IZRADA ANDROID MOBILNE APLIKACIJE ZA ZABAVU

Petanović, David

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

PREDIPLOMSKI STRUČNI STUDIJ ELEKTRONIKA

DAVID PETANOVIĆ

Z A V R Š N I R A D

IZRADA ANDROID MOBILNE APLIKACIJE ZA ZABAVU

Split, rujan, 2021.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

PREDIPLOMSKI STRUČNI STUDIJ ELEKTRONIKA

Predmet: Programiranje

Z A V R Š N I R A D

Kandidat: David Petanović

Naslov rada: Izrada android mobilne aplikacije za zabavu

Mentor: dipl.ing.el. Tonći Kozina, predavač

Komentor: dr.sc. Tonko Kovačević

Split, rujan, 2021.

Sadržaj

Sažetak

Izrada android mobilne aplikacije za zabavu

Ovim radom je prikazan proces izrade jednostavne aplikacije namijenjene za android mobilne uređaje kao alati i programi koji se koriste u njezinoj izradi. Objašnjeni su osnovni pojmovi svakog korištenog programa, njihovi zahtjevi, osnovni dijelovi i dr.

Glavna ideja ovog završnog rada je stvoriti funkcionalnu igru prateći osnovne faze kojih se treba pridržavati kako bi se napravio dobar program. U ovom završnom radu je napravljena nogometna igra i inačica je već postojećih igara. Ova je namijenjena hrvatskom tržištu jer uključuje sastave samo hrvatskih nogometnih klubova.

Ključne riječi: Android aplikacija, nogometna igra, programiranje

Summary

Creating android mobile application for entertainment

This paper presents the process of making a simple mobile game intended for mobiles with android platform, and tools and programs that are used in its preparation. It explains the basic concepts of each used programs, their requirements, basic parts and others.

The main idea of this paper is to create a functional game by following the basic stages that need to be followed in order to make a good program. Game made in this project is a version of the already existing games. This game is intended only for the Croatian market because it consists of Croatian clubs.

Key words: Android application, football game, programming

1 UVOD

Cilj ovog završnog rada je napraviti mobilnu aplikaciju za zabavu. Aplikacija je napravljena poštujući sve faze u izradi programa.

Faze u projektiranju i izradi koje su provedene ovim završnim radom su: definiranje zahtjeva (engl. *requirement*), oblikovanje (engl. *design*), implementacija (engl. *implementation*) i probni rad (engl. *test*). [1]

Sve ove faze su vrlo bitne prilikom razvoja projekta i moraju se poštovati ako se želi postići dobra kvaliteta proizvoda i jednostavno održavanje. Prva po redu je faza definiranja zahtjeva s obzirom na korisničke potrebe, tj. sve funkcionalnosti sustava. Faza oblikovanja definira sve korake koji se trebaju definirati prije pristupanja izrade projekta, a tiču se sustava i njegovih funkcionalnosti. Implementacija predstavlja samo pisanje programa (kodiranje) u nekom od programskih jezika (C, C++, Visual Basic i sl.) kao i izradu funkcijskih i blok dijagrama, te odgovarajuće dokumentacije koja tehnički definira sustav. Test je izuzetno bitan u razvoju projekta i potrebno je testirati sve funkcionalnosti sustava kao i kompatibilnost s ranijim verzijama ako postoje.

U nastavku završnog rada, upoznati ćemo se s korištenim programskim alatima za izradu ove android aplikacije. U 3. poglavlju obrađena je faza definiranja zahtjeva i faza oblikovanja. Zbog jednostavnosti aplikacije ove dvije faze su spojene u jedno poglavlje. U 4. poglavlju je obrađena faza implementacije gdje ćemo se upoznati s dijelovima programskog koda pomoću kojeg su postignute određene funkcije. U 5. poglavlju je obrađena faza probnog rada.

2 KORIŠTENI ALATI ZA PROGRAMIRANJE I DIZAJNIRANJE

Za izradu android aplikacije za zabavu koristili su se sljedeći alati: Unity 2019.4.3f1, Microsoft Visual Studio Code 2019 i Inkscape 1.1 za grafičku obradu slika.

2.1 Unity (Personal Edition)

Unity game engine je program za razvoj igara za PC i mobilne uređaje koji je razvijen od strane Unity Technologies. U vrlo kratkom vremenu postao je najkorišteniji program za izradu video igara. Unity je dostupan korisnicima u Osobnoj (*engl. Personal*) verziji i Profesionalnoj (*engl. Professional*) verziji. Profesionalna verzija ima više funkcija za rad s grafikom, licencu za timski rad, mogućnost pohranjivanja igre na Unity Cloud Build, itd.

2.1.1 Prednosti i nedostatci

Pored Unity alata, koji se koristio u izradi aplikacije, postoji čitav niz alata s kojima se aplikacija mogla napraviti. Jedan od najpoznatijih alata je Unreal engine, koji je po konačnom rezultatu tj. izradi aplikacija za igru, sličan Unity-u.

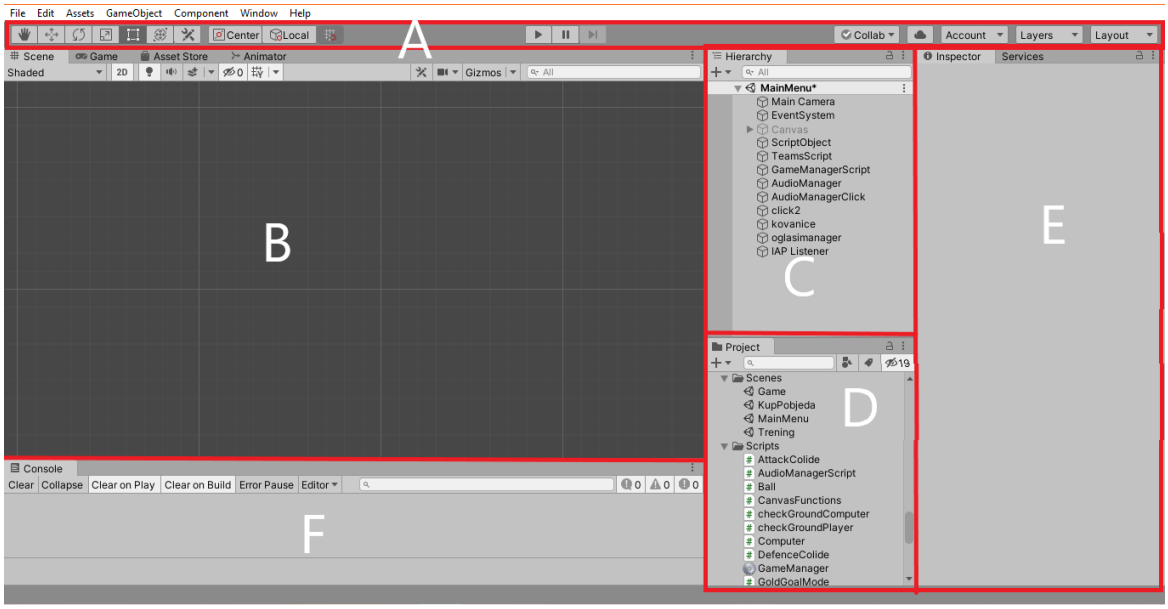
Prednosti Unity u odnosu na Unreal Engine su:

- Prikladniji alat za početnike koji žele razviti svoju igru za razne platforme: računalo, mobitel (ios, android), web, igraće konzole i dr.;
- Sadrži jako dobru implementaciju grafike, animacije kao i napredne vizualne i zvučne efekte;
- Lakša implementacija za mobilne platforme;
- Preko 70% mobilnih aplikacija za android operativni sustav napravljeno je s Unity engine.

Nedostaci su:

- Nema mogućnost vizualnog skriptiranja, gdje se s pomoću miša slažu blokovi i objekti;
- Ima slabiji prikaz i manje grafičke mogućnosti.

Korisničko sučelje[2] Unity programa sastoji se od uređivačkih prozora prikazanih na slici 1.



Slika 1 – Korisničko sučelje

(A) Alatna traka - omogućuje pristup najvažnijim radnim značajkama. Na lijevoj strani sadrži osnovne alate za manipulaciju scena i komponenti unutar programa. Služe uglavnom za mijenjanje točke središta nekog objekta i pomicanje po toj točki.

U sredini su kontrole za simulaciju aplikacije unutar Unity programa. Te kontrole su reprodukcija, pauza i pokretanje korak po korak. Ikone s desne strane omogućuju pristup Unity servisima;

(B) Pogled scene – omogućuje vizualnu navigaciju i uređivanje scene u 2D i 3D perspektivi. U pogledu scene prozor se može promijeniti i na pogled igre te se tada vidi ono što vidi i kamera[3] postavljena u igri;

(C) Hijerarhijski prozor – je hijerarhijski tekstualni prikaz svakog GameObject – a u Sceni (B). Hijerarhija prikazuje strukturu na koji se GameObject – i vežu jedan za drugim;

(D) Prozor projekta – prikazuje biblioteku objekata koji su dostupni za korištenje u projektu.

(E) Prozor inspektora – omogućuje pregled i uređivanje svih svojstava trenutno odabranog GameObject – a. Svaki GameObject ima različita svojstva koji se mogu promjeniti u ovom prozoru;

(F) Statusna traka – pruža obavijesti o Unity procesima i omogućava brzi pristup povezanim alatima i postavkama.

2.1.2 **2D/3D prikaz**

Unity program podržava izradu projekta u 2D ili 3D perspektivi kao i razne kombinacije[4] između te dvije vrste perspektiva. Neke moguće kombinacije su:

- 3D projekt,
- 3D projekt iz ptičje perspektive gdje je sve izrađeno u 3D,
- 2D projekt,
- kombinacija 2D projekta gdje je igra napravljena u 2D ali ima grafiku 3D projekta,
- projekt s 2D igranjem i grafikom ali s kamerom perspektive.

Za 2D tip projekta učitana slika se postavlja kao 2D slika (*engl. Sprite*). Za 3D tip projekta potrebno je izraditi 3D model objekta pomoću nekog drugog programa npr. Blender, Maya,... U projektu 2D tipa, „Pogled na scenu“ je postavljen u 2D perspektivi. U 3D projektu „Pogled na scenu“ je postavljen na 3D pa model možemo promatrati sa svih strana slobodnim kretanjem po prostoru scene. U 2D projektu zadani objekti nemaju izvor svjetla koje na zaslonu pravi sjenu objekta ovisno o usmjerenju svjetla.

U 3D projektu postoji glavno svjetlo koje predstavlja izvor svjetla i u odnosu na kojeg se stvaraju sjene svih objekata. Pored toga, svaki objekt može imati svoje zasebno svjetlo po kojima se dalje stvaraju sjene objekata ovisno o svjetlosnom usmjerenju.

Za ovu aplikaciju je odabran 2D tip projekta. Korištene slike za 2D su napravljene u programu za uređivanje slika Inkscape te učitane u Unity kao 2D slike.

2.1.3 **Kretanje**

Promjenom postavki objekata moguće je osigurati ubrzavanje objekta i reagiranje na sudar s drugim objektima. U našem slučaju, igrač ima: masu, brzinu kretanja, može se definirati gravitacijska sila kao i razne druge mogućnosti koje se postavljaju kroz komponentu Rigidbody2D[7].

2.1.4 **Umrežavanje**

Igra se u Unity-u može napraviti za jednog ili više igrača istovremeno. Ukoliko je igra namijenjena za više igrača, možemo iskoristiti ugrađen multiplayer MLAPI[8].

Izvan Unity-a je moguće napraviti igru za više igrača istovremeno, na način da se poveže Unity projekt s nezavisnim proizvođačem koji pruža izradu multiplayer-a. Neki od mrežnih kodova koji se mogu koristiti za izradu igre za više igrača su: DarkRift2, Foton PUN, Photon Quantum 2.0,...

Izrađena android aplikacija za zabavu namijenjena je samo za jednog igrača.

2.1.5 **Zvuk**

Korištenjem zvuka i zvučnih efekata aplikacija postaje zanimljivija i realnija. Unity ima mogućnost učitavanja, reprodukcije zvuka, stvaranja zvučnih efekta na učitane zvukove, mogućnost mijenjanja i prilagođavanja zvuka kroz skriptu, rad sa audizapisima na vremenskoj traci, upravljanje zvukom pomoću audio miksera, itd.

2.1.6 **Animacije**

Animacije služe za dodatnu vizualizaciju igre. Pomoću animacije svakom GameObject – u možemo mijenjati njegove postavke u stvarnom vremenu. Korištenjem animacija i zvučnih efekata dobivamo stvarniji doživljaj nogometne igre. U ovoj aplikaciji korištena je animacija i zvučni efekti npr. kod udaranja lopte. Prilikom udaranja lopte, pomiče se noga i reproducira se zvuk udarca. U 3D projektu animacijama možemo mijenjati svojstva i kretanje određenog GameObject – a (gravitacijska sila, masa, brzina,...), a kroz 2D projekt animiramo slike[11] ili tzv. Sprite – ove. Animacija Sprite – a može se postići učitavanjem više slika s promjenama ili promjenom postavki jedne slike u ovisnosti o vremenu (vrijeme, položaj i kut).

2.2 Microsoft Visual Studio Code 2019

Skripte predstavljaju temelj za rad aplikacije napravljene u Unity-u. Za izradu skripti se koristi programski jezik C#[17], a iste je moguće programirati pomoću programa Microsoft Visual Studio Code 2019. Microsoft Visual Studio Code je integrirano razvojno okruženje (IDE) i podržava različite programske jezike. Ugrađeni jezici su: C, C++, C#. Prilikom instalacije Microsoft Visual Studio Code je potrebno instalirati dodatak za Unity kako bi se mogao povezati s Unity programom.

Cjelokupni programski kod za ovu aplikaciju se nalazi u skriptama koje su u Prilogu završnog rada. Programskim kodom u skriptama su definirane sve funkcije glavnog izbornika, igre, demo mod-a i promjene postavki sustava.

Za skriptiranje se može koristi i Mono Develop koji je prikladniji za sporija računala dok Visual Studio zahtjeva bolje performanse računala.

2.3 Inkscape

Inkscape je profesionalni uređivač vektorskih grafika za Linux, Windows i macOs. Pomoću ovog programa nacrtane su sve slike koje se koriste u aplikaciji a to su slike za dizajn, lopte, kopačke, igrače itd.

3 DEFINIRANJE I OBLIKOVANJE MOBILNE APLIKACIJE ZA ZABAVU

Za razvoj aplikacije za zabavu poštivane su sve faze za razvoj programa. Faza definiranja zahtjeva i faza oblikovanja u ovom radu su objedinjene u jednom poglavlju radi jednostavnosti aplikacije.

3.1 Faza definiranja zahtjeva

Aplikacija za zabavu napravljena u ovom završnom radu namijenjena je za android mobilne uređaje. Igra je nogometnog tipa, ali ne uobičajeni nogomet kakav smo navikli gledati gdje oba kluba imaju po 11 igrača, već oba kluba imaju po jednog igrača koji se sastoji od glave i noge. Takav tip igre se naziva glavomet (*engl. Head soccer*). Koncept igre je 2D prikaz s 2 igrača. „Našim“ igračem upravljamo pomoću strelica i tipki za skakanje i udaranje lopte. Protivnički igrač je isprogramiran da se kreće sam u prostoru u ovisnosti o položaju lopte u igri. Potpuni izgled koncepta igre prikazan je na slici 2.

Slika 2 – Izgled igre

Modul igre se sastoji od 4 različita načina igre, a to su:

- Demo igra – služi za testiranje kontrola kretanja igrača i udaranja lopte;

- Samostalna igra – igrač može birati bilo koji klub s kojim će igrati i bilo kojeg protivnika, može mijenjati vremenske uvjete, teren i prepreke;
- Liga – igrač bira klub s kojim će igrati, a svi ostali parametri poput protivnika, vremenskih uvjeta, terena i prepreka programski su definirani. Igrač se natječe protiv 9 klubova i skuplja bodove;
- Kup – igrač bira klub s kojim će igrati, programski se raspoređuje u grupu gdje se natječe protiv 3 protivnika. Ovisno o bodovima, prelazi u sljedeću fazu, gdje se natječe do finala. Protivnici su programski generirani slučajnim odabirom.

Moguć je izbor samo hrvatskih nogometnih klubova.

Aplikacija sadrži postavke sustava koje igrač može mijenjati u bilo kojem trenutku preko glavnog izbornika ili za vrijeme igre:

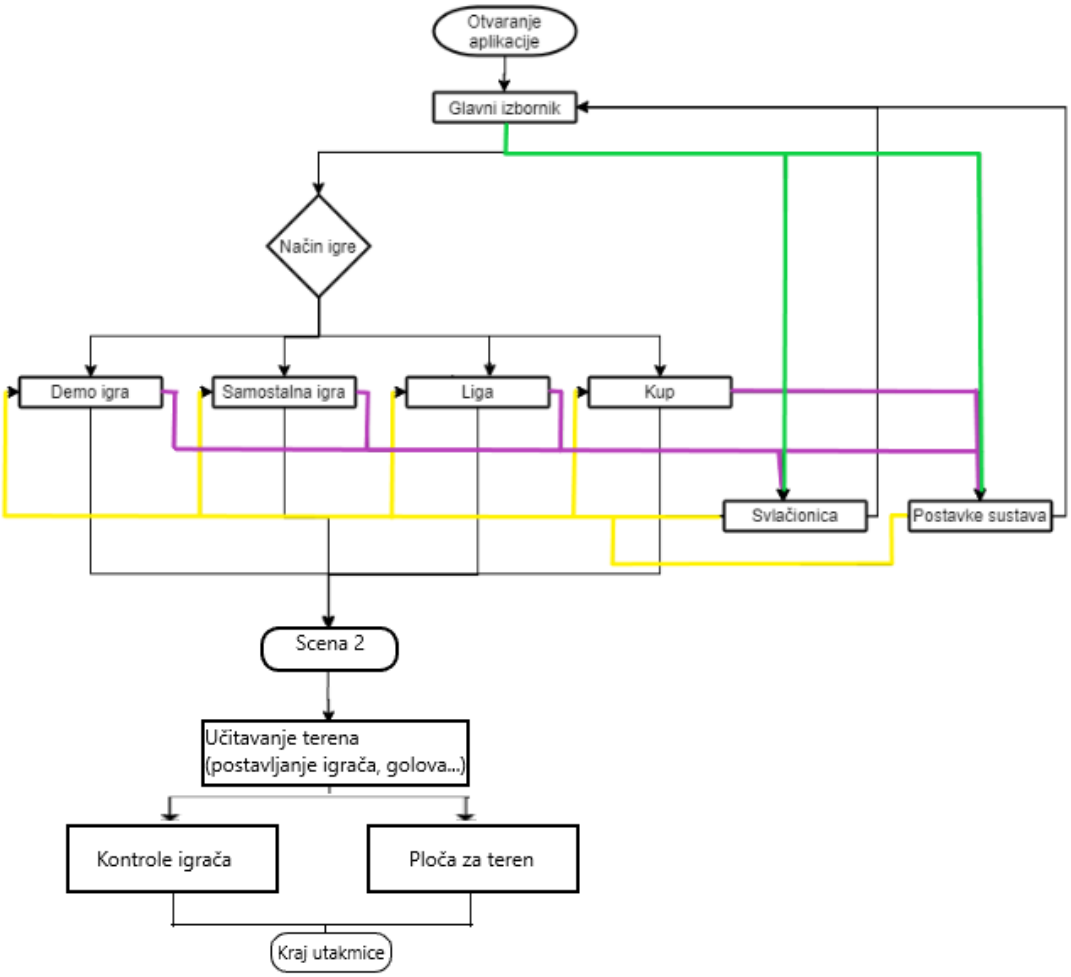
- Kontrole upravljanja igračem – tipkama ili virtualnim joystick-om;
- Jačina zvuka, uključen ili isključen zvuk;
- Vrijeme trajanja utakmice – vrijeme trajanja je ograničeno na maksimalno dvije minute;
- Strana na kojoj će se nalaziti naš igrač.

Odabirom desne strane, u svakom načinu igre, naš igrač nalaziti će se na desnoj strani, a protivnik na lijevoj strani i obrnuto.

Igra sadrži svlačionicu u kojoj možemo našem igraču mijenjati kopačke i izgled, možemo birati loptu s kojom ćemo igrati i dr. Promjene koje napravimo u svlačionici biti će iste kroz sve načine igre, a izgled protivnika i njegova kopačka mijenjaju se svaki put na početku nove utakmice slučajnim odabirom.

3.2 Faza oblikovanja

Na slici 3. dijagramom toka prikazana je međusobna povezanost glavnih funkcionalnih cjelina aplikacije. Iz dijagrama je vidljivo da se iz svakog načina igre možemo vratiti u postavke sustava ili u svlačionicu, a iz njih se možemo vraćati korak nazad ili u glavni izbornik.



Slika 3 – Dijagram toka povezanosti ploča glavnog izbornika

U glavnom izborniku bira se način igre. U „demo“ načinu igre, korisniku je omogućeno upoznavanje s tipkama za kontrolu igrača. Igrač se može pomicati lijevo ili desno i udarati loptu. S njim upravljamo pomoću strelica ili virtualnog joystick-a. Virtualni joystick se sastoji od kruga koji se može pomicati u sve strane. Ako se krug pomakne u desno i igrač će se kretati u desno i obrnuto.

Odabirom samostalne igre, igrač može: izabrati klubove koji će igrati utakmicu, izabrati teren, vremenske uvjete i prepreku po želji.

Igrač kroz igru može igrati ligu ili kup. To igru čini zabavnijom:

- Pobjedom se skupljaju bodovi;
- Ukoliko do kraja svih utakmica skupi najviše bodova, dobiva titulu prvaka;
- Igra se može nastaviti ili započeti novu sezonu;
- Igrač ima informaciju o klubu protiv kojeg će igrati sljedeću utakmicu;
- Ima uvid u tablicu i može vidjeti poredak prema broju bodova.

Cilj igre je ostvariti pozitivnu gol razliku.

Korisnik aplikacije ima mogućnost vraćanja jedan korak unazad. Npr. ako uđe u ligu, iz lige može ići u postavke i onda se može vratiti korak nazad u ligu.

U postavkama igrač može mijenjati postavke sustava: jačinu zvuka, birati stranu igranja, način igranja i vrijeme trajanja utakmice.

Nakon odabranog načina igre otvara se Scena 2 u kojoj se postavljaju sve postavke o odabranome načinu igre. Scena 2 se sastoji od jedne podloge na kojoj se nalazi ploča sa kontrolama igrača i ploča s terenom. Na terenu se nalaze 2 igrača, golovi, lopta itd. Kada vrijeme trajanja utakmice dođe do kraja otvara se ploča sa završnim rezultatom te nakon toga vraćamo se u Scenu 1, gdje ponovno možemo birati novi način igre i započeti novu utakmicu.

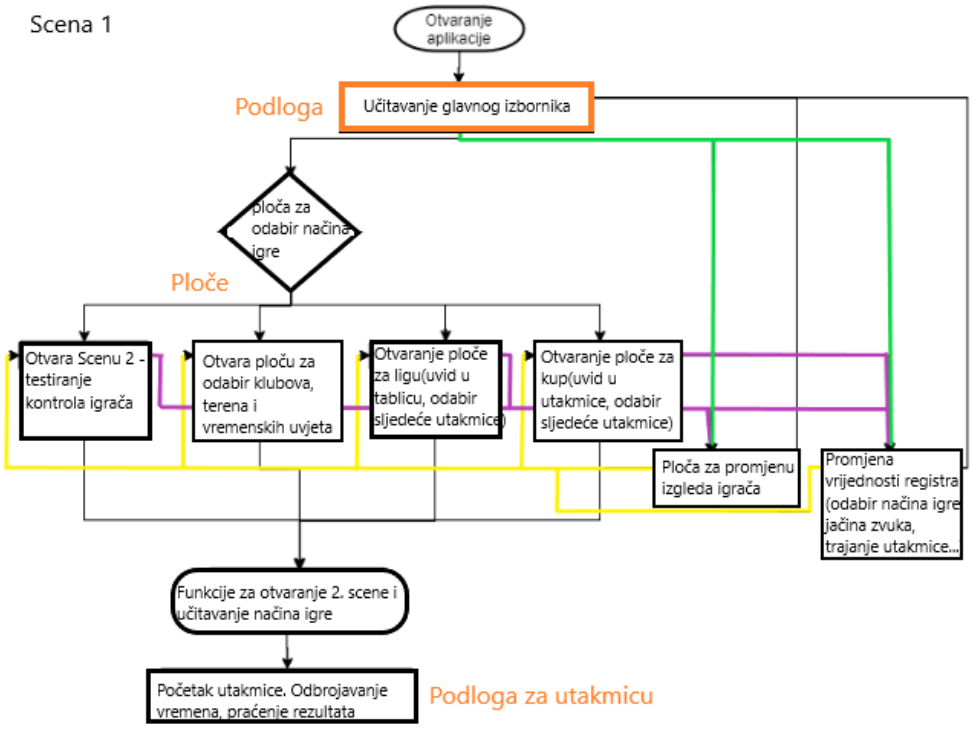
4 IMPLEMENTACIJA

U fazi implementacije vrši se kodiranje na osnovu definiranih zahtjeva i rezultata faze oblikovanja. Kodiranje se može podijeliti na vizualno, korištenjem programa Unity, i funkcionalno, korištenjem programa Microsoft Visual Studio.

4.1 Izrada vizualnog sučelja korištenjem Unity programa

Aplikacija se u Unity programu oblikuje izradom scena[13]. Scene su mjesto na kojem se radi sa sadržajem u programu. Scene sadržavaju cijelu ili dio igre/aplikacije. Na primjer, može se izraditi jednostavna igra sa jednom scenom, dok se za složeniju igru može koristiti jedna scena po razini, svaka sa svojim okruženjima, likovima, preprekama, slikama...

Aplikacija napravljena u ovom završnom radu definirana je kroz 2 scene. U prvoj sceni se nalazi glavni izbornik, a u drugoj sceni igra. Na slici 4 dijagramom toka prikazana je vizualna raspodjela funkcionalnih cjelina koje su grupirane prema scenama, podlogama (engl. *Canvas*) i pločama (engl. *Panel*).



Slika 4 – Dijagram toka vizualne raspodjele funkcionalnih cjelina

4.1.1 **Scena 1 - Glavni izbornik**

Na glavnom izborniku se nalazi način igre, svlačionica kao i postavke sustava. Glavni izbornik je oblikovan kao na slici 5.

Slika 5 – Izgled scene glavnog izbornika

Scena glavnog izbornika prikazuje se prilikom pokretanja igre. Pomoću glavnog izbornika pristupamo svim funkcionalnim dijelovima aplikacije.

Unutar scene 1 definirane su ploče (eng. *Panel*) koje su vidljive ovisno o odabranoj funkciji glavnog izbornika. Odabirom načina igre otvara se ploča na kojoj je moguće izabrati jedan od četiri načina igre. Za svaki način igre je definirana ploča koja postaje vidljiva njenim odabirom:

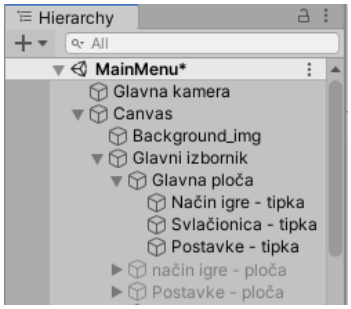
- Demo igra – ploča u kojoj možemo isprobati kontrole igrača
- Samostalna igra – ploča sadrži odabir 2 kluba, vremenske uvjete, teren i prepreke
- Liga – ploča sadržava tablicu klubova, bodove i informaciju o sljedećem protivniku
- Kup – ploča sadržava tablicu klubova, bodove i informaciju o sljedećem protivniku

Odabirom načina igre postavlja se registar i otvara se druga scena u kojoj je definirana sama igra u ovisnosti o odabranom načinu igre.

Za prikaz na zaslonu koristi se podloga[13] (*engl. Canvas*) i kamera[3]. Kamera cijelo vrijeme registrira promjene na podlozi. Sve što se nalazi na podlozi biti će vidljivo i na zaslonu. Glavni izbornik prikazan na slici 5. nalazi se na podlozi. Za mijenjanje funkcija na podlozi, koristi se ploča. Glavni izbornik se nalazi na ploči koja sadrži 3 tipke: Način igre, Svlačionica, Postavke. Pritiskom na svaku tipku poziva se određena funkcija. Pritiskom na tipku „Način igre“ zatvara

se trenutna ploča i aktivira se ploča koja sadrži 4 tipke za odabir jednog od 4 načina igre. Isto tako pritiskom na tipku „Postavke“ aktivirati će se ploča koja sadrži postavke sustava.

Radi prikaza svih elemenata prilikom aktivacije ploče, potrebno je sve elemente te ploče složiti hijerarhijski. Ploče aktiviramo na način da ih postavljamo aktivnima ili neaktivnima pomoću funkcije SetActive(). Ukoliko ploču postavimo neaktivnom tada će se sakriti i svi elementi na toj ploči. Na slici 6. možemo vidjeti hijerarhijsku strukturu ploča.



Slika 6 – Hijerarhijska struktura elemenata

Glavna kamera prikazuje sve aktivne elemente na podlozi.

4.1.2 **Scena 2 – Igra**

Druga scena služi za igru i prikazana je na slici 2. Sastoji se od 3 podloge:

- Prva podloga sadrži: loptu, 2 gola, 2 igrača i pozadinsku sliku stadiona;
- Druga podloga sadrži: kontrole za upravljanje igračem, informacije o rezultatu, trenutnom vremenu i tipku za pauzu;
- Treća podloga sadrži: informacije o završnom rezultatu i tipku za povratak u glavni izbornik. Otvara se na kraju utakmice.

4.2 Izrada skripti korištenjem programa Microsoft Visual Studio

Pomoću Microsoft Visual Studio-a se izrađuju skripte. Unutar skripti nalaze se funkcije koje su napisane korištenjem C# programskog jezika. Funkcije predstavljaju dio programskog koda koji izvršava neku radnju. Funkcije napisane unutar skripti mogu se pozivati na način da se dodjeljuju određenim radnjama, npr. prititisak na tipku ili specifičnu radnju, kao što je

odbrojavanje preostalog vremena utakmice. Funkcija za odbrojavanje poziva se svake sekunde. Kada odbrojavanje vremena dođe do nule, poziva se funkcija za kraj utakmice.

Svaki od objekata ima svojstva (engl. properties), događaje (engl. events) i metode (engl. methods). Svojstva objekta definiraju obilježja kao što su veličina, boja, pozicija objekta, ... Svaki objekt podržava određene događaje. Ukoliko je određenom događaju dodjeljena funkcija, ista će se pozvati kada se taj događaj ostvari. Npr. događaji su OnClick, OnKeyUp, OnKeyDown, ... Funkcija dodjeljena OnClick događaju pokreće se jednim pritiskom lijeve tipke miša kada se pokazivač miša nalazi na objektu ili jednim dodirom objekta na touch screen-u. Metode objekta predstavljaju funkcije pomoću kojih pristupamo i mijenjamo svojstva objekta. Npr. metoda SetActive čini objekt aktivnim ili neaktivnim.

4.2.1 Glavni izbornik

Funkcije dodjeljujemo objektu na način da u inspektoru objekta u polje *OnClick()* upišemo naziv funkcije koju će on pozvati iz programa. Pritiskom na tipku 'Način igre' pozvati će se funkcija koja se zove *OpenGameType()*. Unutar programa, ploče glavnog izbornika, način igre, svlačionica i postavke sustava, su deklarirane kao public kako bi im mogli pristupati iz različitih funkcija i kako bi ih preko opcije DragNDrop povezali s objektima u Unity-u.

```
public GameObject MenuContainer;
public GameObject Gametype;
public GameObject OptionsContainer;
public GameObject ShopContainer;
```

Nakon deklaracije i povezivanja možemo napisati funkcije koje koriste te objekte. Dan je primjer za funkciju OpenGameType().

```
public void OpenGameType()
{
    MenuContainer.SetActive(false);
    OptionsContainer.SetActive(false);
    Gametype.SetActive(true);
    ShopContainer.SetActive(false);
}
```

Pomoću naredbe *SetActive(bool)* postavlja se određena ploča aktivnom ili neaktivnom. Parametar kojim se poziva funkcija SetActive je bool (logički) tip podatka s mogućim vrijednostima *true* ili *false*. Pozivom funkcije SetActive s parametrom *true*, aktivirati će tu ploču. Na isti način su napisane i ostale funkcije za aktiviranje ploča pritiskom na tipke u

glavnom izborniku. Npr. funkcija *OpenOptionsContainer()* aktivira ploču s elementima za postavke, a sve ostale ploče postavlja neaktivnima.

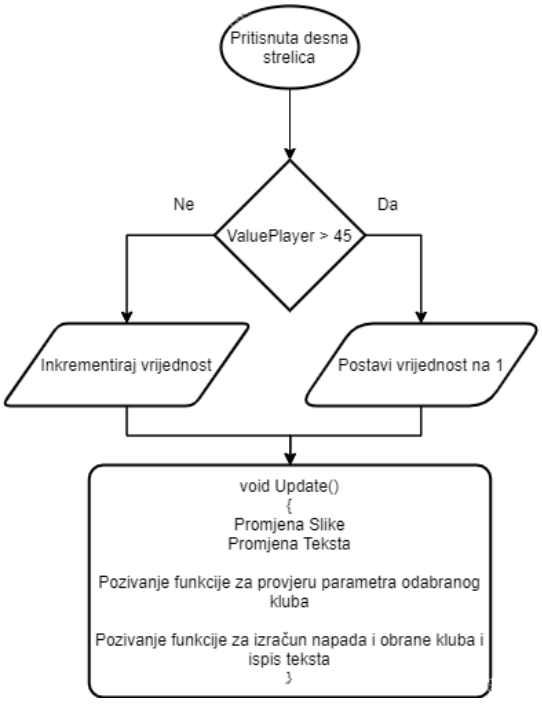
Prilikom pokretanja aplikacije pokreću se dvije funkcije:

- *void Start()* – poziva se samo jedanput i to pri otvaranju aplikacije
- *void Update()* – poziva se za svaki okvir (*engl. frame*). Za 60 fps (*engl. Frames per second*) funkcija će se pozvati 60 puta u sekundi

Unutar funkcije *Start()* se poziva funkcija koja aktivira ploču glavnog izbornika, a sve ostale ploče deaktivira.

4.2.2 **Samostalna igra**

Samostalna igra je način igre u kojoj se odabire željeni klub, protivnika, teren na kojem će se igrati, vremenske uvjete i prepreke. U skripti 'Teams' su spremljena polja sa slikama i nazivima klubova. Svi parametri se mjenjaju pritiskom na tipke. Pritiskom na desnu tipku (strelica desno), vrijednost tog parametra se povećava, a pritiskom na lijevu tipku (strelica lijevo) vrijednost parametra se smanjuje. Na slici 7. prikazan je dijagrama toka koje funkcije se redoslijedom izvršavaju pri promjeni kluba pritiskom na desnu strelicu.



Slika 7 – Dijagram toka promjene odabira kluba

Pritiskom na desnu strelicu poziva se funkcija *ButtonRightPlayer()* kojom se provjerava koja je trenutna vrijednost kluba i je li ona veća od 45 tj. je li veća od maksimalnog broja klubova, Ako je veća od 45, onda se njezina vrijednost postavlja na prvi klub u nizu, a ako nije onda se ta vrijednost povećava za jedan. Nakon toga se poziva funkcija *Update()* koja mijenja sliku i naziv kluba. Funkcija *CheckSettings()* provjerava parametre kluba. Parametri svakog kluba su:

- brzina kretanja,
- brzina kretanja prema nazad,
- jačina udarca,
- visina skoka.

„Jači“ klub ima veću brzinu, a slabiji manju, itd. Ovisno o vrijednostima parametara ispisuje se koliko je klub jak u napadu i obrani, a to se računa pomoću funkcije *Calculate()*. Ove funkcije se nalaze unutar skripte „Single Match“.

4.2.3 **Liga**

Za razliku od samostalne igre, za igru u ligi koriste se funkcije za razvrstavanje tablice i funkcije za odabir protivnika u sljedećoj utakmici.

Nakon odabira kluba i lige poziva se funkcija *SetupTable10()*. Unutar te funkcije formira se lista u koju se upisuju brojevi od 0 do 9. Korištenjem for petlje i naredbe *Random.Range(int, int)* klubovi se razvrstavaju u tablicu slučajnim odabirom. U svakom koraku for petlje klubovima se dodijeljuje varijabla u koju se spremaju njihovi bodovi. Te vrijednosti se dodaju koristeći naredbu *PlayerPrefs()* i spremaju se u registar aplikacije.

Nakon što smo napravili listu s popisom klubova poziva se funkcija *GetUITablica()* koja će ispisati naziv i sliku kluba. Unutar ove funkcije pozivaju se još 2 funkcije, a to su:

- *GetScore()* – funkcija koja uzima podatke o bodovima klubova
- *Poredak()* – funkcija koja radi poredak klubova ovisno o broju bodova

U nastavku je prikazan programski kod za funkciju *GetUITablica():*

```
    public void GetUITablica(){
        GetScore();
        Poredak();
        GameObject Tablica = GameObject.FindGameObjectWithTag("Tablica10");
        for (int i = 0; i < 10; i++){
```

```
        Tablica.GetComponent<LigaTablica>().TeamLogoImg[i].sprite = Teams.instance.teams[PlayerPrefs.GetInt("Tablica1" + i)];
        Tablica.GetComponent<LigaTablica>().NameTxt[i].text = Teams.instance.teams[PlayerPrefs.GetInt("Tablica1" + i)].name;
        Tablica.GetComponent<LigaTablica>().TxtPoint[i].text = PlayerPrefs.GetInt("RezultatTablica1" + i).ToString();
              if (PlayerPrefs.GetInt("Tablica1" + i) == (PlayerPrefs.GetInt("playervalueleague", 1) - 1))
              {
                  Tablica.GetComponent<LigaTablica>().NameTxt[i].color = Color.blue;
                  Tablica.GetComponent<LigaTablica>().TxtPoint[i].color = Color.blue;
                  pozicijakluba = i;
                  PlayerPrefs.SetInt("pozicijakluba", pozicijakluba);
                  PlayerPrefs.SetInt("liganewgame", 1);
                  PlayerPrefs.Save();}
              else{
                  Tablica.GetComponent<LigaTablica>().NameTxt[i].color = Color.black;
                  Tablica.GetComponent<LigaTablica>().TxtPoint[i].color = Color.black;}
          }
          SetupMatchLiga();
      }
```

U for petlji se uzimaju podatci o klubovima iz registra i postavljaju se u tablicu. Iz skripte LigaTablica uzima elemente slike i teksta i postavlja im trenutne vrijednosti. Odabrani klub bit će plave boje, a ostali crne boje. Na slici 8. prikazan je izgled tablice za ligu.

Slika 8 – Tablica za ligu

Funkcijom *SetupMatchLeague()*, ovisno o redoslijedu utakmice, generira se sljedeći protivnik. Sve funkcije za ligu nalaze se unutar skripte „Liga“, a skripta „LigaTablica“ sadrži niz klubova koji igraju 1. i 2. ligu.

4.2.4 **Kup**

Kup je način igre koji je definiran kroz grupnu fazu i nokaut fazu. Klub se prvo natječe u grupnoj fazi te ako je prvi ili drugi, nastavlja natjecanje u nokaut fazi do finala.

Funkcija *GetUIGroupStage()* određuje grupu u kojoj se nalazi odabrani klub i redni broj unutar te grupe.

```
if (PlayerPrefs.GetInt("GrupaA:" + i) == (playervaluetournament - 1)){
                GroupA.GetComponent<UIGroupStage>().NameTxt[i].color = Color.green;
                GroupA.GetComponent<UIGroupStage>().TxtPoint[i].color = Color.green;
                OdabranKlubJeUGrupi = 1;
                PlayerPrefs.SetInt("OdabranKlubJeUGrupi", OdabranKlubJeUGrupi);
                PozicijaOdabranogKluba = i;
                PlayerPrefs.SetInt("PozicijaOdabranogKluba", PozicijaOdabranogKluba);
                GameManager.Instance.TournamentNewGame = 1;
                GameManager.Instance.Save();}
```

Kroz gornju blok if naredbe se provjerava je li trenutni klub jednak odabranom klubu. Ako je to točno, tekst se postavlja u zelenu boju, a u dvije varijable se sprema informacija o grupi i poziciji kluba. Prema poziciji određujemo utakmice kao što je prikazano na slici 9.

1. utakmica	2. utakmica	3. utakmica
0	0	0
1	2	3
2	1	1
3	3	2

Slika 9 – Određivanje protivnika

U grupnoj fazi igraju se tri utakmice. Prvu utakmicu igraju klubovi na poziciji pod rednim brojem 0 i 1 i klubovi pod brojem 2 i 3. Drugu utakmicu igraju klubovi pod rednim brojem 0 i 2 i klubovi pod rednim brojem 1 i 3. Za određivanje protivnika poziva se funkcija *SetupMatchGroupStage* ovisno o rednom broju utakmice. U sljedećim linijama koda određuje se protivnik.

```
int match = PlayerPrefs.GetInt("match");

        switch (match)
       {
            case 1:
              {
                    roundtxt.text = "2. utakmica:";

////////////////////// GRUPA A ////////////////////////////

                ListGroup1A[0] = PlayerPrefs.GetInt("ListaA:" + 0);
                ListGroup1B[0] = PlayerPrefs.GetInt("ListaA:" + 2);
                ListGroup1A[1] = PlayerPrefs.GetInt("ListaA:" + 1);
                ListGroup1B[1] = PlayerPrefs.GetInt("ListaA:" + 3);

                 if (OdabranKlubJeUGrupi == 1)
                 {
                      for (int i = 0; i < 2; i++)
                      {
                        if (ListGroup1A[i] == (playervaluetournament - 1))
                        {
                            PlayerPrefs.SetInt("ValueComputerTournament", ListGroup1B[i]);
                            }
```

```
                    else if (ListGroup1B[i] == (playervaluetournament - 1))
                    {
                        PlayerPrefs.SetInt("ValueComputerTournament", ListGroup1A[i]);
                        }
                }
            }
```

Naredbom *PlayerPrefs.GetInt(„match*“) dobivamo podatak o rednom broju utakmice. U niz ListGroup1A spremljeni su klubovi pod rednim brojem 0 i 1, a u ListGroup1B, klubovi pod rednim brojem 2 i 3. Ti klubovi igraju prvu utakmicu međusobno. Varijabla „OdabraniKlubJeUGrupi“ se postavlja kroz funkciju *GetUIGroupStage().*

4.2.5 **Prepoznavanje načina igre**

Druga scena aplikacije služi za igru i ona uključuje sve tipove igre. Prilikom pokretanja igre učitavaju se postavke ovisno o odabranom načinu igre:

- Samostalna igra – učitavaju se oba odabrana kluba, odabrani teren, vremenski uvjeti i prepreke;
- Liga ili kup – učitava se odabrani klub i protivnik koji je generiran programom. Slučajnim odabirom se određuju teren, vremenski uvjeti i prepreke.

Postavke se učitavaju s funkcijom *Start() koja se* nalazi u skripti „Selection“. Prelaskom s prve na drugu scenu, registar će se postaviti na određenu vrijednost:

- Liga - ključ pod nazivom „GameModeLeague“ će se postaviti na vrijednost '1', ključ za kup pod nazivom „GameModeTournament“ vrijednost '1', a ključ za samostalnu igru na vrijednost '0';
- Kup - ključ pod nazivom „GameModeTournament“ će se postaviti na vrijednost '1', a sve ostale ključeve na vrijednost '0';
- Samostalna igra - ključ za samostalnu igru će se postaviti na vrijednost '1', a sve ostale ključeve na vrijednost '0'.

Ovoranjem druge scene unutar funkcije *Start()* provjerava se koji ključ ima vrijednost 1. U sljedećim linijama koda dat je blok naredbi koje se izvršavaju ukoliko je odabran način igre liga.

```
if (PlayerPrefs.GetInt("GameModeLeague",0) == 1) //ako je odabrana liga izvrši blok naredbi
        {
            if (PlayerPrefs.GetInt("strana") == 1){
//ako smo u postavkama sustava odabrali desnu stranu
            _player.transform.position = new Vector3(7.2f, -2.143304f, 0);
            _computer.transform.position = new Vector3(-7.2f, -2.143304f, 0);
```

```
            }
            if (PlayerPrefs.GetInt("strana") == 2){
//ako smo u postavkama sustava odabrali lijevu stranu
            _player.transform.position = new Vector3(-7.2f, -2.143304f, 0);
            _computer.transform.position = new Vector3(7.2f, -2.143304f, 0);
            }
       //zatim ide odabir nausmičnog broja između 0 i 4 koji bira jednu od 4 pozadina
       //odabir nasumičnog broja između 0 i 3 za odabir kiše, snijega ili sunca
       //nasumični broj između 1 i 13 za odabir vrste prepreke između 13 prepreka
       //početak igre
```

U skripti „Selection“ nalaze se funkcije koje u ovisnosti o načinu igre postavljaju dizajn (teren, izgled igrača, lopte, vremenski uvjeti itd.). Skripte „Player“ i „Computer“ sadrže funkcije koje određuju individualna obilježja igrača (brzina, jačina skoka i jačina udarca). Te dvije skripte sadrže funkcije za upravljanje igračima. U sljedećem kodu ovisno o odabranom načinu igre se određuje protivnički klub i učitavaju njegove postavke:

```
        if (PlayerPrefs.GetInt("GameModeSingleMatch") == 1)
        {
            playervalue = PlayerPrefs.GetInt("playervalue", 1);
            Postavke();
        }

        else if (PlayerPrefs.GetInt("GameModeTournament") == 1)
        {
            playervalue = PlayerPrefs.GetInt("playervaluetournament", 1);
            Postavke();
        }

        else if (PlayerPrefs.GetInt("GameModeLeague", 0) == 1)
        {
            playervalue = PlayerPrefs.GetInt("playervalueleague", 1);
            Postavke();
        }
```

Unutar funkcije postavke mogu se vidjeti individualna obilježja klubova:

```
public void Postavke()
    {
        if (playervalue == 2)  //Dinamo
        {
            speed = 250;
            PLspeedBackward = -0.99f;
            jump = 5.35f;
            shootx = -500;
            shooty = 480;
        }
        else if (playervalue == 3 || playervalue == 4) //Osijek, Rijeka
        {
            speed = 240;
            PLspeedBackward = -0.98f;
            jump = 5.32f;
            shootx = -480;
            shooty = 450;
        }...
```

Svaki klub ima svoju jačinu, brzinu kretanja, visinu skoka i jačinu udarca. Ovisno koji klub je odabran, unutar funkcije *Postavke()* postaviti će se parametri protivnika na određene vrijednosti.

4.2.6 **Kontrole**

Igra se sastoji od 2 igrača. Jednim igračem upravljamo, a drugi igrač je protivnik kojim upravlja program. Igračem upravljamo pomoću strelica za kretanje lijevo i desno. Pritiskom na tipku za skok, igrač će skočiti. Pritiskom na tipku za udarac, igrač će udariti loptu. Unutar funkcije *void FixedUpdate()* izvršava se sljedeća naredba koda:

```
rb_player.velocity = new Vector2(Time.deltaTime * speed * horialAxis, rb_player.velocity.y);
```

Ta naredba služi za kretanje igrača po x i y osi. Funkcija *Update()* se poziva 60 puta u sekundi, a funkcija *FixedUpdate()* 120 puta u sekundi. Za kretanje igrača po x osi množi se stvarno vrijeme s brzinom igrača i sa smjerom kretanja (horialAxis) koji može imati vrijednost 0, 1 ili -1. U funkciji se cijelo vrijeme poziva naredba za kretanje igrača. Igrač će stajati ako nijedna strelica nije pritisnuta i horialAxis će imati vrijednost 0. Pritiskom na desnu strelicu poziva se funkcija *Move()* koja postavlja horialAxis na -1. Ukupni rezultat će biti negativan i u funkciji *FixedUpdate()* igrač će ići desno.

Pritiskom na tipku za skok poziva se funkcija *Jump()*. Unutar te funkcije se izvršava naredba za kretanje igrača rb_player.velocity. Za skok igrača zadržavamo x vrijednost, a za y vrijednost dodaje se neka vrijednost za pomak po y osi. Ta vrijednost je spremljena u varijabli *jump*. To je prikazano u funkciji Postavke() i ovisi o jačini kluba. U nastavku je prikazan dio koda funkcije Jump().

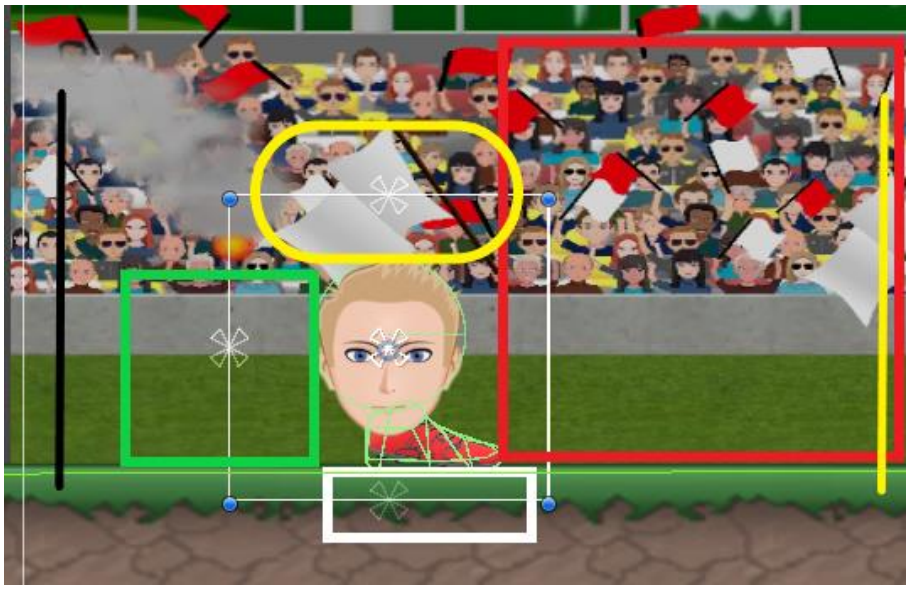
```
    public void Jump()
    {
        if (grounded == true)
        {
            rb_player.velocity = new Vector2(rb_player.velocity.x, jump);        }
        else
        {
         rb_player.velocity = new Vector2(rb_player.velocity.x, rb_player.velocity.y);
        }
```

Najprije se provjerava je li igrač na zemlji. Ukoliko je u zraku tada ne može više skakati, u protivnom dodajemo neku vrijednost na y os. Igrači imaju uključenu RigidBody2D

komponentu, stoga imaju gravitaciju. Nakon skoka vratiti će se na zemlju. a ako je igrač u zraku onda će se zadržati njegova trenutna y os.

Protivnikom upravlja skripta „Computer“, a našim igračem upravljamo mi preko tipki za pomicanje lijevo - desno, skok i udarac. Oba igrača koriste iste funkcije za kretanje.

Svaki objekt u igri ima svoje okvire (*engl. Collider*) koji su nevidljvi. Okviri predstavljaju krajnji dio objekta. Na slici 10. prikazani su okviri protivnika.



Slika 10 – Okviri protivnika u igri

Okvir na glavi protivnika žute boje omogućava da se lopta odbije od njegove glave (okvira). Zelene crtice na kopački su okvir koji služe da igrač stoji na terenu i ne propada. Teren, kao i igrač, ima svoj okvir. Ostali okviri prikazani na slici služe za kretanje igrača na način da ih oni prepoznaju samo onda kada se dodiruju s okvirom lopte. Ovisno o tome koji se okvir dodiruje s loptom, pozivati će se određene funkcije za kretanje:

- Crveni okvir: dodiruje li se lopta s crvenim okvirom, horialAxis protivnika će imati vrijednost 1 i igrač će ići desno. Dodir crvenog okvira sa žutom crtom i loptom znači da je igrač došao do našeg gola i tada će stati;
- Žuta elipsa – dodirom lopte sa žutom elipsom poziva se funkcija za skok igrača. Poziva se samo ako je igrač na zemlji tj. ako nije već u zraku;
- Zeleni okvir – dodirom lopte sa zelenim okvirom horialAxis imati će vrijednost -1 i igrač će ići lijevo.
- Bijeli okvir – dodirom bijelog okvira igrača i okvira terena postavlja se bool vrijednost da se igrač nalazi na zemlji. Ukoliko se ta dva okvira ne dodiruju igrač je u zraku i ne

može skočiti i vrijednost se postavlja na *false*. Bez ove provjere, ukoliko bi se lopta dodirnula sa žutom elipsom igrač bi skočio, lopta bi se dalje dodirivala i on bi opet skočio te bi se počeo nekontrolirano kretati po y osi. Ovom provjerom omogućeno je igraču skočiti samo jednom. Gravitacijska sila komponente RigidBody vraća ga na zemlju i tada će opet moći skočiti;

- Crna crta – služi za obranu.
- Kopačka – ukoliko se lopta dodiruje s okvirom kopačke, poziva se funkcija pomoću koje će igrač udarati loptu prema protivničkom golu.

4.2.7 **Kraj igre**

Završetkom utakmice ovisno o načinu igre smo izvršiti će se određene funkcije:

- Samostalna igra - igra završava i korisnik aplikacije ima mogućnost odigrati utakmicu ponovno s istim klubovima ili se vratiti nazad u glavni izbornik;
- Liga ili kup - ispiše se završni rezultat na ekranu i pobjednik. Igrač nema mogućnost nastavka igre već samo povratak u glavni izbornik. Povratkom u glavni izbornik otvara se odgovarajuća ploča, liga ili kup.

Kada odbrojavanje dođe do nule, iz skripte „GameControl“ će se pozvati funkcija *EndMatch()*. Zavisno o načinu igre funkcija postavlja aktivnima određene tipke. Ploča prikazana na slici 11. otvara se kad je utakmica završena.

Slika 11 – Ploča za kraj utakmice

U funkciji *EndMatch()* ispisuje se završni rezultat s nazivom i slikama klubova, pobjednik, broj golova, ukupna nagrada po golu i po pobjedi. Na vrhu se nalaze tipke kojima se možemo vratiti u glavni izbornik, ponovno igrati, produžetke i 2 tipke za dalje (za ligu i kup).

Dvije su tipke za nastavak igre. Jedna služi za povratak u ligu, a druga za povratak u kup. Ovisno koji način igre smo igrali jedna će biti vidljiva dok druga neće. Pritiskom na tipku dalje pozivaju se skripte i funkcije za bodovanje i odlučivanje ostalih utakmica tko je pobjedio, a tko ne.

5 FAZA TESTA

U razvoju projekta potrebno je testirati funkcionalnost sustava kao i kompatibilnost s ranijim verzijama ako postoje. Aplikacija napravljena u ovom završnom radu je objavljena na Google Play trgovini. Aplikacija je testirana na uređajima različitih specifikacija. Uređaji su se razlikovali s obzirom na dimenziju, brzinu rada i memoriji.

Posebna pozornost bila je usmjerena na uređaje različitih veličina ekrana i prilagodbu igre različitim veličinama ekrana.

Tijekom faze testa otklonjeni su svi uočeni nedostaci. Nakon otklanjanja nedostataka napravljeno je retestiranje svih funkcionalnosti, a prije objavljivanja igre na Google Play trgovini.

U razvoju projekta testirana je funkcionalnost igre na 10 mobilnih uređaja koji su se uglavnom razlikovali po svojim dimenzijama. Aplikacija je uspješno prošla test s obzirom na različite veličine zaslona u rasponu od 480x854 do 1920x1080 cm. Aplikacija je testirana na uređajima različitih veličina memorije i brzina procesora. Mobiteli su imali radnu memoriju od 2 do 8 GB, brzine rada procesora u rasponu od 1200 MHz do 2400 MHz. Aplikacija tijekom uporabe zauzima 25 MB memorije. Problem se javlja kod mobilnih uređaja koji imaju 1 GB radne memorije i pri procesoru manjih brzina uglavnom ispod 1200 MHz, tada je duže vrijeme pokretanja za otprilike 10 do 15 sekundi. Otvaranjem Scene 2, gdje se igra utakmica, problemi s brzinom nisu uočeni.

Među ostalim podatcima u fazi testa, značajna je i povratna informacija od korisnika koja se odnosila na dugačko vrijeme trajanja utakmice. U ovoj fazi je igra ograničena na 90 sekundi trajanja kao standardna postavka, a u postavkama je dodana mogućnost odabira trajanja utakmice između 40 i 90 sekundi.

6 ZAKLJUČAK

Korištenjem programa Unity (Personal Edition) u kombinaciji s programom Microsoft Visual Studio i drugim alatima za vizualno oblikovanje, a poštujući sve faze u projektiranju i izradi aplikacija, uspjeli smo razviti Glavomet „*Head soccer*“ android mobilnu aplikaciju za hrvatske korisnike. Ovaj tip igre uključuje dva igrača koji se sastoje od glave i noge, a koncept igre je 2D prikaz.

U ovoj aplikaciji korištena je animacija i zvučni efekti npr. kod udaranja lopte. Korištenjem zvuka i zvučnih efekata aplikacija postaje zanimljivija i realnija, a animacijama se postigla dodatna vizualizacija nogometne igre. Animacijom svakog GameObject–a moguće je mijenjati njegove postavke u stvarnom vremenu. Korištenjem animacija i zvučnih efekata dobiven je stvarniji doživljaj nogometne igre.

LITERATURA

[1] T. Kovačević: Programiranje, nastavni materijal 2012

https://moodle.oss.unist.hr/pluginfile.php/6659/mod_resource/content/2/Programiranje_C%2B%2B.pdf [01.05.2021]

[2] Unity User Manual 2020.3 (LTS), Unity's interface, https://docs.unity3d.com/Manual/UsingTheEditor.html, [01.05.2021]

[3] class in UnityEngine, Implemented in:UnityEngine.CoreModule,

https://docs.unity3d.com/ScriptReference/Camera.html, [01.05.2021]

[4] Unity User Manual 2020.3 (LTS), 2D or 3D projects

https://docs.unity3d.com/Manual/2Dor3D.html, [01.05.2021]

[5] Unity User Manual 2020.3 (LTS), 2D and 3D mode settings

https://docs.unity3d.com/Manual/2DAnd3DModeSettings.html, [01.05.2021]

[6] Unity User Manual 2020.3 (LTS), Physics

https://docs.unity3d.com/Manual/class-Physics2DManager.html, [01.05.2021]

[7] Unity User Manual 2020.3 (LTS), Rigidbody2D

https://docs.unity3d.com/Manual/class-Rigidbody2D.html, [01.05.2021]

[8] Unity User Manual 2020.3 (LTS), Multiplayer and Networking

https://docs.unity3d.com/Manual/UNet.html, [01.05.2021]

[9] Unity blog, choosing the right netcode for your game

https://blogs.unity3d.com/2020/09/08/choosing-the-right-netcode-for-your-game/?_ga=2.38711840.247458939.1617029658-1759946411.1613931865, [01.05.2021]

[10] Unity Learn, Audio

https://learn.unity.com/search?k=%5B%22q%3AAudio%22%5D, [01.05.2021]

[11] Unity Learn, Animation

https://learn.unity.com/tutorial/introduction-to-sprite-animations#, [01.05.2021]

[12] Unity Learn, Scripting

https://docs.unity3d.com/Manual/ScriptingSection.html, [01.05.2021]

[13] Unity User Manual 2020.3 (LTS), Scene https://docs.unity3d.com/Manual/CreatingScenes.html, [01.05.2021]

[14] Unity User Manual 2020.3 (LTS), Canvas

https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UICanvas.html, [01.05.2021]

[15] Unity User Manual 2020.3 (LTS), Inspector
https://docs.unity3d.com/Manual/UsingTheInspector.html, [01.05.2021]

[16] Unity User Manual 2020.3 (LTS), PlayerPrefs

https://docs.unity3d.com/ScriptReference/PlayerPrefs.html, [01.05.2021]

[17] C# Tutorial

https://www.w3schools.com/cs/index.php, [01.05.2021]

POPIS SLIKA

PRILOZI

Prilog 1 - Programski kod za izrađenu android mobilnu aplikaciju u ovom završnom radu dan je u cijelosti na CD-u koji je sastavni dio ovog završnog rada