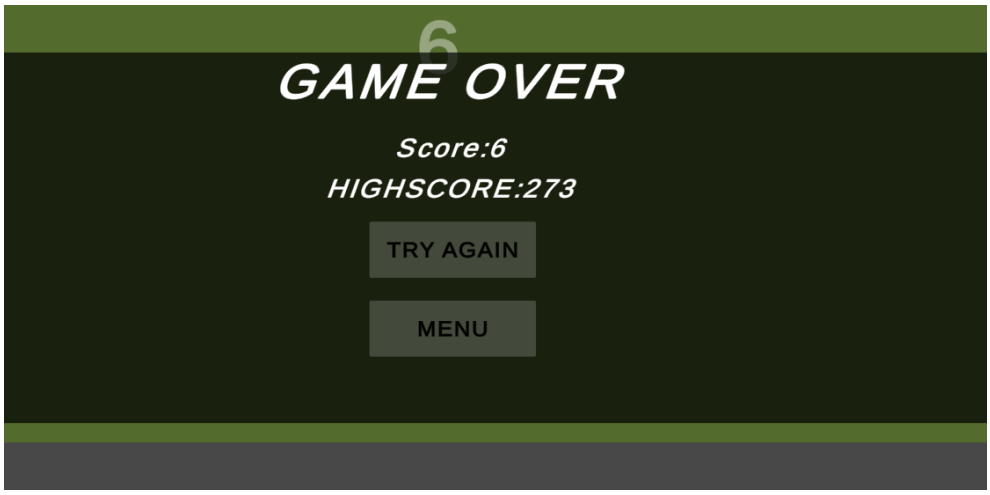
Implementirano je jednostavno, ali funkcionalno korisničko sučelje koje omogućuje igračima intuitivnu navigaciju kroz različite trenutke igre. Sučelje je dizajnirano da bude jasno i lako razumljivo, čime omogućava nesmetanu interakciju s igrom. Na početku igre, igrač je suočen s glavnim izbornikom (slika 4.14.) koji sadrži naziv igre i gumb za pokretanje igre.



Sl. 4.14. Glavni izbornik

Jednostavnim pritisak gumba započinje igru i igrač može krenuti u akciju. Ovaj početni ekran također može sadržavati gumbe za dodatne opcije poput postavki igre, podešavanje glasnoće i slično.

Tijekom igre, na zaslonu se prikazuje minimalni broj elemenata korisničkog sučelja kako bi igrač bio fokusiran na samu igru. Ključni element je brojač rezultata koji se dinamički ažurira kako igra napreduje, prikazujući trenutačni rezultat igrača. Nakon što igrač izgubi, odnosno sudari se sa preprekom, pojavljuje se izbornik koji označava kraj igre (slika 4.15.) s opcijama za ponovno pokretanje koja omogućava igraču da odmah ponovno pokrene igru i pokuša postići bolji rezultat.



Sl. 4.15. Izbornik za kraj igre

Ovaj zaslon sadrži dva gumba koji omogućuju igraču da pokuša ponovno igrati ili da se vrati na glavni izbornik. Također se na ovom zaslonu prikazuje trenutačni rezultat koji je postignut tijekom tog pokušaja i najveći rezultat postignut tijekom prijašnjih pokušaja. Ukoliko igrač obori svoj rekord, najveći rezultat i trenutni rezultat prikazuju iste vrijednosti.

Kod prikazan na slici 4.16. omogućuje upravljanje korisničkim sučeljem. Omogućava prikaz i ažuriranje različitih elemenata korisničkog sučelja, kao što su trenutni rezultat, glavni izbornik i izbornik za kraj igre.

```
public class UIManager : MonoBehaviour
{
    [SerializeField] private TextMeshProUGUI scoreUI;
    [SerializeField] private GameObject startMenuUI;
    [SerializeField] private GameObject gameOverUI;

    [SerializeField] private  TextMeshProUGUI gameOverScoreUI;
    [SerializeField] private  TextMeshProUGUI gameOverHighscoreUI;

    GameManager gm;

    private void Start(){
        gm = GameManager.Instance;
        gm.onGameOver.AddListener(ActivateGameOverUI);
    }
    public void PlayButtonHandler (){
        gm.StartGame();
    }
    public void ActivateGameOverUI(){
        gameOverUI.SetActive(true);
        gameOverScoreUI.text= "Score:" + gm.ShowScore();
        gameOverHighscoreUI.text = "Highscore:"+ gm.ShowHighscore();
    }
    private void OnGUI(){
        scoreUI.text = gm.ShowScore();
    }
```

Sl. 4.16. Kod za upravljanje korisničkog sučelja

Prikazuju se sučelja za mjerenje rezultata, glavnog izbornika i izbornika koji označuje kraj igre. Prilikom pokretanja igre kreira se objekt igrača. Druga metoda omogućuje pokretanje igre pomoću pritiska na gumb. Na kraju igre aktivira se izbornik koji prikazuje trenutni postignuti rezultat i najveći rezultat. Metoda *OnGUI()* ažurira prikaz trenutnog rezultata u realnom vremenu, prikazujući ga dok igra traje. Tako igrači mogu vidjeti svoj rezultat tijekom igranja.

5. ZAKLJUČAK

Cilj rada bio je napraviti funkcionalnu i zabavnu 2D računalnu igru u kojoj igrač izbjegava nadolazeće prepreke. Za izradu računalne igre korišten je C# programski jezik i Unity Engine zbog brzine i poticajnog razvojnog okruženja. Glavni cilj igre je ostati što duže živ i postići veći rezultat te oboriti prethodne rekorde. Igra je zabavna i sadrži sve potrebne algoritme koji omogućavaju povećanje težine prolaskom vremena. Igrač ima mogućnosti dugog i kratkog skoka, prekida skoka u zraku te saginjanja. Nedostatak računalne igre je nemogućnost podešavanja zvuka pozadinske glazbe i zvučnih efekata. Nedostatak igre moguće je riješiti budućim proširenjima. Unatoč nedostatku igra je funkcionalna i izazovna.

LITERATURA

[1] C. Assets, 8-Bit Coin Sound Effect, 2024. [Mrežno]. Dostupno: https://creatorassets.com/a/8-bit-coin-sound-effects. [Pokušaj pristupa Rujan 2024.].

[2] D. Fesliyan, Royalty Free 8-Bit Background Music, Fesliyan, 2024. [Mrežno]. Dostupno: https://www.fesliyanstudios.com/royalty-free-music/downloads-c/8-bit-music/6. [Pokušaj pristupa Rujan 2024.].

[3] S. Moeys, Poki, Poki B.V, 2014.. [Mrežno]. Dostupno: https://poki.com/. [Pokušaj pristupa Rujan 2024.].

[4] S. Moeys, Dinosaur Game, Poki B.V, 2014. [Mrežno]. Dostupno: https://poki.com/en/g/dinosaur-game. [Pokušaj pristupa Rujan 2024.].

[5] S. Moeys, Jetpack Joyride, Poki B.V, 2014. [Mrežno]. Dostupno: https://poki.com/en/g/jetpack-joyride. [Pokušaj pristupa Rujan 2024.].

[6] S. Moeys, Temple Run 2, Poki B.V, 2014. [Mrežno]. Dostupno: https://poki.com/en/g/temple-run-2. [Pokušaj pristupa Rujan 2024.].

[7] M. McDonnella, Flappy Bird, 2014. [Mrežno]. Dostupno: https://flappybird.io/. [Pokušaj pristupa Rujan 2024.].

[8] N. F. J. A. David Helgason, Unity, Unity Technologies, 2004. [Mrežno]. Dostupno: https://unity.com/. [Pokušaj pristupa Rujan 2024.].

[9] N. F. i. J. A. David Helgason, Unity Asset Store, Unity Technologies, 2010. [Mrežno]. Dostupno: https://assetstore.unity.com/. [Pokušaj pristupa Rujan 2024.].

[10] Microsoft, Visual Studio Marketplace, Microsoft, 2015. [Mrežno]. Dostupno: https://marketplace.visualstudio.com/. [Pokušaj pristupa Rujan 2024.].

SAŽETAK

Unutar ovog rada prikazana je izrada 2D računalne igre u Unity Engine-u. Cilj igre je ostati što duže živ i postići što veći rezultat te oboriti prethodne rekorde koji su bili postavljeni. Igra je napravljena u Unity okruženju, a kodovi su pisani pomoću C# programskog jezika. Svi likovi unutar igre su napravljeni u Piskel web aplikaciji, a zvuk uređen u AudioMass web aplikaciji. U igri igrač kontrolira plavi kvadrat koji mora izbjeći nadolazeće prepreke i ukoliko želi skupljati dodatne bodove. Prepreke se vremenom češće pojavljuju i postaju brže što ih čini izazovnijima za izbjegavanje. Mehanike igre su pokretane skriptama. Igra se sastoji od glavnog izbornika koji nudi opciju za pokretanje igre, prikaza trenutnog rezultata unutar igre i izbornika za kraj igre koji nudi opcije za ponovno pokretanje ili povratak na glavni izbornik. Prikazuje trenutni i najveći rezultat.

Ključne riječi: 2D računalna igra, C#, Piskel, Unity

ABSTRACT

2D computer game

This paper presents the development of a 2D computer game in Unity Engine. The objective of the game is to stay alive as long as possible and achieve the highest score, while breaking previous records that have been set. The game is developed in the Unity environment, and the code is written in the C# programming language. All characters in the game are created using the Piskel web application, and the sound is edited in the AudioMass web application. In the game, the player controls a blue square that must avoid oncoming obstacles and, if desired, collect additional points. The obstacles appear more frequently and become faster over time, making them harder to avoid. The game mechanics are driven by scripts. The game consists of a main menu offering the option to start the game, an in-game score display, and a game over menu offering options to restart the game or return to the main menu. It displays both the current and highest scores.

Keywords: 2D computer game, C#, Piskel, Unity