Izrada prototipa STEM igre za Android platformu

Buterin, Nika

VELEUČILIŠTE U BJELOVARU
PREDDIPLOMSKI STRUČNI STUDIJ RAČUNARSTVO

IZRADA PROTOTIPA STEM IGRE ZA *ANDROID* PLATFORMU

Završni rad br. 01/RAČ/2020

Nika Buterin

Bjelovar, mjesec 2020.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Kandidat: **Buterin Nika** Datum: 09.06.2020. Matični broj: 001846

JMBAG: 0314017912

Kolegij: **RAZVOJ RAČUNALNIH IGARA**

Naslov rada (tema): **Izrada prototipa STEM igre za Android platformu**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Tomislav Adamović, mag. ing. el.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Ivan Sekovanić, mag.ing.inf.et comm.techn., predsjednik**
2. **Tomislav Adamović, mag.ing.el., mentor**
3. **Krunoslav Husak, dipl.ing.rač., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 01/RAČ/2020

U radu je potrebno implementirati prototipnu STEM edukativnu igru za Android platformu. Navedenu igru potrebno je izgraditi koristeći se programskim okvirom Unity. Igra je namijenjena za mlađe dobne skupine od 6-10 godina. Igra treba sadržavati početno korisničko sučelje s postavkama, uputama i najboljim rezultatima. Postavke upravljaju zvukom i drugim specifičnostima igre na dinamičan način. Igra mora imati priču i tijek u vidu razina. Igra i korisničko sučelje trebaju biti prilagođeni za Android mobilnu platformu. Za potrebe ovoga rada dio vizuala ili animacija potrebno je izraditi samostalno.

Zadatak uručen: 09.06.2020. Mentor: **Tomislav Adamović, mag. ing. el.**

REPUBLIKA HRVATSKA BJELOVAR VELEUČILIŠTE U BJELOVARU

Zahvala

Zahvaljujem mentoru Tomislavu Adamoviću mag.ing.el.odobravanje predložene teme završnog rada.

Voljela bih izraziti zahvalnost prema prof. Anti Javoru na danim savjetima i smjernicamaprilikom izradezavršnog rada. Zahvalna sam na slobodi izbora prilikom izrade praktičnog djela rada, kao i podržavanju ideja, konzultacijama te ostalim oblicima pomoći koji su bili korisni pri samoj izradi.

Također, želim izraziti zahvalnost premaVeleučilištu u Bjelovaru na stečenom znanju i vještinama prilikom pohađanja stručnog studija računarstva.

Sadržaj

1. Uvod

U današnje vrijeme, popraćeno naprednim razvojem tehnologije i uređaja, igranje je postalo sastavni dio ljudskog života. Na raspolaganju je velika raznolikost igara, bilo računalnih, mobilnih ili onih namijenjenih igraćim konzolama. Razvoj igara spoj jeviše aspekata kao što su razvoj softvera, grafički dizajn, kompozicija glazbe izvučnih efekata te pričanje priče. Za razvoj igre,osim same idejepotrebno je osmisliti dizajn i smisao igre te izraditi razvojni plan.Izrada igre uključuje više disciplina i područja znanja, aproces izrade vrlo je kompleksan i predstavlja velik izazov.

Industrija igara na globalnoj razini bilježi enorman uspjeh.Globalno tržište nudi razne žanrove video igara, namijenjene različitim platformama. Na tržištu su prisutnerazne edukativne igrenamijenjene djeci, no većina takvih igara izrađena je na engleskom jeziku i ne sadrži jezičnu podršku. Cilj ovog rada je izrada zabavne i edukativne *STEM* igre, namijenjene dobnom razdoblju od 6 do 10 godina.Svrha izrade igre jest stvaranje interesa prema određenom*STEM* području.Igranjem igre pojedinac bi trebao steći osnovno znanje kroz samostalno rješavanje zadataka i problematike na zabavan način. Igra ne zahtjeva sposobnost čitanja već samostalno zaključivanje koje se temelji na vizualno izrađenim i ponuđenim odgovorima te drugim elementima igre.Pozitivan i očekivan rezultat igranja igre je spoznaja interesa i usmjeravanje prema *STEM* područjima u ranoj životnoj dobi.

U završnom radu objasnit će se proces realizacije edukativne 2D igre zamobilnu *Android*platformu. Objasnit će se samostalno kreiranje resursa igre (engl. Game Assets), izrada projektnog zadatkau *Unity* okruženju, pisanje programskog koda i prijenos igre na *Google Play trgovinu*.

Kroz drugo poglavlje rada opisan je *Android* operacijski sustav, verzije sustava, osnovni pojmovi i konfiguracija operacijskog sustava (OS).Objašnjena je implementacija mobilne igre za *Android* platformu. Treće poglavlje opisuje korištenu programsku podršku koja uključuje rad s programima kao što su*Unity, VisualStudio, AdobeIllustrator*i *Adobe Photoshop*.Objašnjen je rad s pojedinim programskim alatom te implementacija istoga pri izradi praktičnog dijela. Četvrto poglavlje opisuje razvojni proces koji započinje idejom, a završavarealizacijomprototipa igre. Proces realizacije projektnog zadatkazahtjeva izradu razvojnog plana, pripremu grafičkih komponenata, izradu projekta i pisanje programskog

koda. U završnom dijelu rada objašnjen je prijenos igre na *Google Play* trgovinu. Opisano je interno i otvoreno testiranje igre te izrada produkcijskog izdanja.

2. OPERACIJSKI SUSTAV *ANDROID*

Android je mobilni operacijski sustavkojeg je razvio*Google* u vlasništvu saveza *Open Headset*. Operacijski sustav temelji se na *Linux* jezgrii osmišljen je kako bi zadovoljio potrebe uređaja s osjetljivim zaslonom, poput mobitela i tableta. *Android* je besplatan sustavotvorenog koda, dostupan korisnicima uz poštivanje licence.*Android* je licenciran pod licencom namijenjenom poslovnim subjektima (*Apache/MIT*)[1].

2.1 Razvoj operacijskog sustava *Android*

Google gotovo svake godine ili u kraćem vremenskom razdoblju objavljuje nadogradnju *Android* sustava. Nadogradnja se sastoji od novih i poboljšanih značajki sustava. Svaka verzija sustava sadrži svoj *SDK*,skraćeno od *Software Development Kit*, a koristi se za izradu aplikacija ili igara kompatibilnih s aktualnom verzijom sustava. Operacijski sustav pogodan je za uređaje različitih veličina ekrana i gustoće zaslona. Ova značajka sustava nudi veliku prednost, ali unatoč tome što sustav provodi skaliranje i mijenjanje na različitim veličinama zaslona. Realizacija igre koja je namijenjena raznim platformama (engl. Cross-Platform)zahtjevaulaganje velikihnaporaprilikom optimizacije. Tablica 2.1 prikazuje sve nadogradnje *Android* sustava[2].

Android verzija 8.0 i novijeverzije sadrže identifikacijski ID format***PVBB.YYMMDD.bbb[.Cn]*** gdje:

- ***P*** označava početno slovo izdanja. Primjer 2.1: N označava Nougat
- ***V*** označava podržani *vertical*
- ***BB*** je alfanumerički kod koji omogućuje Google-u identifikaciju koda koji je korišten za izradu nadogradnje
- ***YYMMDD*** identificira datum kada je izdanje
- ***bbb*** identificira individualne verzije povezane istim datumskim kodom koji započinje s 001
- ***Cn*** označava opcionalnu alfanumeričku oznaku koja identificira popravak ili ažuriranje (engl. *Hotfix),* a započinje s A1

Tablica 2.1: Verzije *Android* sustava [2]

Izdanje	**Broj verzije**	***API* razina / NDK izdanje**
*Android*11	11	*API* razina 30
*Android*10	10	*API* razina 29
Pie	9	*API* razina 28
Oreo	8.1.0	*API* razina 27
Oreo	8.0.0	*API* razina 26
Nougat	7.1	*API* razina 25
Nougat	7.0	*API* razina 24
Marshmallow	6.0	*API* razina 23
Lollipop	5.1	*API* razina 22
Lollipop	5.0	*API* razina 21
KitKat	4.4 - 4.4.4	*API* razina 19
Jelly Bean	4.3.x	*API* razina 18
Jelly Bean	4.2.x	*API* razina 17
Jelly Bean	4.1.x	*API* razina 16
Ice Cream Sandwich	4.0.3 - 4.0.4	*API* razina 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	*API* razina 14, NDK 7
Honeycomb	3.2.x	*API* razina 13
Honeycomb	3.1	*API* razina 12, NDK 6
Honeycomb	3.0	*API* razina 11
Gingerbread	2.3.3 - 2.3.7	*API* razina 10
Froyo	2.2.x	*API*razina 8, NDK 4
Eclair	2.1	*API*razina 7, NDK 3
Eclair	2.0.1	*API*razina 6
Eclair	2.0	*API*razina 5
Donut	1.6	*API*razina 4, NDK 2
Cupcake	1.5	*API*razina 3, NDK 1
(no codename)	1.1	*API*razina 2
(no codename)	1.0	*API*razina 1
Froyo	2.2.x	*API*razina 8, NDK 4

2.2 Arhitektura operacijskog sustava

Arhitektura operacijskog sustava *Android* sastoji se od 6 slojeva koji su prikazani u obliku hijerarhijskog modela.Svaki sloj sadrži softverske komponente te ima vlastite karakteristike i ulogu.Zadaća nižih slojeva jest pružanje usluga slojevima viših razina [3, 4]. Slika 2.1 prikazuje arhitekturu operacijskog sustava.

2.2.1 *Sistemske aplikacije*

Sistemske aplikacije (engl. *System Apps*) nalaze se na vrhu hijerarhijskog modela. *Android* platforma sastoji se od skupa osnovnih aplikacija u koje ubrajamo e-poštu, SMS, kalendar, web preglednik, kontakte, kameru itd. Osnovne aplikacije mogu biti zamijenjene korisničkim odabiromaplikacija trećih strana. Primjer 2.2: Aplikacija trećih strana može zamijeniti zadani web preglednik. Sistemske aplikacije funkcioniraju kao i osnovne aplikacije, ali pružaju ključne mogućnosti poput unaprijed izrađenih funkcionalnosti koje su lako dostupne razvojnim programerima. Primjer 2.3: Ukoliko aplikacija želi isporučiti SMS poruku, nije potrebno izraditi zasebnu funkcionalnost već pozvati zadanu SMS aplikaciju koja će isporučiti poruku[3, 4].

2.2.2 *Aplikacijski okvir*

Aplikacijski okvir (engl. *Java API Framework*) nalazi se na drugom sloju hijerarhijskog modela.Operacijski sustav *Android* pruža skup značajki koje su dostupne putem aplikacijskog programskog sučelja (*API*)[3, 4].Aplikacijska programska sučelja tvore građevne blokove koji su potrebni za izradu *Android* aplikacija, pojednostavljujući ponovnu upotrebu modularnih komponenata i usluga koje uključuju:

- **Sustav prikaza** (engl. *View System*) koji služi za izradu korisničkog sučelja aplikacije. Sustav prikaza uključuje liste, tekstne okvire, gumbe i ugrađeni preglednik,
- **Upravitelj resursa** (engl. *Resource Manager*) koji pruža pristup resursima koji nisu izrađeni putem programskog koda, kao što je grafika,
- **Upravitelj obavijesti** (engl. *NotificationManager*) koji omogućava prikaz prilagođenih obavijesti na statusnoj traci aplikacije,
- **Upravitelj aktivnosti** (engl. *Activity Manager*) koji upravlja životnim ciklusom aplikacija,

- **Pružatelji sadržaja** (engl. *Content Providers*) koji omogućuju pristup podacima drugih aplikacija, poput kontakata ili dijeljenja vlastitih podataka.

2.2.3 *Programske biblioteke*

Programske biblioteke (engl. *Native C/C++ Libraries*) nalaze se na trećem sloju hijerarhijskog modela uz *Android* radno okruženje. Osnovne komponente i usluge *Android* sustava, poput *Android* okruženja i sloja hardverske apstrakcije izrađene su od programskog koda koji zahtijeva programske biblioteke[3, 4].Programske biblioteke pisane su u programskim jezicima C i C++ i u njih ubrajamo:

- ***Webkit*** je mehanizam kojeg koriste preglednici kao što su *Safari, Google Chrome, Mail, App Store* za prikaz Web stranica. *Webkit* je dostupan za *macOS, iOS* i *Linux* platformu[5],
- ***SQLite*** je biblioteka programskog jezika *C* koja implementira *SQL* bazu podataka. *SQLite* jedna je od najkorištenijih baza podataka, a izvorno je ugrađena u većinu mobilnih telefona i računala[6],
- ***Apache Harmony*** je implementacija Java izvornog koda. *Apache Harmony* razvio je *Apache Software Foundation* 2005. godine [7],
- ***OpenGL***je aplikacijsko programsko sučelje koje pruža skup funkcija, a one omogućuju manipulaciju grafikom i slikama. *OpenGL* predstavlja skup biblioteka za 3D grafiku [8],
- ***OpenSSL*** je potpuno opremljen paket alata za *Transport Layer Security* (TLS) i *Secure Sockets Layer* (SSL) protokol. *OpenSSL* služi i kao biblioteka kriptografije opće namjene [9].

2.2.1 *Sloj hardverske apstrakcije*

Sloj hardverske apstrakcije (engl. *Hardware Abstraction Layer*) nalazi se na četvrtom sloju hijerarhijskog modela. Sloj hardverske apstrakcije, skraćeno HAL definira standardno sučelje za interakciju s ugrađenim hardverskim komponentama. HAL se sastoji od niza biblioteka, odnosno modula, od kojih svaki implementira sučelje za određenu hardversku komponentu[3, 4].

2.2.2 *Android radno okruženje*

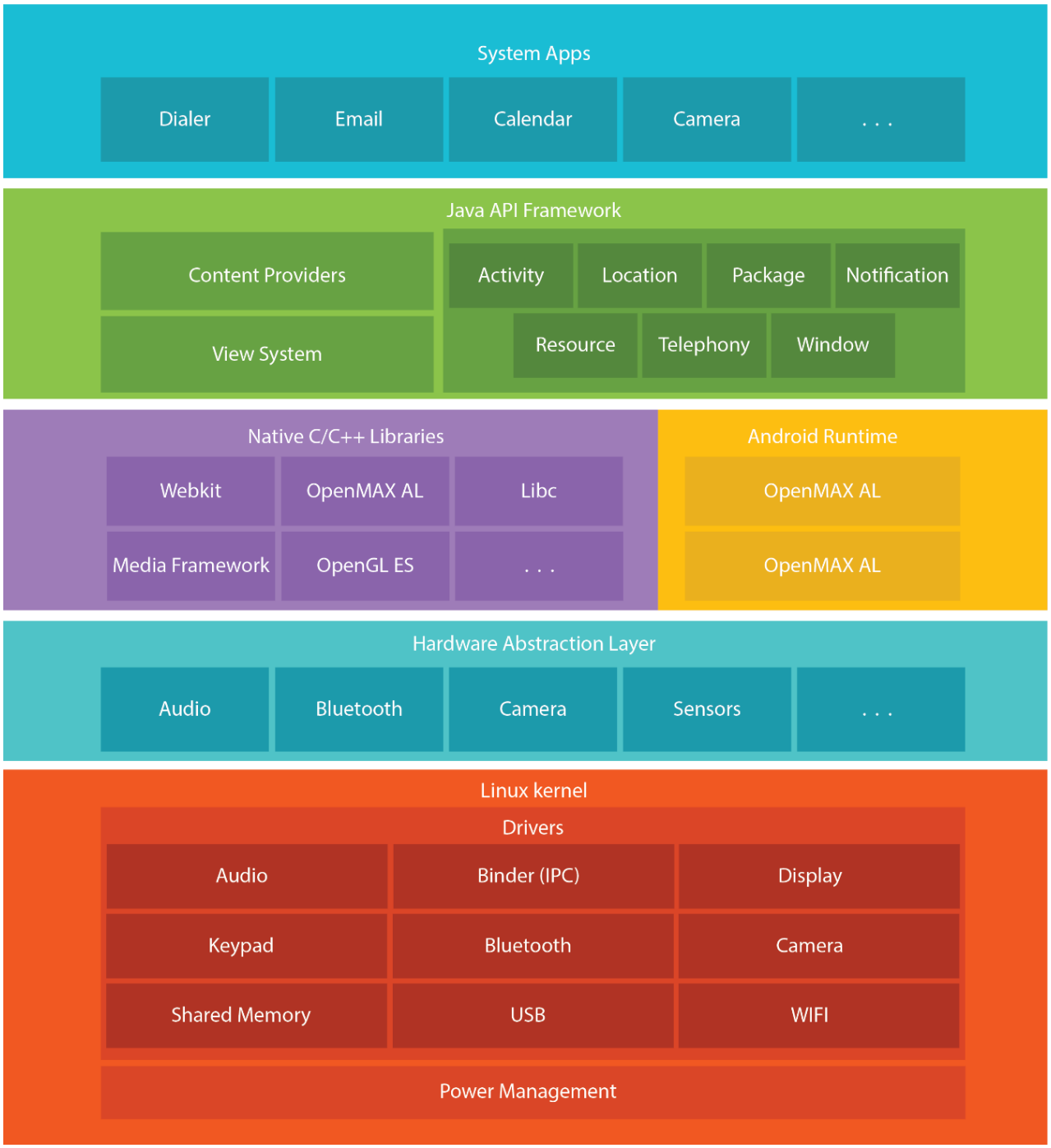
Android radno okruženje (engl. *Android Runtime*) nalazi se na trećem sloju hijerarhijskog modela uz programske biblioteke. *Android*radnookruženje, skraćeno ART omogućuje pokretanje više virtualnih uređaja na uređajima koji ne sadrže mnogo memorije izvršavanjem *Dalvik Executable* datoteka (engl. *DEX*). *DEX* format dizajniran je i osmišljen za rad s *Android* sustavom te je optimiziran za minimalno korištenje memorije Prilikom instalacije, *ART* kompajlira aplikacije pomoću programskog alata dex2oat. Program prihvaća ulazne DEX datoteke i generira izvršnu aplikaciju za ciljani uređaj. Svaka se aplikacija pokreće kao zaseban proces na virtualnom stroju Dalvik.

Dalvikje virtualni stroj (VM) koji je posebno dizajniran i izrađen za *Android* platformu. Dizajnirao ga je Dan Bornstein uz pomoć inženjera tvrtke *Google*, kao dio *Android* platforme. VM je optimiziran za potrebe slabije memorije te je osmišljen kako bi istovremeno dopustio pokretanje višestrukih instanci *VM*. *Dalvik* VM koristi vlastiti bajt kod i pokreće Dalvik Executable format datoteke (*DEX*).

***DEX* format** konfigurirana je datoteka *Dalvik* VM i nalazi se u .apk datoteci na samom uređaju. Datotekasadrži programski kod koji se izvršava u *Android Runtime* okruženju. Datoteke mogu biti automatski kreirane prevođenjem konfiguriranih aplikacija, n*API*sanih u programskom jeziku Java. Format datoteke koristi međusobne, specifične konstante kao primaran mehanizam za očuvanje memorije. Konstante pohranjuju korištene vrijednosti poput polja, varijabli, klasa, sučelja i imena metoda[10].

2.2.3 *Linux jezgra*

Linux jezgra (engl. *Linux Kernel*) nalazi se na dnu hijerarhijskog modela. *Linux* jezgra pruža osnovne funkcionalnosti sustava kao što su upravljanje procesima, upravljanje memorijom i upravljanje uređajima. Jezgra sadrži pogonske programe od kojih su najbitniji program za međuprocesnu komunikaciju i program za upravljanje napajanjem. Program za međuprocesnu komunikaciju omogućuje razmjenu podataka između različitih ili unutar istog procesa[3, 4].



Slika 2.1: Arhitektura *Android* sustava

2.3 Razvoj igara za *Android* platformu uz Unity

Android je široko rasprostranjeni operacijski sustav kojeg koriste razni uređaji.Razvoj *Android* igara počeo se razvijati u trenutku kada je industrija mobilnih igara migrirala s *Symbian*, Java ili drugih operacijskih sustava na *Android* ili druge sustave.Razvoj igara za *Android* platformu može biti izrađen putem raznih programskih okvira ili pokretača igre od kojih su najpopularniji *Unity* i *UnrealEngine*. *Unity*je *Cross-Platform Engine*koji omogućuje izradu igara za različiteplatforme.

Izrada igara za *Android* platformu koristeći Unity pokretač igre zahtjevainstalaciju *Android Build Support* modula. Prilikom instalacije modula, bitno je izvršiti i instalaciju programskih alata koji uključuju *Android Software Development* (SDK), *Android Native Development kit* (NDK) i *OpenJDK*.*Android SDK* neophodan je za razvoj igre i pruža platformu za razvoj igre. Programski alati omogućuju izgradnju i izvršavanje programskog koda na *Android* uređajima[11, 12].

3. PROGRAMSKA PODRŠKA

3.1 *Unity*

Unity je razvila tvrtka *Unity Technologies SF*, osnovna 2004. godine. *Unity* je pokretač igrei potpuno integrirano razvojno okruženje (IDE) za izradu 2D i 3D igara.*Unity* omogućuje razvoj igarazaplatforme kao što su*Windows, macOS, Linux Standalone, tvOS, iOS, Lumin, Android, WebGL, PS4 i Xbox One*i druge platforme[13, 14].

Unity Game Engine sastoji se od licenci koje se razlikuju prema upotrebi i namjeni. Tipovi licenci su:*Unity Personal, Unity Plus, Unity Pro i Unity Enterprise*. Licence se razlikuju prema mjesečnoj članarini, značajkama i korisničkoj podršci. Prema ugovoru o licenci programske podrške (*EUL),* korisnici koji koriste besplatnu verziju ne smiju imati godišnje prihodeveće od 100.000 USD. Ukoliko korisnik prijeđe određeni limit, potrebno je nadograditi licencu za *Unity Plus* ili *Unity Pro* koji omogućuju godišnje prihode do 200.000 USD. Prilikom zarade veće od 200.000 USD potrebno je nadograditi licencu na *Unity Enterprise*[13, 14].

3.2 Unity Hub

Unity Hub je samostalna aplikacija koja pojednostavljuje način upravljanja projektima, preuzimanjima i instalacijama [14].*Unity Hub* može biti korišten za:

- Upravljanje licencama
- Stvaranje projekata
- Pokretanje različitih *Unity* verzija
- Pokretanje dvije *Unity* verzije istovremeno
- Instalaciju i nadogradnju komponenata potrebnih za realizaciju projekta
- Korištenje predložaka pri izradi projekata

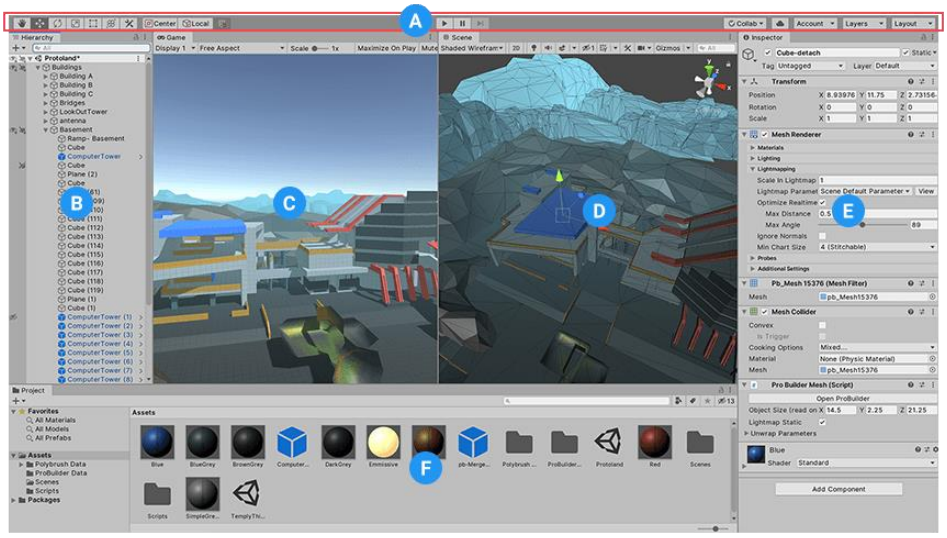
3.3 Unity Remote

Unity Remote zasebna je aplikacija koja može biti preuzeta, a namijenjena je *Android, iOS* i *tvOS* platformama. Aplikacija se povezuje s razvojnim okruženjem i emitira sadržaj projekta na zaslonu povezanog uređaja. *Unity Remote* omogućuje testiranje projekata u stvarnom vremenu[13, 14].

3.4 Razvojno okruženje

3.4.1 *Korisničko sučelje*

Korisničko sučelje dio je pokretača igre, a služi za izradu novog projekta, kreiranje scena, pohranu resursa igre i izradu animacija. Korisničko sučelje surađuje s programima za razvoj programskog koda kao što su *VisualStudio* ili *MonoDevelop*[14].



Slika 3.1: *Unity* Editor [14]

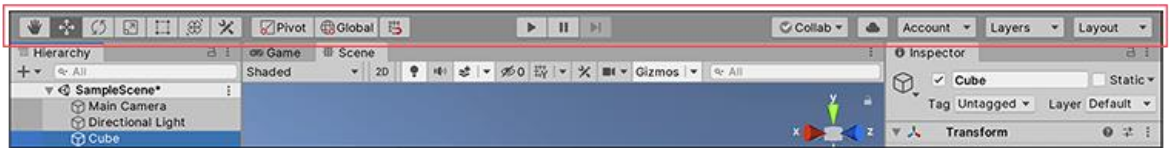
Slika 3.1 prikazuje korisničko suč,elje. Korisničko sučelje sastoji se od alatne trake (A), hijerarhijskog prozora (B), prikaza igre (C), prikaza scene (D), inspektora (E) i projekta (F).

3.4.2 *Alatna traka*

Alatna trakapruža pristup najbitnijim značajkama projekta. Alatna traka prikazana je na slici 3.2.Lijeva strana alatne trake sastoji se od upravljačkih alata. Središnji dio alatne trakesadrži kontrolepokreni, zaustavi i koraci. Desna strana alatne trake sadrži kontrole koje omogućuju pristup *Unity Colloborate-u*, *Unity* oblak servisu i *Unity* korisničkom računu. Slijedi izbornik vidljivosti sloja i *Editor Layout* izbornik [14].Tablica 3.1 opisuje osnovne kontrole alatne trake i njihovu funkcionalnost.

Tablica 3.1: Opis alatne trake [14]

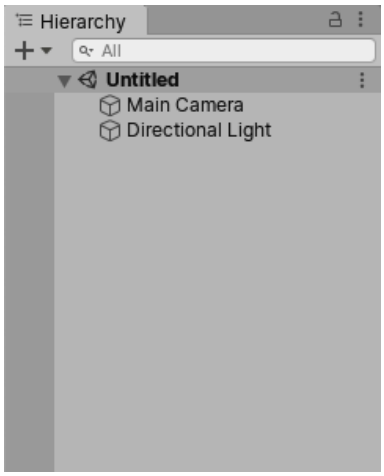
Naziv kontrole	Opis
Zaslon	Odabir kamere.
Omjer	Ova opcija služi kako bi se igra testirala ili prilagodila uređajima različitih veličina.
Skaliranje	Ova opcija omogućuje skaliranje kako bi se dobio prikaz cijelog zaslona na kojem je rezolucija uređaja veća od veličine prozora igre.
Uvećaj	Omogućuje uvećavanje prikaza igre, odnosno stopostotno korištenje veličine uređivača prilikom reprodukcije.
Utišaj zvuk	Isključivanje zvuka prilikom reprodukcije.
Statistika	Prikaz statistike prikazivanja zvuka i grafike. Služi za nadziranje performansi igre prilikom reprodukcije.
Gizmo	Omogućuje prikaz *Gizmosa* tijekom reprodukcije.



Slika 3.2: Alatna traka [14]

3.4.3 ***Hijerarhijski prozor***

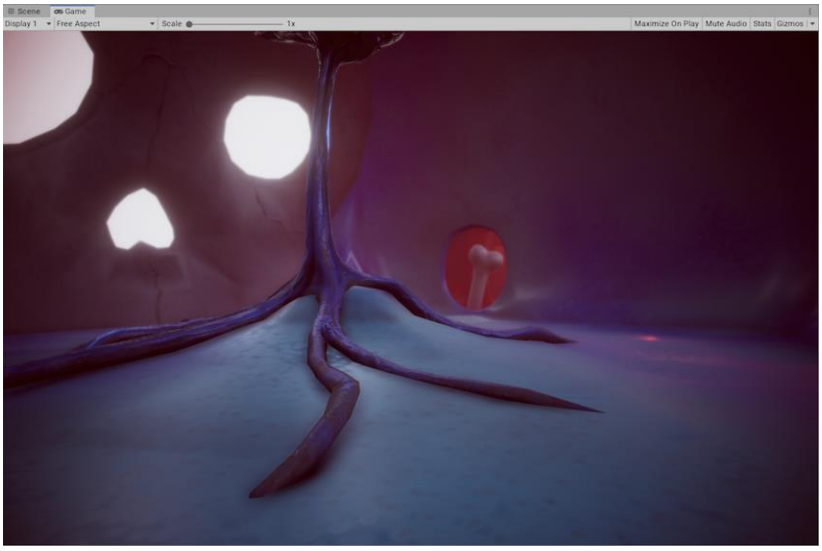
Hijerarhijski prozor prikazuje prisutnost svih objekata koji se nalaze u sceni ili projektu. Hijerarhija predstavlja strukturu izrađenih objekata prema redoslijedu kreiranja[14].Slika 3.3 prikazuje hijerarhijski prozor novoizrađenogprojekta. Prilikomizrade novog projekta, *Unity* prema standardnim postavkama dodjeljuje scenu *Untitled,* kameru i osvjetljenje.



Slika 3.3: Hijerarhijski prozor [14]

3.4.4 *Prikaz igre*

Prikaz igre simulira prikaz igre kroz scensku kameru. Simulacija započinje prilikom pritiska na gumb pokreni [14]. Slika 3.4 prikazuje prikaz igre u slobodnom aspektu prema zadanim postavkama. Igru je moguće prikazati u već definiranim veličinama ekrana, no moguće je i izraditi nove veličine.



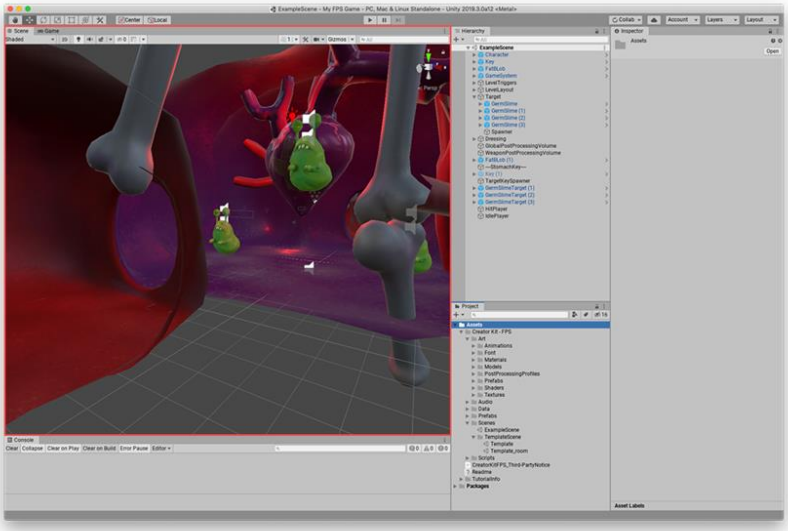
Slika 3.4: Prikaz igre [14]

3.4.5 *Prikaz scene*

Prikaz scene omogućava vizualno kretanje i uređivanje scene.Prikaz scene može prikazivati 2D ili 3D perspektivu, ovisno o odabranoj vrsti projekta. Prikaz je moguće koristiti za odabir i pozicioniranje prizora, objekata, kamere, rasvjete i ostalih objekata

igre[14].Slika 3.5 prikazuje prikaz scene.Prikaz scene se sastoji od navigacijskih kontrola koje omogućuju kretanje kroz scenu i u njih ubrajamo:

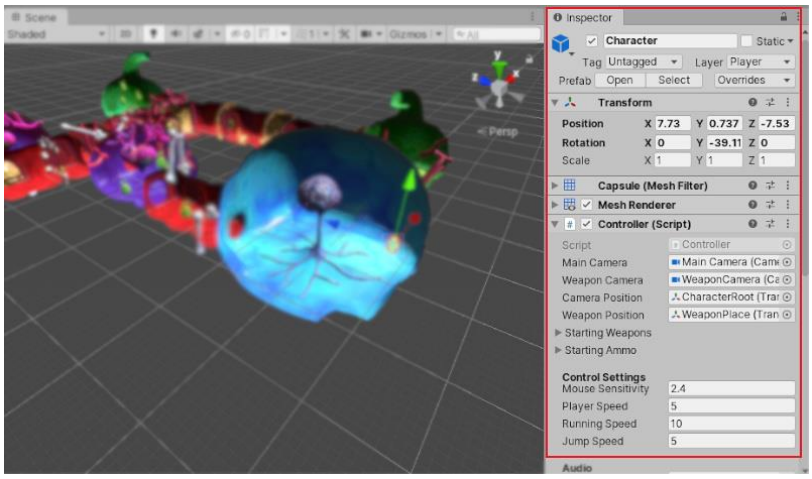
- ***Scene Gizmo***koji se nalazi u gornjem desnom kutu scene. *Scene Gizmo* prikazuje kameru, trenutnu orijentaciju kamere i omogućuje izmjenu kuta gledišta i načina projekcije,
- **Alate** pomakni, uvećaj i orbita,
- **Alat za centriranje** koji omogućuje centriranje scene prema objektu.



Slika 3.5: Prikaz scene [14]

3.4.6 *Inspektor*

Inspektoromogućuje pregled i uređivanje svih svojstava prethodno odabranog objekta.Projekti se sastoje se od više objekata koji sadrže skripte, zvukove, grafičke elemente i druge elemente koji tvore igru. Inspektor prikazuje informacije odabranog objekta, uključujući komponente, njihova svojstva i funkciju. Kada objekt sadrži zasebnu skriptu, inspektor prikazuje javne varijable koje su definirane u programskom kodu. Varijable mogu biti uređene na jednak način kao i ugrađene postavke komponenata [14].Prozor inspektora prikazan je na slici 3.5.



Slika 3.6: Prikaz inspektora [15]

3.4.7 ***Projekt***

Projektprikazuje sve datoteke koje su povezane s projektom i služi kao navigacija za brz pronalazak resursa. Projektni se prozor otvara prilikom izrade novog projekta [14]. Slika 3.7 prikazuje izgled projektnog okvira.Tablica 3.2 opisuje svojstva i kontrole projektnog prozora.

Tablica 3.2: Opis projektnog okvira [14]

Naziv kontrole	**Opis**
Izrada izbornika	Prikazuje resurse igre.
Pretraži	Pretražuje datoteke unutar projekta.
Pretraži po vrsti	Omogućuje ograničenu pretragu. Na primjer: *Mesh, Prefab*
Pretraži po oznaci	Omogućuje pretragu prema oznaci.
Broj skrivenih paketa	Određuje vidljivost paketa unutar projektnog prozora.

Slika 3.7: Prikaz projektnog prozora [14]

3.5 *Objekti igre*

Objekti igre (engl. *Game Objects*) osnovne su komponente igre koje predstavljaju likove, rekvizite i krajolike. Objekti služe kao spremnici komponenata, koje su dodijeljene i implementiraju stvarnufunkcionalnost. Komponenta transformacije dodijeljenaje objektu prema zadanim postavkama i nije ju moguće ukloniti. Ova komponenta služi za predstavljanje položaja i orijentacije. Ostale komponente koje objektu daju funkcionalnost mogu biti dodane unutar samog uređivačaili putem programskog koda, čija je skripta dodijeljena objektu [14].

3.5.1 ***Komponente***

Komponente su funkcionalni dio svakog objekta igre gdje objekti služe kao spremnici. Prema zadanim postavkama svi objekti sadrže komponentu transformacije, dok su ostale dodijeljene prema potrebi[14].

Komponenta transformacije (engl. *Transform*)određuje položaj, rotaciju i skaliranje svakog objekta. Ova se komponenta sastoji od nekoliko polja koja sadrže vlastita svojstva ifunkcije[14].Tablica 3.3 opisuje svojstva komponenti transformacije, kao i njihovu glavnu funkciju.

Tablica 3.3: Svojstva komponente transformacije [15]

Svojstvo	**Funkcija**
Pozicija (engl. *Position*)	Položaj transformacije unutar X,Y i Z koordinata
Rotacija (engl. *Rotation*)	Rotacija transformacije oko X, Y i Z osi, mjereno u stupnjevima
Skaliranje (engl. *Scale*)	Transformacija duž X, Y i Z osi. Vrijednost 1 je izvorna veličina.

Rigidbody 2Dje komponenta koja pridružuje objektu svojstva fizikalnog motora, dostupna pri izradi 2D i 3D projekata. Fizikalni motor omogućuje pomicanje komponente Collider 2D koja se vežeuz*Rigidbody 2D* komponentu. *Rigidbody 2D,Collider2D i* komponenta transformacijemeđusobno surađuju. Kada je komponenta *Collider2D*vezana uz*Rigidbody 2D,* ove dvije komponente djeluju kao jednai zajedno se kreću[14].

Collider 2Dkomponente definiraju oblik 2D objekata za potrebe fizičkih kolizija. *Collider* je nevidljiva komponenta i koristi se uz komponentu *Rigidbody 2D*[14].Vrste *Collider 2D* komponenata koje mogu biti vezane uz *Rigidbody 2D*su:

- *Circle Collider 2D* za područja kružne kolizije
- *Box Collider 2D* za područja kružne kolizije
- *Polygon Collider 2D* za područja pravokutne kolizije
- *Edge Collider 2D* za područja prostoručne kolizije
- *Capsule Collider 2D* za područja kružne ili rombidne kolizije
- *Composite Collider 2D* za pridruživanje *Box Collider 2Ds Polygon Collider 2D*

2D materijal (engl. *Physics Material 2D*) koristi se za definiranje trenja i odskoka koji se javljaju između 2D fizičkih objekata.Materijali mogu biti samostalno stvoreni. Tablica 3.4 opisuje svojstva 2D materijala i njihovu funkciju[14].

Tablica 3.4: Svojstva 2D materijala

Svojstvo	**Funkcija**
Trenje	Koeficijent trenja za *2D Collider*
Odskok	Stupanj koji definira odskok kolizije s tla. Vrijednost 0 ukazuje na to da nema odbijanja, a vrijednost 1 označava savršen odskok bez gubitka energije.

3.5.2 ***Oznake***

Oznake (engl. *Tags*) određuju referentnu riječ koja može biti dodijeljena jednom objektu ili više njih. Primjer 3.1: moguće je definirati oznaku imena igrač za sve objekte koji služe kao igrači ili neprijatelj za likove koji nisu pod kontrolom igrača. Predmeti koje igrač može sakupiti u igri, poput novčića mogu sadržavati oznaku novčić.Oznake pomažu identificirati objekte za potrebe pisanja programskog koda. Oznake su korisne radu s okidačima u unutar *Collidera* u samoj skripti. Primjer3.2:pomoću funkcije *GameObject.FindWithTag* moguće je pronaći objekt na način da ga drugi objekt traži prema oznaci. Osim samostalne izrade oznaka, *Unity* sadrži zadane oznake koje je moguće upotrijebiti[14].

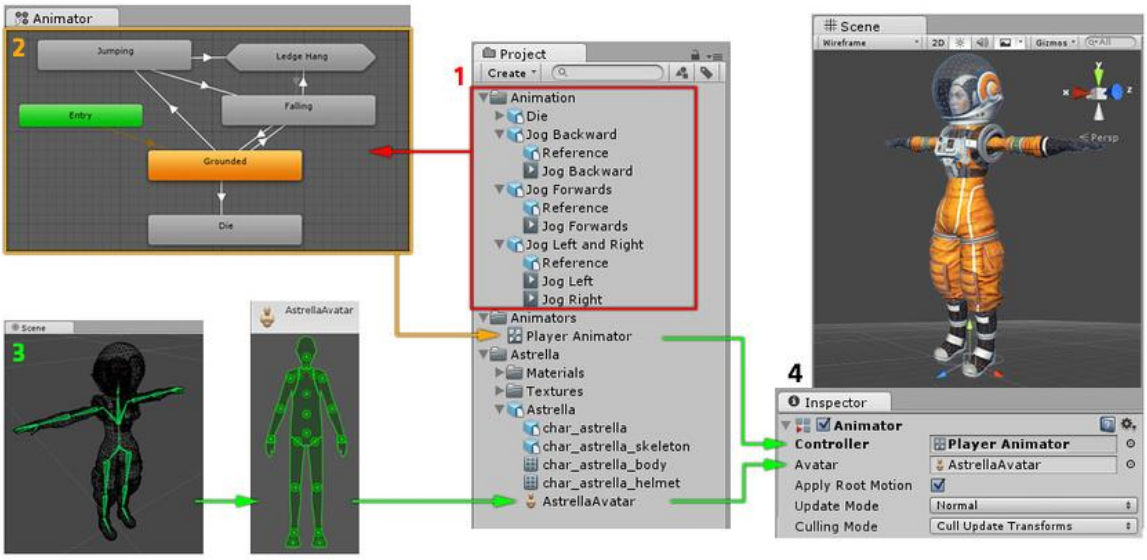
3.5.3 ***Sprites***

Sprites(engl. Sprites) predstavljaju 2D grafičke objekte. Usporedno s 3D načinom rada,*Sprites*definiraju osnovne teksture koje zahtijevaju posebne tehnike upravljanja tijekom rada na projektu. *Unity*sadrži*Sprite Creator, Sprite Editor, Sprite Renderer i Sprite Packer*[14].

- ***Sprite Creator***omogućuje stvaranje *Sprite* rezervnog mjesta u projektu,
- ***Sprite Editor***omogućuje izdvajanje grafike iz veće slike i uređivanje određenog broja komponenata unutar slike koja sadrži jednu teksturu,
- ***Sprite Renderer***omogućuje grafički prikaz,
- ***Sprite Packer***optimizira upotrebu i performanse video memorije u projektu.

3.5.4 ***Animacije***

Unity pruža bogat i moderan animacijski sustav koji omogućuje jednostavnu izvedbu animacija. Animacijski sustav zasnovan je na konceptu *Animation Clips*koji sadrži informacije kako bi pojedini objekti trebali mijenjati položaj, rotaciju itd. Animacijske isječke umjetnici najčešće stvaraju upotrebom alata poput *Autodesk 3DS Max, Autodesk Maya* i drugih[14]. Isječci se zatim organiziraju u strukturirani sustav nalik dijagramu toka koji se naziva *Animator Controller*.*Animator Controller*prati i pruža informacije o tome kako bi se pojedini isječak trebao prikazati. Animacijski sustav pruža posebne značajke za rad s humanoidnim likovima pomoću *Unity Avatar*sustava.*Animation Clips, Animator Controller* i *Unity Avatar* dodjeljuju se objektu upotrebom *Avatar* komponente[14].



Slika 3.8: Izrada animacije u Unity programskom okruženju [14]

3.6 *Unity*2D i 3D način rada

Iako je poznat po svojim 3D mogućnostima, *Unity* se koristi i za izradu 2D igara. Tip projekta određuje se prilikom stvaranja, no međutim postoji mogućnost naknadne izmjene izrađenog projekta.Prilikom izrade 3D igara, najčešće se koristi trodimenzionalna geometrija, a materijali i strukture prikazani su na površini objekata testvaraju realno okruženje. Kretnje kamere sežu unutar i oko scena te su popraćene svjetlom i sjenama po uzoru na realnost. Scene 3D igara, obično su prikazane koristeći perspektivan način rada kamere pa se objekti prikazuju veći nego što jesu. Ponekad se u igrama koristi 3D način rada uz upotrebu ortografske kamere. To je uobičajena tehnika koja se koristi u igrama koje pružaju ptičju perspektivu. Ovaj se način ponekad naziva i 2.5D. Pri izradi ovakvog tipa igre, potrebno je raditi u 3D načinu rada iako ne sadrži perspektivan prikaz, izrada se vrši uz rad s 3D modelima i resursima igre. Potrebno je promijeniti način rada kamere i prikaza scene u ortografski način rada.Primjer 3.3: kamera može prikazati pomicanje sa strane, alikretnje uređaja moguće su samo u dvije dimenzije. Na ovaj način igra i dalje koristi 3D modele koji stvaraju prepreke i 3D perspektivu kamere[14]. Tablica 3.5 opisuje osnovne razlike između 2D i 3D načina rada.

Tablica 3.5: Postavke u 2D i 3D načina rada [14]

<table>
<tr><th colspan="2">2D način rada</th><th colspan="2">3D način rada</th></tr>
<tr><td colspan="2">Uvezene se slike smatraju kao 2D slike i postavljene su u 2D način rada.</td><td colspan="2">Uvezene se slike ne smatraju se 2D slikama.</td></tr>
<tr><td colspan="2">Sprite Packer: omogućen</td><td colspan="2">Sprite Packer: onemogućen</td></tr>
<tr><td colspan="2">2D pogled na scenu</td><td colspan="2">3D pogled na scenu</td></tr>
<tr><td colspan="2">Zadani objekti nemaju svjetlo, usmjereno u stvarnom vremenu</td><td colspan="2">Zadani objekti imaju svjetlo, usmjereno u stvarnom vremenu</td></tr>
<tr><td colspan="2">Pozicija kamere: 0, 0, -10</td><td colspan="2">Pozicija kamere: 0, 1, -10</td></tr>
<tr><td colspan="2">Ortografski način rada kamere</td><td colspan="2">Perspektivan način rada kamere</td></tr>
<tr><td rowspan="5">Lighting Window</td><td>Skybox: onemogućen za nove scene</td><td rowspan="5">Lighting Window</td><td>Skybox: ugrađeni zadani Skybox materijal</td></tr>
<tr><td>Ambient Source: Color</td><td>Ambient Source: Skybox</td></tr>
<tr><td>Realtime Global Illumination: isključen</td><td>Realtime Global Illumination: uključen</td></tr>
<tr><td>Baked Global Illumination: isključen</td><td>Baked Global Illumination: uključen</td></tr>
<tr><td>Auto-Building: isključen</td><td>Auto-Building: uključen</td></tr>
</table>

3.6.1 ***2D način rada***

Najistaknutija karakteristika 2D načina rada jest*Mode* kontrola. Kada je omogućen 2D način rada, automatski je postavljen i ortografski prikaz kamere. Kamera gleda duž Z osi Y osi koja je uvećana prema gore što omogućuje vizualizaciju scene iako se koriste 2D objekti[14].

3.6.2 ***Postavke igrača 2D platforme***

Postavke igrača dostupne su za svaku od navedenih platformi, no u ovom slučaju razmatrat će se postavke igrača *Android* platforme. Slika 3.9 prikazuje postavke igrača prilikom izrade projekta za*Android* platformu. Postavke se sastoje od funkcija koje se razlikuju prema odabranoj platformi[14].

Postavke igrača prilikom odabira *Android* platforme čine:

- Ikona koja služi za prikaz na radnoj površini ili drugim platformama,
- Rezolucija i prezentacijasadrže postavke razlučivosti zaslona te ostale pojedinosti prezentacije, poput orijentacije,
- *Splash* slika prikazana je tijekom pokretanja igre,
- Postavke objavljivanja prikazuju detalje koji se koriste prilikom prijenosa na trgovinu,
- XR postavkekoje su specifične za *VR, AR ili MR.*

Slika 3.9: Postavke igrača *Android* platforme [15]

3.7 Integrirano razvojno okruženje *Visual Studio*

Visual Studio je integrirano razvojno okruženje koje je razvio *Microsoft*.*VisualStudio* objedinjuje sve aspekte potrebne pri razvoju softvera kao što suplaniranje, dizajn korisničkog sučelja, programiranje, testiranje, uklanjanje pogrešaka, analiza kvalitete.*Visual Studio* pruža programsku podršku za programske jezik *C#, C i C++, JavaScript, F#* i *Visual Basic*. Razvojno okruženje može biti korišteno pri izradi:

- Aplikacija, računalnih i mobilnih igara za razne platforme,
- Web stranica i servisa temeljenih na *ASP.NET, Jquery, AngularJS,*
- Igara i aplikacija za *Windows* uređaje, uključujući i *Xbox*koristeći*DirectX*[15].

3.7.1 *Programski jezik C#*

C# jeobjektno orijentirani programski jezik, prije svega namijenjen i razvijen za *.NET*platformu.Jezik je dizajnirao manji tim razvojnih programera uz vodstvo *Microsoftovih* inženjera Andersa Hejselberga i Scotta Wiltamuth. Jezik je izrađenpo uzoru

na *C++*, *JavuiPascal*, a sama je sintaksa vrlo slična Javinoj.Većina objektno orijentiranih programskih jezika sadrži podršku te omogućuje definiranje i rad s klasama koje definiraju novi tip podataka. Za razliku od većine jezika, *C#* sadrži ključne riječi koje služe pri deklariranju klasa, metoda i poliformizma [16].

Programski kod 3.1: Pozdrav svijete napisan u programskom jeziku *C#*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ZdravoSvijete
{
class Program
   {
Static voidMain(string[] args)
{
Console.WriteLine(“Pozdrav Svijete!”);
}
    }
}
```

3.8 Adobe Illustrator

AdobeIllustrator softverski je program za izradu vektorskih crteža i ilustracija. Program jepredstavila tvrtka *Adobe Systems* 1987. godinekao grafički uređivač i aplikacija za izradu logotipa. Ideja izvorne platforme, bila je integracija matematičkih jednadžbi kako bi se postigle glatke i iskrivljene linije te oblici, poznati pod nazivom *Bezier Curves*. Tokom godina i unaprijeđenih verzija, program je postao dio *Adobe Creative Clouda*, skupa aplikacija i servisa kojima upravlja *Adobe Inc.*

AdobeIllustrator pretežno koriste grafički dizajneri, web dizajneri, profesionalni ilustratori i hobisti za stvaranje digitalnih i ispisnih slika, uključujući ilustracije, grafikone, dijagrame, logotipe i drugo. Jedna od najvažnijih značajki programa jest kvaliteta umjetničkog djela, stvorena pomoću programa. Djelo ne ovisi o rezoluciji u kojoj se prikazuje, odnosno slika se može smanjiti ili uvećati bez gubitka kvalitete. Ovo je atribut vektorskog crtanja gdje se koriste matematičke veze pri izradi linija, lukova i drugih komponenata. Vektorska grafika skup je poligona koji tvore sliku i sastavljeni su od vektora. Svaki vektor prolazi kroz mjesto poznato kao čvor ili kontrolna točka, definirana na x i y osi. Uloga čvora je određivanje putanje vektora koji sadrži različite atribute poput

boje, krivulje, ispuna, oblik i debljina. Položaj vektora može se međusobno povezati s matematičkim formulama koje precizno preračunavaju položaj kada se promjeni veličina slike[17, 18].Slika 3.9 prikazuje primjer objekta koji jeizrađen uprogramu *Adobe Illustrator*.Program posjeduje sposobnost kreiranja i modificiranja vektorske grafike, koja zahtjeva spremanje u određene formate vektorske grafike. Često korišteni formati vektorske grafike uključuju: *SVG, PDF, EPS, WMF, VML*.

PDF (engl. *Poratble Document Format*) omogućuje prikaz slika i teksta neovisno o hardveru, softveru ili operacijskom sustavu. *PDF* sadrži potpuni opis dokumenata, izgled dokumenta, korištene fontove, grafiku i tekst. Ovaj format uključuje strukturirani sustav za pohranu koji je skup elemenata i tvori jednu datoteku. Također uključuje podskup *PostScript-a* za generiranje grafike i sustav za pridruživanje fontova dokumentima [17, 18].

EPS (engl. *Encapsulated PostScript*) je podskup *PostScript* formata s dodatnim ograničenjima koja omogućavaju pohranu grafičkih datoteka. Grafičke datoteke obično su samostalne i mogu biti smještene u drugu *PostScript* datoteku. EPS datoteka u osnovi je *PostScript* program koji sadrži pregled slike u niskoj rezoluciji [17, 18].

WMF (engl. *Windows Metafile*) je format koji je *Windows* izvorno koristio devedesetih godina, a *Illustrator* ga je mogao izvoziti. Ovaj format može pohraniti vektorsku grafiku i mape, omogućujući da ga se koristi na način sličan *SVG* formatu. *WMF* datoteke sadrže popis poziva funkcija koje *GDI* koristi za prikaz slike [17, 18].

VML (engl. *Vector Markup Language*) je bio *XML* format dvodimenzionalne vektorske grafike u sklopu *Office Open XML* standarda. Od 2012. godine *Internet Explorer* više ne podržava *VML* format, no još je uvijek uključen u *Office Open XML* za određene svrhe [17, 18].

SVG (engl. *Scalable Vector Graphics*) format temelji se na *XML* jeziku koji podržava dvodimenzionalnu grafiku za izradu interaktivnih slika i animacija. Slike su definirane pomoću *XML* datotekama što omogućuje sažimanje, indeksiranje i skriptiranje te pretraživanje. Datoteke se mogu uređivati upotrebom raznih uređivača i zamjenskim aplikacijama za crtanje. *World Wide Web Consoritum (W3C)*podržava *SVG* specifikacijekao otvoren standard od 1999. godine[17, 18].

Slika 3.10: Primjer vektorske grafike

3.9 *AdobePhotoshop*

Adobe Photoshop softverski je program koji služi za obradu slike, razvijen 1987. godine. Program se koristi za uređivanje slika, retuširanje, stvaranje kompozicija, izradu web dizajna itd. Od jednostavnog alata za uređivanje slika *Photoshop* se razvio u sveobuhvatni paket za upravljanje slikama.Ranija verzija softvera, nastala je u veljači 1990. godine i omogućavala je korisnicama prikazivanje i spremanje datoteka u više formata na *MacOS* uređajima. Korisnici su također mogli prilagoditi nijansu, ravnotežu i zasićenost slika[19, 20].

Adobe Photoshop najvažniji je alat za dizajnere, grafičke umjetnike, fotografe i kreativne profesionalce.Usporedno s programom*AdobeIllustrator*koji koristi vektorsku grafiku i služi za stvaranje iste, *AdobePhotoshop* koristi se za stvaranje rasterske grafike. Rasterska grafika najviše se koristi za nelinearne umjetničke slike, digitalizirane fotografije, skenirane umjetnine i sl. Nelinearne umjetničke slike najbolje su predstavljene u rasterskom obliku jer one obično uključuju gradacije, nedefinirane crte i oblike te složenu kompoziciju.Rasterske slike temelje se na pikselima i trpe degradaciju slike. Uobičajeni rasterski formati uključuju *TIFF, JPEG, GIF, PCX i BMP*[20].

4. IMPLEMENTACIJA STEM IGRE ZA *ANDROID* PLATFORMU

4.1 *STEM*

Znanost, tehnologija, inženjerstvo i matematika (*STEM*) prožimaju svaki aspekt današnjeg svijeta. Inovacije koje proizlaze iz navedenih područja podupiru ekonomski razvoj koji vodi ka uspostavljanju kreativnih poduzeća i nagrađivanih karijera. *STEM* obrazovanje višestruko nadilazi glavne discipline koje čine *STEM* akronim. Temelji obrazovanja započinju u ranom djetinjstvu, od najranijih godina, kroz igru i okruženje koji mogu služiti kao motivacija za znanost, tehnologiju, inženjering i matematiku. Mala djeca prirodno se uključuju u rano istraživanje *STEM*-a kroz praktična i kreativna iskustva te aktivnosti. Kroz samo iskustvo djeca stvaraju radoznalost, kritičko razmišljanje i samostalno rješavanje problema koji su izgrađeni na temelju njihovog iskustva.

Znanost omogućava razvijanje interesa za životno, materijalno i fizičko razumijevanje svijeta te razvija vještine suradnje, istraživanja i kritičkog ispitivanja te eksperimentiranja.
Prema Državnom zavodu za statistiku, trenutna klasifikacija znanosti sastoji se od devet znanstvenih područja. Pri izradi rada, odnosno odabiru sadržaja za izradu igre korištene su znanstvene grane iz područja prirodnih znanosti i područje tehničkih znanosti[21].
Nadalje, klasifikacija znanosti i polja sastoji se od:

- Područja prirodnih znanosti
- Područja tehničkih znanosti
- Područja biomedicine i zdravstva
- Područja biotehničkih znanosti
- Područja društvenih znanosti
- Područja humanističkih znanosti
- Umjetničko područje
- Interdisciplinarna područja znanosti
- Interdisciplinarna područja umjetnosti

Tehnologija pokriva niz područja koja uključuju primjenu znanja, vještina i računalnog razmišljanja kako bi se proširile ljudske mogućnosti i zadovoljile ljudske želje i potrebe.

Inženjerstvo se bavi dizajnom i kreiranjem proizvoda i procesa, oslanjajući se na znanstvene metode potrebne za rješavanje problema.

Matematika uči, te osposobljava vještine potrebne za interpretaciju i analizu informacija, rješavanje problema, procjenu rizika, pojednostavljenje problema i razumijevanje svijeta kroz modeliranje apstraktnih i konkretnih problema.

4.2 Opis igre

Edukativna *STEM* igra namijenjena je dobnoj skupini od 6-10 godina, odnosno učenicima od prvog do 4 razreda osnovnih škola. Također, igra može biti predviđena i mlađoj, dobnoj skupini od navedene tj. naprednim skupinama mlađe dobi. Igra se sastoji od zabavnog dijela, iznanstvenogkoji zahtijeva logičko razmišljanje.Cilj igre jest upoznavanje pojedinca s osnovnim pojmova iz *STEM* područja kako bi stvorilo interes prema određenoj znanosti, grani znanosti ili pojedinom području. Interesi stvoreni u mlađoj dobi vode ka željama u budućnosti, kao što su daljnje obrazovanje i rad. Razvoj igre zahtjeva i izradu razvojnog plana koji uključuje istraživanje i prikupljanje informacija, planiranje, izradu dizajna. Jasan koncept ciljeva olakšava planiranje koraka koji su popraćeni prilagodbama prilikom razvoja.

Slika 4.1 prikazuje korisničko sučelje.Korisničko sučelje predstavlja početno stanje igre i sastoji se od tri gumba:

1. Start – omogućuje ulazak u igru, odnosno prijelaz na sljedeću scenu.
2. Upute – omogućuje čitanje uputa u kojima je objašnjeno kako igrati igru.
3. Izlaz– omogućuje izlaz iz igre

Svima trima gumbima je dodijeljen objekt igre koji sadrži skriptu *IzmjenaScene*i *AudioSource*komponentu. Dodijeljena skripta omogućuje izmjenu scene s vremenskom odgodom, dok *AudioSource* omogućuje izvođenje pozadinske glazbe.

Slika 4.1: Korisničko sučelje igre

4.2.1 *Dizajn korisničkog sučelja*

Dizajn korisničkog sučelja (engl. *UI Design*) odnosi se na cjelokupan dizajn igre. Pri izradi dizajna potrebno je obratiti pozornost na to tko je krajnji korisnik i komu je igra namijenjena.Za izradu dizajna korisničkog sučelja i grafičkih komponenata korišteni su programski alati *AdobeIllustrator* i *AdobePhotoshop*.Uz osnovnu ideju i uz poznavanje rada programskih alata, proces izrade može biti vrlo jednostavan. Slika 4.2 prikazuje dizajn elemenata korisničkog sučelja. Prikazani su gumbi korišteni pri izradi igre.

Slika 4.2: Dizajn elemenata korisničkog sučelja

4.2.2 ***Logotip***

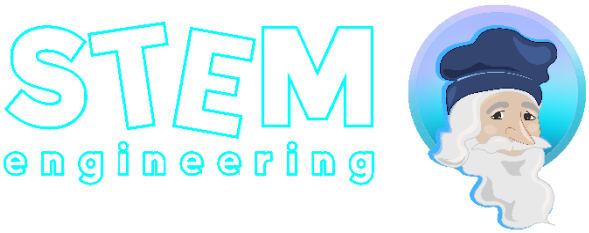
Dizajn i logotip igre izrađeni su na način da bi stvorili interes kod djece. Slika 4.2 prikazuje je logotip igre.Logotip igre sastoji se od teksta i predstavlja zaštitni znak igre.Tablica 4.1 prikazuje obilježja primarnog logotipa *STEM* igre. Logotip je izrađen uz korištenje fonta *IndigoOutline* i odabirom boje koja će biti jasno istaknuti tekst na tamnijoj pozadini. Logotip je popraćen grafičkim sredstvima koji su vidljivi isključivo na tamnoj pozadini.

Slika 4.2: Primarni logotip igre

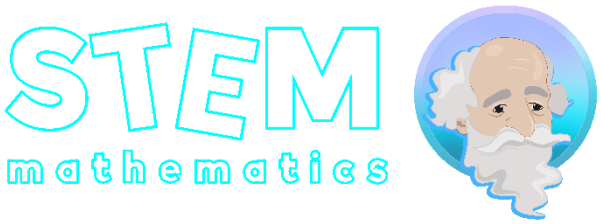
Tablica 4.1: Obilježja primarnog logotipa STEM igre

Tipografija:	*Indigo Outline Font Regular*		
Primarna boja:	#00ffff	R 0 G255 B255	C100% M0% Y 0% K 0%
Slobodan prostor:	235 px		

Naknadno su izrađeni i sekundarni logotipi koji simboliziraju zasebne skupine, odnosno znanost, tehnologiju, inženjering i matematiku. Vizualni identiteti sadrže paletu boja, tipografiju, ilustraciju i grafička sredstva. Postoji mogućnost proširivanja elemenata vizualnog identiteta, no to u ovom slučaju nije nužno. Slike 4.3, 4.4, 4.5 i 4.6 prikazuju sekundarne logotipe igre.



Slika 4.3: Sekundarni logotip – inženjering

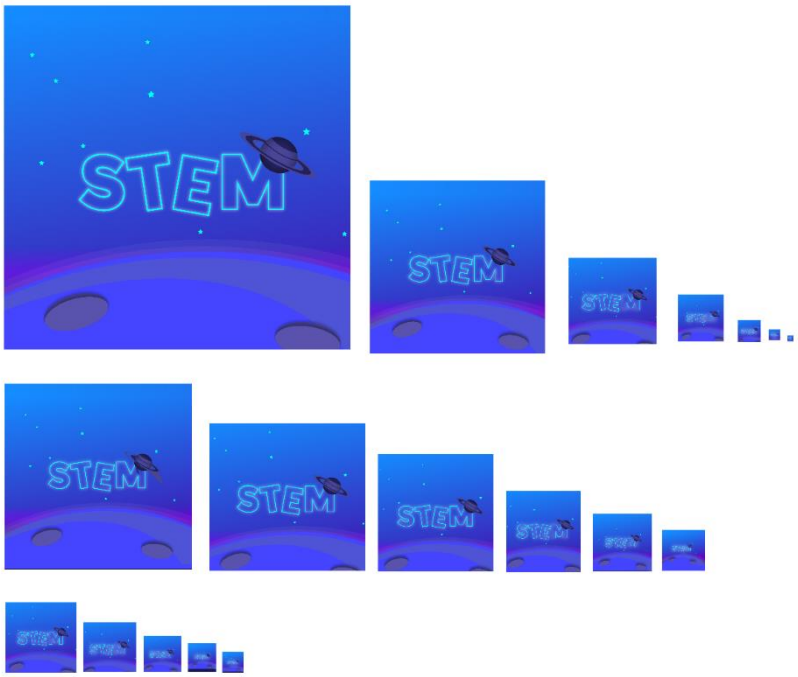


Slika 4.4: Sekundarni logotip – matematika

Slika 4.5: Sekundarni logotip – tehnologija

Slika 4.6: Sekundarni logotip – znanost

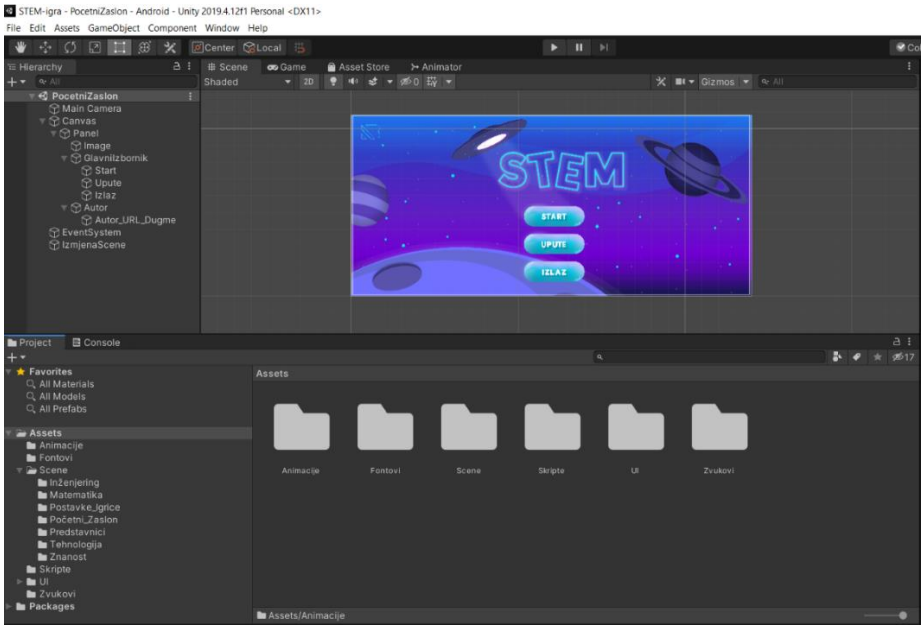
Ikona(engl. *Favicon*) igre izrađena je u programu*AdobeIllustrator*i predstavlja prenosivu sliku koja karakterizira igru. Ikona igre nosi vrlo jednostavan naziv *STEM*.Standardna veličina ikone iznosi 512 x 512 px. *Unity* standardnu veličinu ikone prikazuje na svim platformama, no moguće je zasebno zamijeniti ikonu za pojedinu platformuunutarpostavki igrača. Slika 4.7 prikazuje izvedbe ikone za potrebe različitih platforma.



Slika 4.7: Ikona igre

4.3 *Unity* projekt

Unity projekt nosi naziv *STEM*-igra.Projekt se sastoji od različitih resursa i elemenata kao što su zvukovi, grafičke komponente, ilustracije, skripte s programskim kodom, animacije itd. Slika 4.8 prikazuje izrađeni projekt koji nosi naziv STEM-igra.

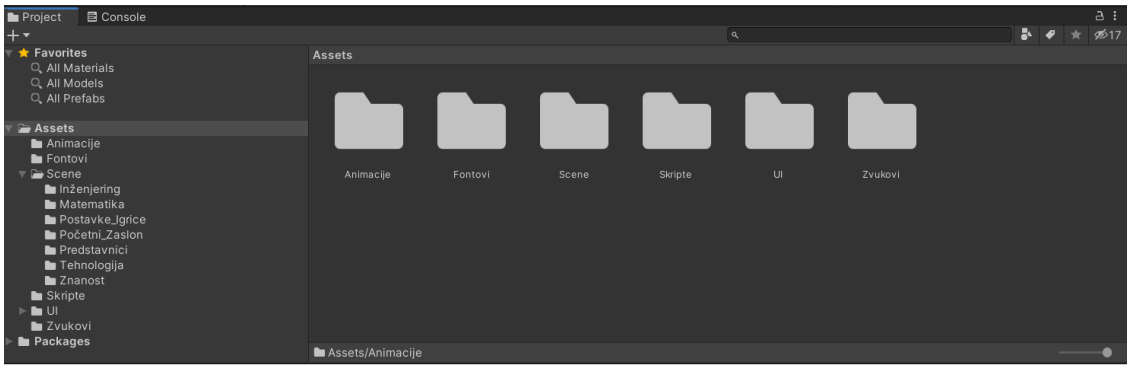


Slika 4.8: Prikaz projekta

Resursi igre smješteni suu zasebne mape kao što je prikazano na slici 4.9.

Glavna podjela sastoji se odsljedećih mapa:

1. **Animacije** – mapa sadrži izrađene animacije,
2. **Fontovi –** mapa sadrži fontove koji su korišteni pri izradi igre.
3. **Scene –** mapa sadrži izrađene scene koje su naknadnosmještene u zasebne mape,
4. **Skripte** – mapa sadrži *C#* skripte s programskim kodom,
5. **UI** – mapa sadrži grafičke komponente koje tvore dizajn korisničkog sučelja,
6. **Zvukovi**– mapa sadrži korištene zvukove pri izradi igre.



Slika 4.9: Komponente projekta

4.4 Programski kod

Programski kod je izrađen u programskom jeziku*C#*unutarintegrirano razvojnog okruženja *VisualStudio*.Skripte s programskim kodom pridružene su objektima igre i pružaju određenu funkcionalnost.*Unity* prema zadanim postavkama prilikom izrade nove skripte dodjeljuje funkcije*Start* i *Update* koje igra poziva u različitim intervalima. Funkcija *Start* poziva sejednom kada je igra pokrenuta, a funkcija *Update* jednom po kadru.

Funkcija *Update*je mjesto za definiranje programskog koda koji se odnosi na okvir igre za pojedini objekt. To može uključivati kretanje, pokretanjei reakciju na korisnički unos.Sve metode i funkcije moraju biti pozvane u *Update* ili *FixedUpdate* funkciji.

Funkcija *Start*biti će pozvana prije samog pokretanja igre, odnosno prije nego što se pozove funkcija *Update*. Funckija *Start*služi za izradu inicijalizacije. Važno je napomenuti da se inicijalizacija objekata ne vrši putem konstruktora.

Funkcija *Awake* poziva se samo jednom, prijefunkcije*Start* i izvodi se dok se igra učitava.Ova se funkcija koristi za inicijalizaciju varijabli ili stanja igre prije samo početka. Svako buđenje odvija se nasumičnim redoslijedom između objekata .

Funkcija*FixedUpdate*poziva se tijekom svakog okvira, što znači da će vremenski interval između svakog *FixedUpdate* biti jednak. Ova funkcija sadrži frekvenciju fizike.

Funkcija *LateUpdate* poziva se nakon svih *Update*funkcija. Ova se funkcija u osnovi koristi za praćenje kamere, odnosno predmeta.

Programski kod 4.1 prikazuje dodijeljenu *C#* skriptu prilikom stvaranja novog projekta. Skripta stvara vezu s radom unutar *Unity*-a na način da implementira klasu koja proizlazi iz ugrađene klase *MonoBehaviour*. Klasa može služiti kao predložak za stvaranje nove komponente koja može biti pridružena objektu igre. Svaki put kada se komponenta pridružuje pojedinom objektu, stvara se nova instanca istog objekta. Naziv klase preuzima se iz imena koje je navedeno prilikom stvaranja datoteke. Također, naziv klase i datoteke mora biti isti kako bi komponenta bila pridružena objektu.

Programski kod 4.1: Zadana C# skripta

```
using System.Collections;
using System.Collections.Generic;
usingUnityEngine;

publicclassNewBehaviourScript : MonoBehaviour
{
// StART is called before the first frame update
void StART()
    {

    }

// Update is called once per frame
void Update()
    {

    }
```

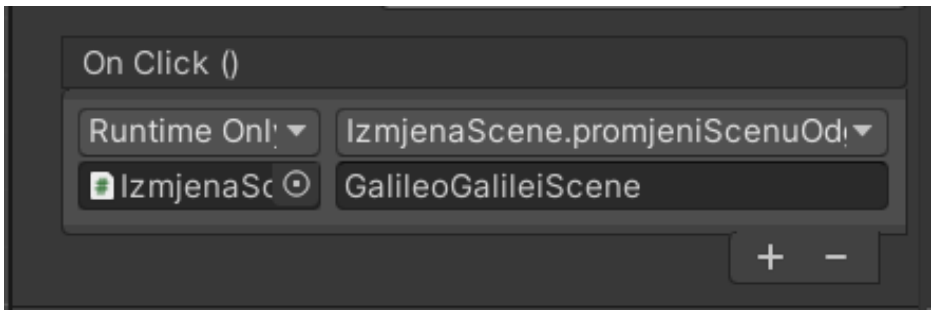
4.4.1 *Izmjena scene*

Programski kod 4.2 opisuje izmjenu scena igre. Svaka scena sadrži zadatak koji je potrebno prijeći odabirom točnogodgovora kako bi se pristupilo sljedećoj sceni. Unutar skripte poziva se korutina *PromjenaSceneOdgoda*. Izvršava se promjena scene prema

nazivu koji definiran unutar samog *UnityEditora* te sadrži odgodu od 0.2f. Odgoda je definirana da bi se postigao određen period između zvuka gumba i izmjene scene. Skripta je dodijeljena praznom objektu igre *Izmjena_Scene_Odgoda2f* te je isti dodijeljen točnom odgovoru. Slika 4.10 prikazuje odabir sljedeće scene koja će biti prikazana nakon definirane vremenske odgode prilikom pritiska gumba.

Programski kod 4.2: Izmjena scene

```
using System.Collections;
using System.Collections.Generic;
usingUnityEngine;
usingUnityEngine.SceneManagement;

publicclassIzmjenaScene : MonoBehaviour
{
publicvoid promjeniScenuOdgoda(string naziv_scene)
    {
        StARTCoroutine(PromjenaSceneOdgoda(naziv_scene));

    }
public IEnumerator PromjenaSceneOdgoda(string naziv_scene)
    {
yieldreturnnew WaitForSeconds(0.5f);
        SceneManager.LoadScene(naziv_scene);
    }

}
```



Slika 4.10: Definiranje izmjene scene korištenjem skripte

4.4.2 ***Prikaz teksta***

Programski kod 4.3 opisuje način prikaza teksta u obliku dijaloga. Klasa *EfektTeksta* sadrži definirane tipove varijabli *Float* i *String*. Poziva se korutina *PrikazTeksta* koja definira određenu vremensku odgodu. Na ovaj je način postignut efekt *Typewritera*kojim se pokušava stvoriti vizualni efekt dijaloga.

Programski kod 4.3: Način ispisa teksta

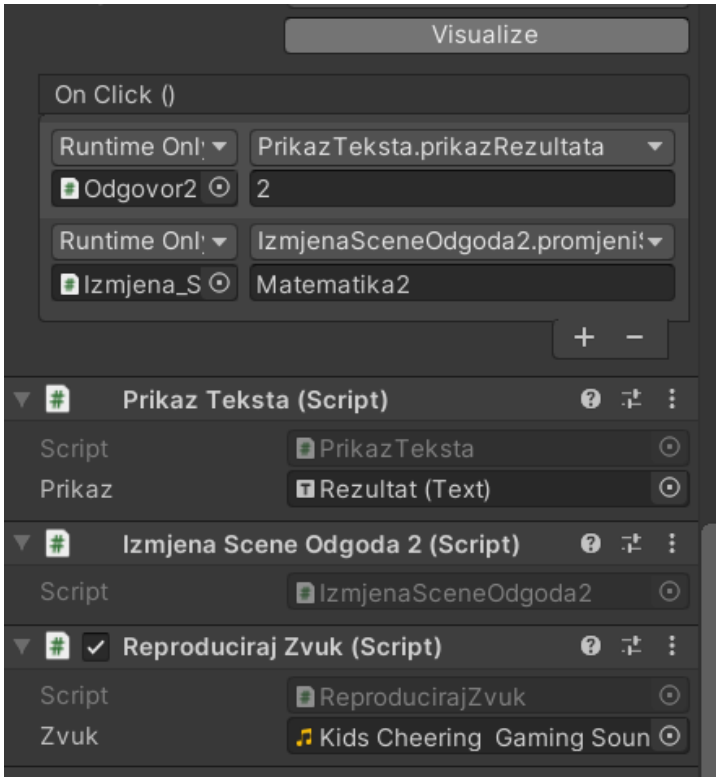
```
using System.Collections;
using System.Collections.Generic;
usingUnityEngine;
usingUnityEngine.UI;

publicclassEfektTeksta : MonoBehaviour
{
publicfloat odgoda = 0.1f;
publicstring tekstPrikaza;
privatestring trenutniTekst = "";

voidStARTO
    {
        StARTCoroutine(PrikazTeksta());
    }

    IEnumerator PrikazTeksta()
    {
for (int i = 0; i < tekstPrikaza.Length; i++)
        {
            trenutniTekst = tekstPrikaza.Substring(0, i);
this.GetComponent<Text>().text = trenutniTekst;
yieldreturnnew WaitForSeconds(odgoda);
        }
    }
}
```

4.4.3 *Prikaz rezultata*

Programski kod 4.4 prikazuje rezultate koji su popraćeni pritiskom dugmeta. Definirano je tekstno polje naziva *Prikaz*. Programski kod omogućuje prikaz rezultata u tekstnom okviru te je dodijeljena svim ponuđenim odgovorima. Slika 4.11 prikazuje dodjeljenju skriptu za prikaz rezultata nakon pritiska gumba.

Programski kod 4.4: Prikaz rezultata

```
using System.Collections;
using System.Collections.Generic;
usingUnityEngine;
usingUnityEngine.UI;
usingUnityEngine.UIElements;

publicclassPrikazTeksta : MonoBehaviour
{
public Text Prikaz;

publicvoid prikazRezultata(string tekst)
    {
        Prikaz.text = tekst;
    }

}
```



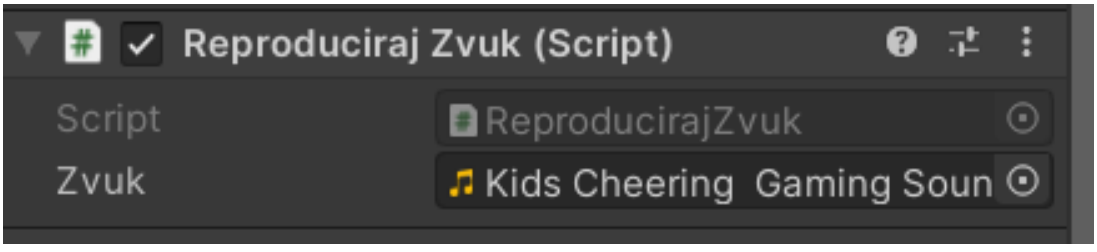
Slika 4.11: Prikaz rezultata

4.4.4 *Reprodukcija zvuka*

Programski kod 4.5 omogućuje reprodukciju zvuka koji se izvodi prilikom pritiska određenog gumba. Zvuk je dodijeljen unutar korisničkog sučelja. Slika 4.12 prikazuje odabir željenog zvuka za reprodukciju.

Programski kod 4.5: Reprodukcija zvuka

```
using System.Collections;
using System.Collections.Generic;
usingUnityEngine;
usingUnityEngine.UI;

[RequireComponent(typeof(Button))]
publicclassReproducirajZvuk : MonoBehaviour
{
public AudioClip zvuk;

private Button dugme
    {
get { return GetComponent<Button>(); }
    }
private AudioSource zvukDugmeta
    {
get { return GetComponent <AudioSource>(); }
    }

voidStART()
    {
        gameObject.AddComponent<AudioSource>();
        zvukDugmeta.clip = zvuk;
        zvukDugmeta.playOnAwake = false;

        dugme.onClick.AddListener(() => reproducirajZvuk());
    }

void reproducirajZvuk()
    {
        zvukDugmeta.PlayOneShot(zvuk);
    }
}
```



Slika 4.12: Odabir i reprodukcija zvuka

4.5 Izrada okruženja za *Android* platformu

Izrada mobilne igre za *Android* platformu u *Unity* okruženju zahtijeva specifične korake pri samoj izradi projekta koji uključuju:

1. Preuzimanje i instalaciju *Unity Huba-a*
2. Pokretanje *Unity Hub-a* i odabir verzije *UnityEditor* koja podržava 64-bitne aplikacije. 64-bitne verzije podržavaju *Android App Bundles* koji omogućavaju manja i optimizirana preuzimanja.
3. U *Unity* Hub-u, unutar instalacija, potrebno je izvršiti instalaciju modula *AndroidBuildSupporta*, koji treba uključivati *AndroidSDK* i *NDK* alate te *OpenJDK*.
4. Pokretanje i izrada novog projekta.
5. Preporučen je uvoz *Google* priključaka

Nakon uspješne instalacije modula i izrade projekta, na uređaju je potrebno omogućiti uklanjanje pogrešaka putem USB-a. Za omogućavanje ispravljanja pogrešaka putem USB-a potrebno je omogućiti opcije razvojnog programera na uređaju. U postavkama uređaja, unutar polja podaci o softveru, potrebno je pronaći broj verzije te dodirnuti sedam puta, nakon čega će se pojaviti skočna obavijest da je način programera uključen.

4.5.1 *Testiranje Android igre*

Pri izradi rada, za potrebe testiranja mobilne igre, korišten je mobilni uređaj*Samsung Galaxy S10 Lite*. Uređaj je redovito ažuriran te koristi *Android* verziju 10. Za potrebe testiranja igre za *Android* uređaj potrebno je instalirati aplikaciju *Android Remote 5* i slijediti navedene korake:

1. Potrebno je provjeriti postavke izgradnje. U postavkama je potrebno dodati scenu te označiti platformu i urediti postavke iste. Scene se unose na način povlačenja scene iz projekta. Scene se nalaze unutar mape*Assets>Scenes*. Nakon prijenosa scena, odabire se platforma, u ovom slučaju *Android* te se mogu urediti dodatne postavke. Pristup se vrši sljedećim naredbama: *File>BuildSettings*.Ovaj korak nije nužan pri testiranju, ali je nužan pri prijenosu igre na mobilni uređaj, no također se može koristiti u svrhe testiranja.
2. Potrebno je provjeriti jesu li instalirani alati potrebni za *Android* platformu.
   Pristup se vrši sljedećim naredbama: *Edit>Preferences>ExternalTools*.

3. Potrebno je odrediti tip uređaja i rezoluciju unutar postavka *Unity-a*. Prema zadanim postavkama tip uređaja nije postavljen. Potrebno je promijeniti postavke na bilo koji *Android* uređaj i normalnu rezoluciju.
   Pristup se vrši sljedećim naredbama: *Edit>ProjectSettings>Editor.*
4. Mobilni je uređaj potrebno spojiti s računalom putem USB-a, pokrenuti aplikaciju *Unity Remote 5* te pokrenuti igru na računalu.

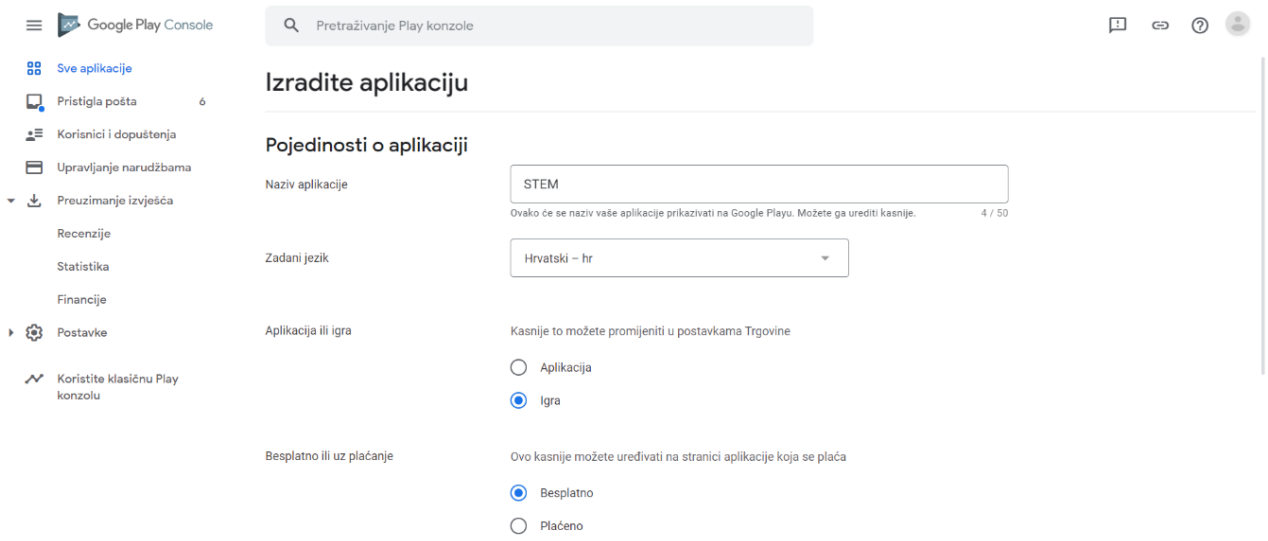
4.6 Google Play trgovina

Google Play, ranije poznat kao *Android Market* plasiran je u javnost 2008. godine kada je i postao dostupan za interakciju s krajnjim korisnicima. *Google Play* je internetska trgovina koja korisnicama omogućuje preuzimanje i kupnju sadržaja poput glazbe, videozapisa, knjiga, aplikacija, igara i sl. Trgovina je primarno namijenjena i dostupna *Android* uređajima. Korisnici imaju mogućnost pretraživati sadržaj, ocjenjivati ga i preuzeti te instalirati na uređaj. Kako bi korisnik pristupio *GooglePlay* trgovini, potrebno je izraditi te postaviti *Google* račun. Nakon registracije *Google* računa, korisnik može preuzeti sadržaj ili ga kupiti upotrebom kreditne kartice na *GoogleCheck out-u* ili pomoću naplate mobilnog operatera. Korisnik može poništiti zahtjev u vremenskom periodu od 15 minuta od trenutka kupnje.Razvojni programeri moraju registrirati račun putem *GooglePlay* konzole uz jednokratnu naknadu u iznosu od 25 američkih dolara kako bi mogli prenositi aplikacije u trgovinu. Objavljen sadržaj može biti besplatan ili se može naplatiti putem prilikom preuzimanja. Cijene sadržaja mogu varirati, odnosno, biti različite za svaku zemlju.Prilikom kupnje sadržaja, potrebno je izvršiti registraciju na *Google Checkout-u* te otvoriti korisnički račun. Registracija računa je besplatna, nakon čega se sve transakcije vode putem korisničkog računa.*Google* je također razvio sustav kupnje putem aplikacije, integrirane s *GooglePlay* trgovinom i *GoogleCheckout-om*. Zaseban *API* dostupan je programerima za obradu transakcija i kupnja putem aplikacije.

4.6.1 *Prijenos sadržaja na Google Play trgovinu*

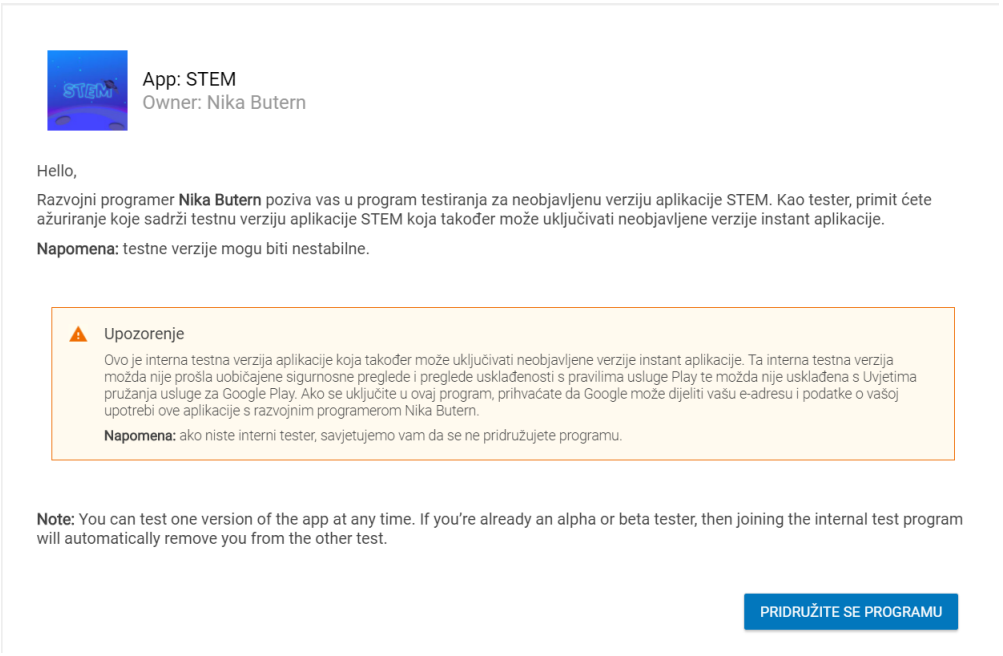
Nakon uspješno obavljene registracije na *GooglePlay* platformi moguće je pristupiti nadzornoj ploči i krenuti s izradom aplikacije. Prvi koraci obuhvaćaju ispunjavanje forme gdje je potrebno navesti naziv aplikacije, zadani jezik, odabrati tip aplikacije (aplikacija ili igra) te odrediti hoće li aplikacija biti besplatna za krajnje korisnike ili će biti preuzeta uz naknadu.Nakon uspješnog popunjavanja podataka o aplikaciji i prihvaćanja navedenih

deklaracija, izrađuje se novi projekt s nazivom aplikacije. Slijede koraci početnog postavljanja aplikacije, a oni uključuju određivanje parametara.Slika 4.21 prikazuje nadzornu ploču prilikom kreiranja nove aplikacije.



Slika 4.13: *GooglePlay* konzola

Nakon prethodno izvršenih koraka slijedi izrada izdanja za unutarnje testiranje. *Google Play* konzola zahtjeva *Android App Bundles* (.aab) ili *APK* datoteke čija je minimalna *API*razina 29. Unity prema zadanim postavkama nudi *API* razinu 28 koja nije kompatibilna za prijenos na *Google Play* konzolu te je potrebno izvršiti promjene unutar projekta.Unutar izvorne mape projekta, potrebno je izraditi praznu tekstnu datoteku *repositories.cfg*, a unutar *Unity Editora* potrebno je pristupiti postavkama igrača i izmijenitiminimalnu API razinu. Slijedi izmjena Java direktorija putem naredbenog retka koji mora biti pokrenut iz uloge administratoraWindows sustava i instalacija *Android SDK*.Kako bi bili zadovoljeni svi uvjeti za prijenos aplikacije, potrebno je izraditi ključ koji će služiti za potpisivanje igre i postaviti 32-bitnu i 64-bitnuverziju izvornog koda prilikom izvoza aplikacije. Slika 4.14 prikazuje uspješno prenesenu igru koja je spremna za interno testiranje.



Slika 4.14: Interno testiranje igre

5. ZAKLJUČAK

U ovom radu uspješno je realizirana ideja izrade prototipa edukativne *STEM* igre koja je namijenjena *Android* platformi. Prilikom izrade igre stečena su znanja programskog jezika C#, kao i raznih programskih alata koji su pridonijeli realizaciji projektnog zadataka.Samostalna izrada projektnog zadataka upoznaje pojedinca s problematikom te potiče na pronalaženje mogućih rješenja. Naglasak rada odnosi se razvoj igre pomoću*Unity Game Engine-a*uzkorištenje *C#*programskog jezikate naupoznavanje *Android* platforme.

Razvijena igra omogućuje krajnjem korisniku upoznavanje s osnovnim pojmovima iz *STEM* područja, na zanimljiv, vizualno istaknut način. Prednost igre jest jednostavnost, dočarana grafičkim elementima i fotografijama. Igra zadovoljava osnovne uvjete, no uz kontinuirani rad u budućnosti, njena bi se funkcionalnost mogla proširiti. Moguće buduće preinake mogle bi biti višejezičnost, izrada verzije za *IOS* platformu, usavršavanje UI-a igre te dorade funkcionalnosti.

Razvoj igre dovodi do novih saznanja i ideja koje bi mogle biti primjenjive u budućnosti na sličnim ili različitim projektima.

6. LITERATURA

[1]Alfieri C. *Android* Programming CookBook [Online]. Java Code Geeks; 2016.Dostupno na: http://enos.itcollege.ee/~jpoial/allalaadimised/reading/*Android*-Programming-Cookbook.pdf (10.06.2020.)

[2] *Android* Source. Set up for *Android* Development [Online]. Dostupno na: https://source.*Android*.com/ (10.06.2020.)

[3]Brahler S. Analysis of the *Android* Architecture [Online]. Karlsruhe institute of technology; 2010. Dostupno na: https://os.itec.kit.edu/downloads/sa_2010_braehler-stefan_*Android*-architecture.pdf(11.06.2020.)

[4]*Android* Developers. Platform Architecture [Online]. Dostupno na: https://developer.*Android*.com/guide/platform (12.06.2020.)

[5] Apple Inc. WebKit [Online]. Dostupno na: https://webkit.org/ (12.06.2020.)

[6] SQLite. What is SQLite? [Online]. Dostupno na: https://www.sqlite.org/index.html (17.06.2020.)

[7] Wikipedia. Apache Harmony [Online]. Dostupno na: https://en.wikipedia.org/wiki/Apache_Harmony (17.06.2020.)

[8] Learn OpenGL. OpenGL [Online]. Dostupno na: https://learnopengl.com/Getting-st*ART*ed/OpenGL (17.06.2020.)

[9] OpenSSL Cryptography and SSL /TLS Toolkit. Welcome to OpenSSL! [Online]. Dostupno na: https://www.openssl.org/ (17.06.2020.)

[10] Yadav A., Vats A., Nagpal A., Yadav A. Indian Journal of Engineering: Dalvik Virtual Machine. Discovery Journals [Elektronički časopis]. 2012. str. 1-5. Dostupno na: http://www.discoveryjournals.org/engineering/current_issue/2012/A22.pdf (14.06.2020.)

[11] Iternational Journal of Emerging Trends & Technology in Computer Science (IJETTCS). Game Development for *Android* Device using Unity 3D [Online]. Dostupno na: https://www.openssl.org/ (17.06.2020.)

[12] Zechner M., DiMarzio J.F., Green R. Beginning *Android* Games. [Online]. Third Edition. Graz: Apress; 2016 Dostupno na:

https://bbooks.info/b/w/94a85e97139c19af90a83e4e618eeac6cc0c9884/beginning-*Android*-games-3rd-edition1.pdf (20.06.2020.)

[13] The University of Queensland. Introduction to *Unity* [Online]. 2016. Dostupno na: https://www.eait.uq.edu.au/filething/get/20519/INTRODUCTION%20TO%20UNITY.pdf. (20.06.2020.)

[14] *Unity* Documentation [Online]. Dostupno na: https://docs.unity3d.com/Manual/index.html(20.06.2020.)

[15] Halvorsen H. Introduction to Visual Studio and *C#* [Online].Porsgrunn: Postboks 203. Dostupno na: https://www.halvorsen.blog/documents/tutorials/resources/Introduction%20to%20Visual%20Studio%20and%20CSharp.pdf(21.06.2020.)

[16] Liberty J. Programming *C#* [Online]. O'Reilly; 2002. Dostupno na: http://indexof.es/Programming/CSharp/O'Reilly%20%20Programming%20C%23%202nd%20Edition.pdf(21.06.2020.)

[17] American Graphics Institute. [Elektronički časopis]. 2019. Dostupno na: https://www.agitraining.com/adobe/illustrator/classes/what-is-adobe-illustrator (21.06.2020.)

[18] American Graphics Institute. What is *AdobeIllustrator* [Online]. 2019. Dostupno na: https://www.agitraining.com/adobe/illustrator/classes/what-is-adobe-illustrator(22.06.2020.)

[19] *Adobe* Photoshop 6.0 User Guide[Online]. 2000. Dostupno na: http://kfrserver.natur.cuni.cz/obecne/soubory/PhotoShop6/UserGuide.pdf (22.06.2020.)

[20] The Printing Connection. Raster Images vs. Vector Graphics. [Online]. 2019. Dostupno na: https://www.printcnx.com/resources-and-support/addiational-resources/raster-images-vs-vector-graphics/ (22.06.2020.)

[21] Državni zavod za statistiku. Klasifikacija znanstvenih i umjetničkih područja i polja [Online]. 2019. Dostupno na: https://www.dzs.hr/Hrv/important/Obrasci/08-Obrazovanje/Obrasci/Klasifikacija%20podrucja%20znanosti.pdf (22.06.2020.)

7. OZNAKE I KRATICE

AI – *AdobeIllustrator* (*Adobe* vizualni uređivač)

API – Application Programming Interface (Aplikacijsko korisničko sučelje

APK – *Android* Package (*Android* paket)

AR – Augumented Reality (Proširena stvarnost)

ART – *Android* Runtime (*Android* okruženje)

C#– C Sharp (Programski jezik)

DEX – Dalvik Executable Format (Izvršni format Dalvik)

dp – Density-Independent pixels (Broj piksela neovisnih o gustoći zaslona)

dpi – Dot Per Inch (Točka po inču)

DVM – Dalvik Virtual Machine (Virtualni stroj Dalvik)

EPS – Encapsulated PostScript

GDI – Graphics Device Interface

GPS – Global Positioning System (Globalni položajni sustav)

GUI – Graphical User Interface (Grafičko korisničko sučelje)

HAL – Hardware Abstraction Layer (Sloj hardverske apstrakcije)

IDE – Integrated Development Environment (Integrirano razvojno okruženje)

IDL – Interface Definition Language (Jezik definicije sučelja)

JDK – Java Development Kit (Java razvojni paket)

JRE – Java Runtime Environment (Java okruženje)

JVM –Java Virtual Machine (Virtualni stroj Java)

MR – Mixed Reality (Miješana stvarnost)

OpenGL – Open Graphics Library (Javna grafička biblioteka)

OpenGL ES/GLES – OpenGL for Embeded Systems (OpenGL za ugrađene sustave)

OS – Operating System (Operacijski sustav)

USB – Universal Serial Bus (Univerzalna serijska sabirnica)

PDF – Portable Document Format

SSL – Secure Sockets Layer (Sloj sigurnosnih utičnica)

STEM – ScienceTechnologyEngineeringMathematics (Znanost, tehnologija, inženjerstvo i Matematika

SVG – Scalable Vector Graphics

TLS – Transport Layer Security (Transportni sigurnosni sloj)

VM – Virtual Machine (Virtualna stroj)

VML – Vector Markup Language

VR – Virtual Reality (Virtualna stvarnost)

VS – Visual Studio (Vizualni studio)

XML – Extensible Markup Language (Jezik za označavanje podataka)

WMF – Windows Metafile

W3C – World Wide Web Consoritum

8. SAŽETAK

Naslov: Izrada prototipa *STEM* igre za *Android* platformu

Cilj ovog rada je izrada prototipa edukativne mobilne igre, namijenjene *Android* platformi. Igra je primjerena za dobno razdoblje od šest do deset godina, odnosno za učenike od prvog do četvrtog razreda osnovne škole. Sadržaj mobilne igre objedinjuje opće znanje iz znanosti, tehnologije, inženjerstva i matematike. Ideja i sam cilj igre je upoznavanje pojedinca s osnovnim pojmovima iz *STEM* područja te stvaranje interesa prema određenoj znanosti ili disciplini. Upoznavanje djece od rane dobi sa *STEM* područjima može pogodno rezultirati u budućnosti zbog traženih zanimanja. Također i bez sumnje, s razvojem novih tehnologija rast će i broj traženih zanimanja.

Praktični dio rada, odnosno, 2D igra je izrađena u razvojnom okruženju *Unity* koristeći programski jezik *C#*. Programski kod izrađen je u *Microsoft*ovom integriranom razvojnom okruženju *VisualStudio*. *UnityGame Engine*omogućuje jednostavnu realizaciju programskog koda u obliku skripti te surađuje s većinom programskihalata od kojih su najpopularniji *MonoDevelop* i *VisualStudio*. Skriptni programski jezik JavaScript moguća je alternativa programskom jeziku *C#*. Osim same funkcionalnosti, razvoj grafike i dizajna vrlo je bitan proces kod izrade igara. Lako je zaključivo da će igre s boljom grafikom privući više pažnje.

Budući da je igru predviđeno testirati od strane krajnjih korisnika izvršena je priprema za testiranje. Prva verzija edukativne *STEM* igredistribuirana je i dostupna, bez naknade, na*Google* Play trgovini.

Ključne riječi: *Unity, C#, Visual Studio*, igra.

9. ABSTRACT

Title: Prototyping a *STEM* game for an *Android* platform

The aim of this paper is to make a prototype of educational mobile games, intended for *Android* platforms. The game is suitable for children between the ages of six and ten, respectively for students from first to fourth grade of primary school. The content of the mobile games is a combination of general knowledge from science, technology, engineering and mathematics. The idea and the main goal of the game is to introduce individual and basicconcepts of the *STEM* field and to build an interest in certain sciences or disciplines.Learning *STEM* from an early age can be adequate to achieve results in future employment. Also, without a doubt, with the development of new technologies, the number of working spots will keep growing.

The practical p*ART* of the work is a 2D game that was created in *Unity* Real-Time Development Platform with usage of C #. The programming code was created in *Microsoft*'s integrated Visual Studio development environment. The *Unity* Game Engine enables easy realization of programming code in the form of scripts and is compatible with most other program tools, but the most popular are MonoDevelop and Visual Studio. The JavaScript scripting language is a possible alternative to the C # programming language. In addition to the functionality itself, the development of graphics and design is also a very important process when making games. It is easy to conclude that games with better graphics will attract more attention.

Since the game is composed to be tested by end-users, the preparation for testing has been made. The first version of the educational *STEM* game is also available, free of charge, in the *Google* Play store.

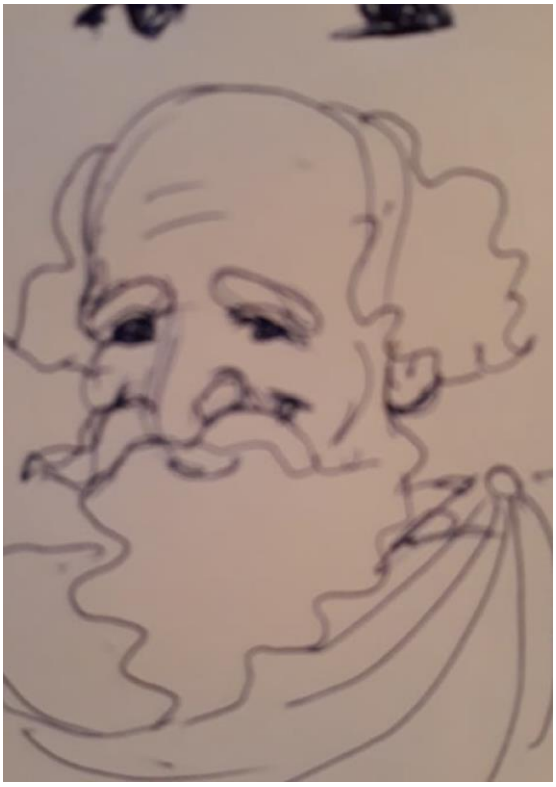
Keywords: *Unity, C#, Visual Studio,* game.

10. PRILOZI

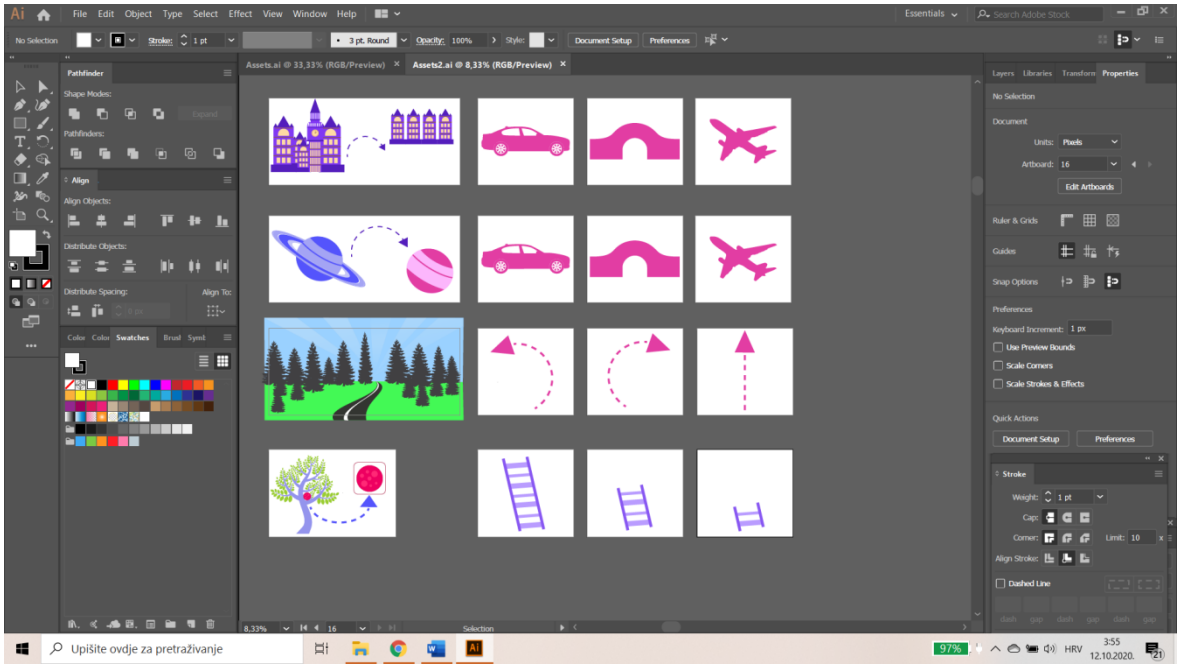
Slika 10.1: Crtež: Nikola Tesla

Slika 10.2: Crtež: Leonarda DaVinci

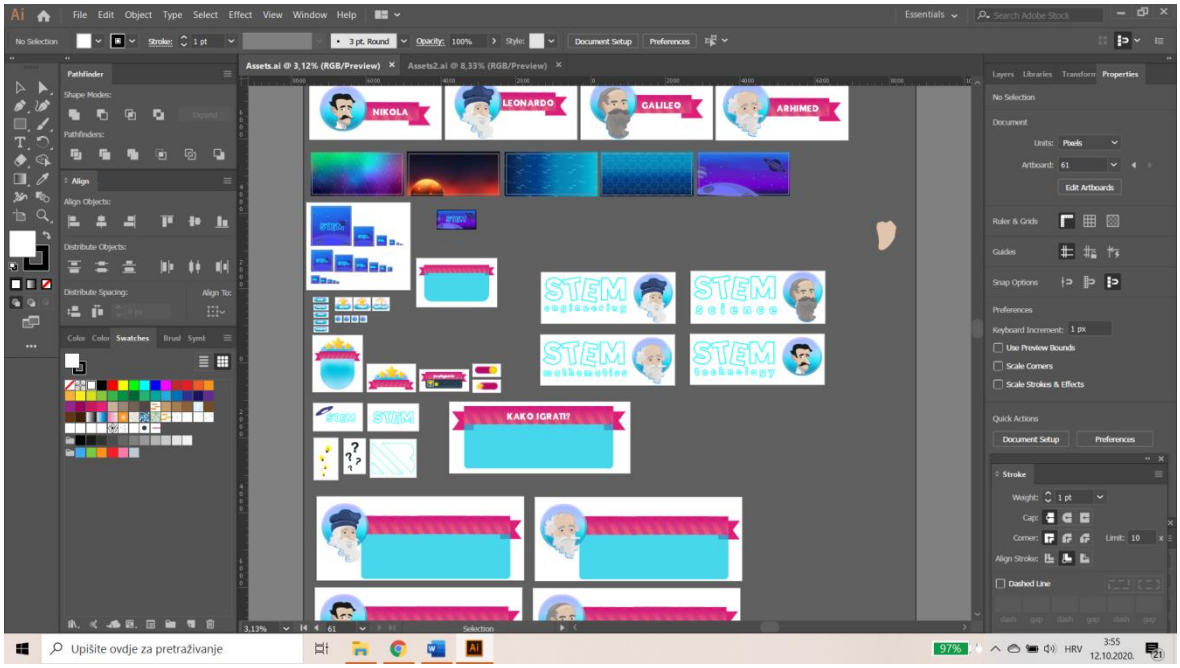
Slika 10.3: Crtež: Arhimed



Slika 10.4: Crtež: Galileo Galilei



Slika 10.5: Resursi igre



Slika 10.6: Resursi igre

Prema Odluci Veleučilišta u Bjelovaru, a u skladu sa Zakonom o znanstvenoj djelatnosti i visokom obrazovanju, elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru bit će pohranjene i javno dostupne u internetskoj bazi Nacionalne i sveučilišne knjižnice u Zagrebu. Ukoliko ste suglasni da tekst Vašeg završnog rada u cijelosti bude javno objavljen, molimo Vas da to potvrdite potpisom.

Suglasnost za objavljivanje elektroničke inačice završnog rada u javno dostupnom nacionalnom repozitoriju

NIKA BUTERIN

ime i prezime studenta/ice

Dajem suglasnost da se radi promicanja otvorenog i slobodnog pristupa znanju i informacijama cjeloviti tekst mojeg završnog rada pohrani u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu i time učini javno dostupnim.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 30.10.2020.

Buti

potpis studenta/ice

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 30.10.2020.	NIKA BUTERIN	Butin