Univerzitet u Beogradu – Elektrotehnički fakultet
Katedra za računarsku tehniku i informatiku

# Razvoj 2D video igre *Golf* u razvojnom okruženju *Unity*

diplomski rad osnovnih akademskih studija

Mentor

Dr Igor Tartalja, v.prof.

Student

Milan Gigić
2018/0372

Beograd, 2022.

# Apstrakt

Ovaj rad opisuje 2D računarsku igru „Golf“ osmišljenu kao nadogradnja laboratorijske vežbe i domaćeg zadatka predmeta Računarska grafika, ali koristeći drugačiji programski jezik i okruženje. Za razliku od domaćeg zadatka i laboratorijske vežbe, koji su bili osmišljeni da budu realizovani koristeći programski jezik *Java* i biblioteku *JavaFX*, u izradi ovog rada je korišćen programski jezik *C#* i razvojno okruženje *Unity*.

Igra „Golf“ opisana u ovom radu kao osnovni cilj ima ubacivanje loptice, ispaljujući je iz topa, u rupu na terenu. Igra se sastoji od pet nivoa, gde svaki sledeći otežava igru postavljanjem novog tipa prepreke na teren. Pobedom na svakom od pet nivoa, završava se igra sa određenim brojem poena. Deset najbolje plasiranih igrača se prikazujena rang listi. Igrač takođe ima mogućnost da stvara svoje personalizovane terene i sačuva ih, sa mogućnošću da nakon toga igra na tom terenu.

# Sadržaj

# 1. Uvod

Računarska grafika je disciplina koja se bavi generisanjem slika uz pomoć računara i interakcijom sa korisnikom. Ona ima veliku ulogu u mnogim industrijama koje koriste tehnologije nastale kao proizvod ove discipline. Razlog zbog kog je računarska grafika toliko značajna jeste sposobnost da omogući laku i jednostavnu komunikaciju između računara i korisnika. Kod nekih industrija ova disciplina predstavljala je osnovu za stvaranje same industrije, bez koje ona ne bi mogla ni da postoji. Jedna takva industrija je industrija video igara.

Prve video igre nastale su kao naučna istraživanja sa svrhom demonstriranja tehnologije, gde je uloga igre bila samo da održi interes ljudi. Nakon godina unapređivanja tehnologije započeto je i stvaranje igara sa svrhom komercijalizacije. Ovim je nastala industrija video igara koja i do danas raste i trenutno predstavlja jednu od najvećih industrija zabave na svetu. Ovaj rast industrije je takođe i podstakao rast i razvoj discipline računarske grafike napretkom hardvera potrebnog za sve veće tehničke zahteve igara.

Poboljšanjem dostupne tehnologije za pravljenje igara, programeri su postepeno dobijali veću slobodu da prave igre koje žele, bez brige o tehničkim ograničenjima. Ovo se izrazilo u rastu broja žanrova video igara, inovaciji u okviru već postojećih žanrova i kombinovanju više žanrova u okviru iste igre. Slično kombinovanje je izraženo u igri *Golf* koja je tema ovog rada. Ona za inspiraciju uzima aspekte golf igara iz sportskog žanra, ali i neke aspekte igre kao što je *Peggle* [1] iz *Puzzle* [2] žanra.

Širenjem industrije video igara povećavao se i broj ljudi zainteresovanih za razvoj video igara. Jedan od najvećih problema za ovakve ljude bila je kompleksnost procesa razvoja igre potpuno od početka bez specijalizovanih alata. To je bio razlog i osnovna motivacija za razvoj i komercijalizaciju javno dostupnih razvojnih okruženja višeg nivoa, kao što su *Unity* [3] i *Unreal Engine* [4]. Za izradu igre *Golf* korišćeno je razvojno okruženje *Unity* i programski jezik *C#* koji omogućavaju jednostavnu implementaciju standardnih vizuelnih komponenti i pozadinskih procesa potrebnih za video igre.

Razvojno okruženje *Unity* je u prethodnoj deceniji postalo prvo okruženje kojem se okreću mali timovi programera koji žele da naprave sopstvenu video igru. Iz ovog razloga je jedna od najvećih motivacija za izradu ovog rada bila učenje načina rada u *Unity* okruženju kroz proces razvoja jednostavne video igre kao što je *Golf*.

Razvoj igre *Golf* pružio je autoru šansu za učenje o načinu pisanja koda za funkcionalnosti koje se veoma često pojavljuju u video igrama, kao što su pomeranje igrača i drugih elemenata na terenu, njihove interakcije, dinamička muzika i zvučni efekti. Takođe, autor je imao poseban izazov i stekao nova iskustva kroz razvoj alata u okviru igre koji igraču pruža šansu da napravi svoju igru.

U ostatku rada detaljnije će biti opisani relevantni aspekti razvoja video igre *Golf*, tehnologija korišćena za taj razvoj, kao i funkcionalnosti , softverska arhitektura i interesantni detalji implementacije igre. Drugo poglavlje opisuje rešavani problem, navodeći definisane zahteve za stvaranje igre. Treće poglavlje opisuje funkcionalne mogućnosti igre i njen tok rada. Četvrto poglavlje opisuje način rada alata *Unity*, korišćenog za razvoj igre, kao i razlog za korišćenje datog alata. Peto poglavlje daje

detaljniji opis arhitekture softvera. Šesto poglavlje navodi i objašnjava neke od zanimljivijih detalja implementacije softvera. Ovaj rad obuhvata i Prilog A, koji sadrži uputstvo za korišćenje igre.

# 2. Problem

Ovo poglavlje opisuje konkretne probleme koji moraju da se reše pri razvijanju igre po specifičnim, unapred zadatim, zahtevima. Neki od ovih problema vezani su za korisničko iskustvo igrača, a neki za alat koji se koristi za razvoj igre i ograničenja koja on nameće.

Predmet rada je razvoj 2D video igre *Golf*. U ovoj igri igrač upravlja topom koji po zahtevu igrača ispaljuje loptice prema rupama na terenu punom raznovrsnih statičnih i pokretnih prepreka. Cilj igre je pogoditi odgovarajuću rupu na terenu.

Funkcionalni zahtevi igre zadati su kroz projektne zadatke laboratorijskih vežbi (pokazne i kontrolne), domaći zadatak (projekat) i diplomski rad na predmetu *Računarska grafika*. Ovi zahtevi opisuju tok i način funkcionisanja igre. Najznačajniji zahtevi su:

- Top koji ispaljuje lopticu određenom jačinom koja zavisi od dužine držanja levog dugmeta miša
- Više vrsta prepreka koje negativno deluju na lopticu svaka na svoj način (odbijanje, usporavanje, ubrzavanje, uklanjanje, zaustavljanje, pomeranje)
- Postavljanje žetona koji pozitivno deluju na lopticu, na nasumična mesta na terenu
- Pauziranje igre
- Jasni korisnički interfejs (eng. *interface*) koji igraču prikazuje sve potrebne informacije za trenutnu igru
- Mogućnost biranja između različitih topova i terena pri pokretanju nove igre
- Rang lista najboljih rezultata
- Meni za audio i video podešavanja igre
- Editor terena koji omogućava igraču da pravi nove terene na kojima se može igrati igra
- Čuvanje informacija o trenutnoj igri, nazivu igrača, postavljenih podešavanja i napravljenih terena u fajlu na računaru igrača

Sa ciljem stvaranja igre koja pruža zadovoljstvo korisniku potrebno je navedene funkcionalnosti implementirati na takav način da igraču pruže glatko i prijatno iskustvo. Pored funkcionalnih zahteva igre, ključno je igraču pružiti intuitivan i vizuelno atraktivan korisnički interfejs.

Jedna od osobina igre koja je veoma značajna kod modernih igara je mogućnost nadogradnje. Dodavanje novih funkcionalnosti i rešavanje problema (otklanjanje grešaka i konceptualnih nedoslednosti) kod starih je znatno lakši proces ako je prethodno ostvareno svođenje komponenti istih ili sličnih funkcionalnosti na zajedničku

osnovu čijom se izmenom menjaju sve izvedene komponente. Ovo takođe omogućava i lakše snalaženje novih osoba koje treba da započnu rad na igri.

Okruženje u kojem programer radi može veoma da olakša rešavanje nekih od navedenih problema, ali može i da stvori svoje. Pri izradi igre *Golf*, razvojno okruženje *Unity* pružilo je veoma jednostavna rešenja za problem apstrakcije komponenti igre, stvaranja jasnog korisničkog interfejsa kao i za neke od traženih funkcionalnosti, ali nije bilo bez svojih problema. Ovih problema nije bilo puno zbog jednostavne prirode igre i u većini slučajeva mogli su biti zaobiđeni koristeći komplikovaniju implementaciju, ali na pojedinim mestima su stvorili potrebu za promenom načina funkcionisanja pojedinih delova igre. O ovim problemima okruženja biće reči u poglavlju 4.

# 3. Funkcionalna specifikacija

Funkcionalna specifikacija igre čini opis njenog toka i svih mogućnosti dostupnih igraču u okviru igre. U ovom poglavlju biće opisana funkcionalna specifikacija igre *Golf* na kompaktniji način, dok je detaljni tekst dat u dokumentu **Prilog – Uputstvo za korišćenje**.

Opisana igra može se podeliti na nekoliko većih celina:

1. Glavni meni, kojim se pristupa svim drugim delovima igre
2. Sporedni meniji, koji obuhvataju menije za podešavanja, rang listu, uputstva, kontrole i informacije o igri i autoru igre
3. Igra, koja obuhvata meni za biranje topa i terena, i samu igru
4. Editor terena, koji obuhvata meni za biranje i brisanje terena, i alate za kreiranje, modifikaciju i snimanje terena.

Glavni meni predstavlja glavni način navigacije igrača kroz igru, kao i prva stvar koju igrač vidi pri pokretanju igre. On omogućava pokretanje nove igre uz izbor terena i topa za igru, nastavak prethodno pokrenute igre, izbor imena igrača, prikaz rang liste najboljih igrača, pristup video i audio podešavanjima, prikaze komandi, uputstva za korišćenje i osnovnih informacija o programu, pokretanje editora i izlazak iz igre.

Cilj ove video igre jeste da igrač prođe kroz sve nivoe bez gubitka svih svojih života i osvoji što više poena, pre nego što istekne određeno vreme. Igrač može da pređe jedan nivo tako što, koristeći top koji kontroliše, ispali lopticu i pogodi jednu od datih rupa na terenu (Slika 3.1). Ako igrač izgubi sve svoje živote, ili istekne dato vreme, igra se završava.


Slika 3.1 – Prikaz igre

Igrač kontroliše top koji se nalazi na dnu ekrana. Postoje tri topa između kojih igrač može da bira pri pokretanju nove igre, gde svaki ima različito maksimalno vreme punjenja topa i različitu maksimalnu brzinu ispaljivanja loptice. Igrač kontroliše rotaciju topa tako da je vrh topa uvek uperen ka poziciji nevidljivog kursora vođenog mišem. Na taj način igraču je pri ciljanju onemogućeno da vidi da li je kursor postavljen na rupu koju gađa. Top se može puniti do promenljivog nivoa od kojeg zavisi početna brzina kretanja ispaljene loptice. Igra može u svakom trenutku da se pauzira (zaustavlja se vreme koje se inače smanjuje, trenutno zaustavlja kretanje loptice i prikazuje meni za pauzu).

Na ekranu u toku igre na nekom nivou su u svakom trenutku vidljivi broj života, broj poena, trenutni nivo i preostalo vreme. Broj života predstavljen je srcima čiji broj odgovara broju preostalih života. Preostalo vreme je prikazano numerički, ali i linijom koja se postepeno smanjuje. Jedan deo interfejsa koji nije vidljiv u svakom trenutku jeste indikator brzine ispaljivanja loptice. On se pojavljuje samo u trenucima u kojim igrač puni top. Ovaj indikator predstavljen je pravougaonikom koji je orijentisan u smeru ispaljivanja loptice, čija dužina zavisi od vremena punjenja topa.

Svaki nivo igre se igra na jednom od postojećih terena. Tereni su ograđeni ogradom koja ograničava kretanje loptice tako da ona ne može da izađe van terena, što postiže elastičnim sudarom sa ogradom. Osim ove ograde, na terenu se nalazi nekoliko vrsta komponenata koje deluju na kretanje loptice (Slika 3.2). Ove komponente obuhvataju prepreke raznih veličina koje elastično odbijaju lopticu, polja blata i leda koja utiču na brzinu loptice, polja za teleportaciju loptice, ptice koje kradu lopticu pri kontaktu sa njom i jezera u kojima loptica tone.

Slika 3.2 – Prikaz komponenti terena

Na terenu se u nasumično vreme na nasumičnom mestu, tako da se ne preklapa sa određenim drugim komponentama, može pojaviti žeton. Postoji nekoliko vrsta žetona i sve pozitivno deluju na lopticu, odnosno igrača (Slika 3.3). Zlatni žetoni donose poene, zeleni uklanjaju uticaj blata i jezera na lopticu, ljubičasti vraćaju deo vremena koje je isteklo i crveni vraćaju izgubljene živote.

Slika 3.3 – Vrste žetona

Igrač tokom igre skuplja poene pogađanjem rupa i zlatnih žetona na terenu. Rupe daju različit broj poena u zavisnosti od njihove boje, koja predstavlja težinu pogađanja rupe. Ove boje su žuta, narandžasta i crvena, gde je žuta najlakša a crvena najteža za pogađanje. Pogađanjem rupe igrač prelazi na naredni nivo, osim ako je već na poslednjem, gde se igra završava i izračunava se konačni rezultat. Ovaj rezultat se upisuje u rang listu ako spada u deset najboljih rezultata i ako igrač nije već ranije imao bolji rezultat.

Jedna igra podeljena je na pet nivoa, gde se na svakom nivou na teren dodaje nova vrsta komponente sa ciljem otežavanja igre. Prvi nivo dodaje prepreke koje odbijaju lopticu, drugi nivo dodaje polja blata i leda, treći nivo dodaje polja za teleportaciju, četvrti nivo dodaje ptice i peti nivo dodaje jezera.

Pored ugrađenih terena koji dolaze uz igru, igrač ima mogućnost da napravi sopstveni teren. Ovo postiže koristeći editor terena koji pruža alate za uređivanje terena upisom željenog broja žetona i dodavanjem komponenti dostupnih u paleti na levoj strani ekrana (Slika 3.4). Ove komponente mogu se prevući na teren na desnoj strani ekrana, ili se mogu izabrati i koristiti kao pečat omogućavajući postavljanje više istih komponenti za redom. Postavljene komponente mogu biti selektovane, što omogućava njihovo pomeranje na drugo mesto ili njihovo brisanje. Napravljen teren se može sačuvati pod postojećim imenom ili kao novi teren sa novim imenom, ali samo pod uslovom da na terenu postoji rupa i da je upisano ime terena jedinstveno. Sačuvane informacije sadrže naziv terena, broj željenih novčića i pozicije svih postavljenih komponenti.


Slika 3.4 – Editor terena

# 4. Alati za razvoj

Razvoj igre *Golf* odrađen je koristeći razvojno okruženje *Unity* i programski jezik koji ovo okruženje podržava, *C#*. U ovom poglavlju detaljnije će biti opisano okruženje *Unity*, njegova istorija, mogućnosti i struktura.

Okruženje *Unity* prvo je nastalo kao pogon igre (eng. *game engine*) čiji su ciljni korisnici bili programeri za *Mac OS* operativni sistem. Vremenom se postepeno proširilo da pruži podršku velikom broju drugih platformi kao što su *Windows*, *Linux*, *Android*, *IOS*, *Playstation* i druge. Zbog ove mogućnosti podržavanja velikog broja platformi i činjenice da je jednostavno za početnike, okruženje *Unity* postalo je jedno od najpopularnijih pogona igre za početnike i male timove programera [5].

Okruženje *Unity* ima jasan korisnički interfejs bez nepotrebnih informacija, gde su najvažniji alati i informacije uvek dostupni (Slika 4.1). Ovaj korisnički interfejs može se podeliti na nekoliko delova [6]:


Slika 4.1 – Razvojno okruženje *Unity*

A. Traka sa alatima (eng. *Toolbar*) koja pruža pristup *Unity* nalogu, *Unity Cloud* servisima, kontrolama za pokretanje, pauziranje i zaustavljanje programa, meniju za vidljivost slojeva i meniju za raspored editora.

B. Prozor hijerarhije (eng. *Hierarchy window*) koji predstavlja hijerarhiju svih *GameObject* entiteta u Sceni (eng. *Scene*).

C. Prikaz igre (eng. *Game View*) simulira izgled konačno iscrtane (eng. *rendered*) igre kroz perspektivu kamere dodeljene sceni.

D. Prikaz scene (eng. *Scene View)* omogućava vizuelnu navigaciju i izmenu scene. Ovim može da se prikaže 2D ili 3D perspektiva scene.

E. Prekrivači (eng. *Overlays*) koji sadrže osnovne alate za manipulisanje prikazom scene i *GameObject* entiteta u njoj.

F. Inspektorski prozor (eng. *Inspector Window*) dopušta pregled svih osobina trenutno selektovanog *GameObject* entiteta. Zbog različitih vrsta *GameObject* predmeta, sadržaj i raspored inspektora tih predmeta se takođe razlikuje.

G. Prozor projekta (eng. *Project Window*) prikazuje biblioteku svih sredstava (eng. *asset*) koji su dostupni u projektu. Pri uvozu sredstava u projekat, ona se pojavljuju u ovom prozoru.

H. Traka za status (eng. *Status Bar*) prikazuje obaveštenja o velikom broju procesa okruženja *Unity* i brzi pristup alatima i podešavanjima vezanim za ta obaveštenja.

Standardni projekat u okruženju *Unity* sastoji se od skupa scena koje se mogu zamenjivati, što se postiže kroz kod funkcijom *LoadScene* biblioteke *UnityEngine.SceneManager*. Jedna scena se sastoji od skupa entiteta *GameObject* koji čine neku hijerarhiji, gde je *GameObject* osnovna klasa za sve entitete u scenama. Jedan takav entitet može da bude dete (eng. *child*) ili roditelj (eng. *parent*) drugom entitetu. Svi entiteti *GameObject* sadrže neke osobine. Ove osobine nazivaju se komponente (eng. *Component*) i svaki entitet ima osnovnu komponentu koja se sastoji od njegovog naziva, dodeljene oznake (eng. *tag*) i sloja (eng. *layer*) kojem pripada (Slika 4.2). Neke od značajnijih komponenti koje su korišćene u ovom radu su:

- *Transform* – zadaje poziciju, veličinu i rotaciju entiteta u koordinatnom sistemu roditelja

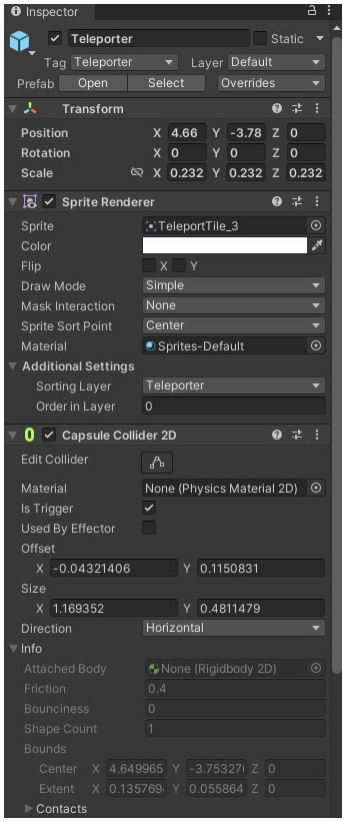
- *Rect Transform* – zadaje poziciju, veličinu i rotaciju entiteta u odnosu na roditeljski entitet, sa opcijom da ove osobine budu iskazane relativno u odnosu na osobine roditelja

- *Sprite Renderer* – prikazuje sprajt (eng. *sprite*) na poziciji entiteta, sa opcijom postavljanja boje, sloja i načina prikaza.

- *Collider 2D* – omogućava detekciju sudara između drugih entiteta i ove komponente u 2D prostoru. Postoji više vrsta ove komponente koje se razlikuju po obliku. Postoje *Box, Capsule, Circle, Composite, Custom, Edge, Polygon* i *Tilemap* 2D sudarači (eng. *collider*).

- *Rigidbody 2D* – dozvoljava primenu fizike na entitete dodeljujući osobine kao što je masa, trenje, jačina gravitacije i vrsta materijala.

- *Script* – omogućava direktno pisanje koda proizvoljne funkcionalnosti entitetu kome je




Slika 4.2 – Komponente *GameObject* entiteta

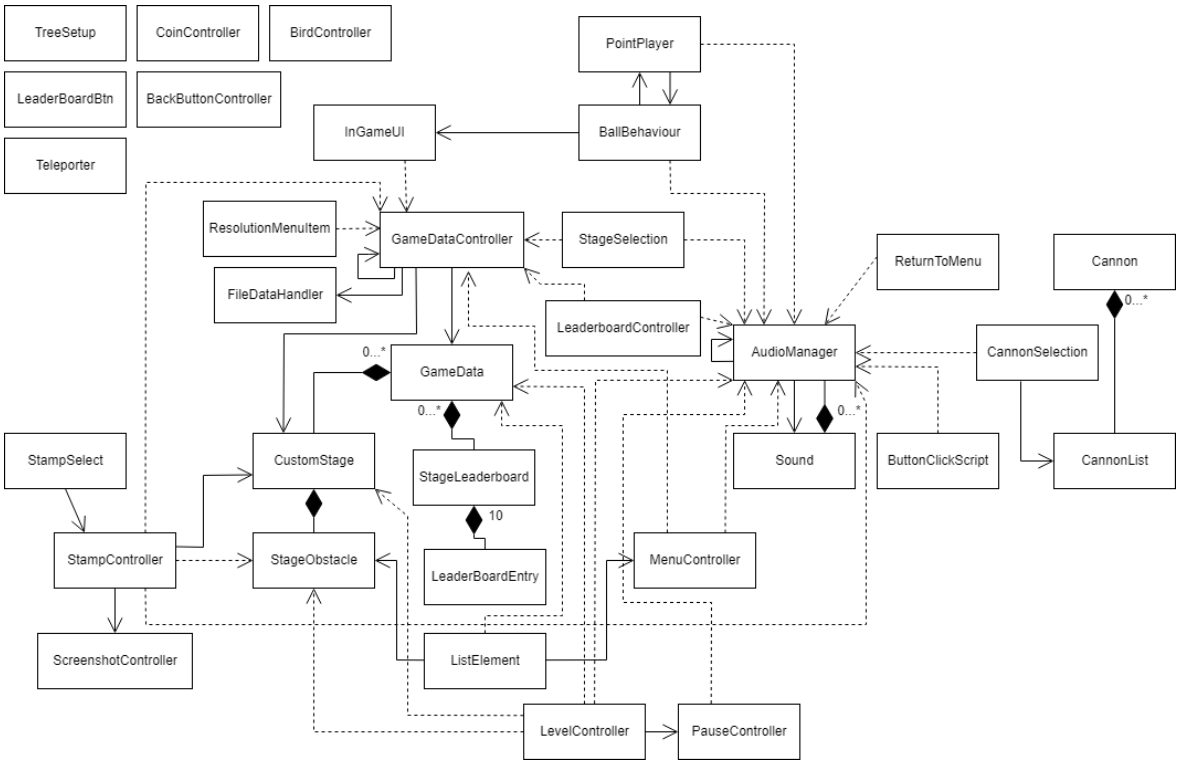
dodeljen. Ove skripte uvek su stvorene kao klase izvedene iz klase *MonoBehaviour* koja implementira fukcije *Start*, koja se poziva pri inicijalizaciji komponente, i *Update*, koja se poziva nakon svakog frejma (eng. *frame*). Ovo izvođenje ne mora da postoji i može se ukloniti po želji programera. Jedan entitet može imati proizvoljno puno skripti.

Jedna od funkcionalnosti *Unity* okruženja koja olakšava rad programera jeste postojanje šablona (eng. *prefab*) koji dozvoljavaju razvoj na visokom stepenu apstrakcije. Pravljenje i korišćenje šablona omogućava proizvoljno postavljanje primeraka datog šablona u scenu i omogućava laku izmenu svih primeraka (klonova) menjanjem samo jedne. Ova funkcionalnost bila je ključna kod implementacije postavljanja prepreka na teren u editoru terena u igri *Golf*. Još jedna funkcionalnost okruženja *Unity* koja je od velike koristi je Sistem događaja (eng. *Event System*) koji dozvoljava jednostavnu primenu slušalaca (eng. *listener*) događaja na već postojeće događaje u određenim tipovima entiteta kao što je dugme, ali i u novim koje može da implementira programer. Postojanje prozora scene takođe olakšava rad programera dajući mogućnost proizvoljnog pozicioniranja elemenata i pregleda izgleda scene.

Pored navedenih prednosti okruženja, *Unity* sadrži i par mana. U ove mane spadaju loš kvalitet standardnog osvetljenja, nemogućnost korišćenja eksternih biblioteka, poteškoće pri dobijanju izvornog koda i loše performanse kod kompleksnijih video igara. Nijedna od pomenutih mana nije uticala na ovaj rad zbog njegove jednostavne prirode, ali jedna mana jeste postala jasna, a to je rad sa prozorima. *Unity* sadrži veoma malo alata i funkcija za rad sa prozorom aplikacije zbog nestandardizovane prirode prozora kroz sve moguće platforme.

# 5. Projekat softvera

U ovom poglavlju opisuje se arhitektura softvera konceptualnim dijagramom klasa (Slika 5.1) i opisom najznačajnijih klasa u projektu.




Slika 5.1 – Konceptualni dijagram klasa

Ovaj projekat sadrži veliki broj klasa, gde su neke izvedene iz klase *MonoBehaviour* i koriste se kao komponente pridružene entitetima u sceni, dok su druge pomoćne i služe kao strukture podataka. Najznačajnija takva klasa je *GameData* koja sadrži sve sačuvane informacije o igri, gde su neke od tih informacija u formi primeraka klasa *LeaderBoardEntry, StageLeaderboard*, *CustomStage* i *StageObstacle*. Klasa koja sadrži referencu na primerak klase *GameData je GameDataController* i ona je *Unikat* (eng. *singleton*) klasa, čiji primerak se prvi put stvara pri pokretanju igre. Druga ovakva klasa u projektu je *AudioManager* koja sadrži reference na sve audio fajlove kroz klasu *Sound* i nju zovu sve klase koje žele da reprodukuju audio. Neke od najvažnijih klasa su *LevelController* i *StampController* koje su zadužene da učitavaju sve elemente terena pri otvaranju tog terena u igri, odnosno u editoru. Klasa *StampSelect* klasi *StampController* šalje informacije o prepreci izabranoj iz palete, koja dalje koristi tu informaciju i obrađuje ulazne komande igrača. Pri čuvanju terena, zove se funkcija klase *ScreenshotController* da slika trenutni izgled terena. Podaci sačuvani u okviru klase *GameData* pri izlasku iz aplikacije upisuju se u eksterni fajl koristeći primerak klase *FileDataHandler*.

Pri otvaranju glavnog menija, stvara se primerak klase *MenuController* koji obrađuje interakciju sa većinom menija koji se otvaraju iz glavnog. Neki meniji koriste dodatne klase za učitavanje svog sadržaja i to su *LeaderboardController* za meni rang

liste, *ResolutionMenuItem* za meni podešavanja, *ListElement* za meni editora terena i *CannonSelection* i *StageSelection* za izbor topa i terena, respektivno, u meniju za novu igru.

Pri pokretanju igre iz glavnog menija ili iz menija za novu igru, prelazi se u scenu za igru gde se pomenuta klasa *LevelController* bavi postavljanjem prepreka. Klasa *BallBehaviour* bavi se nadgledanjem kolizija loptice i njenog kretanja. Ovo kretanje se pokreće iz klase *PointPlayer* koja se bavi ciljanjem i pucanjem iz topa, nakon čega šalje klasi *BallBehaviour* informacije o brzini loptice. Informacije na korisničkom interfejsu prate se u klasi *InGameUI*, dok obradu pauziranja igre vrši klasa *PauseController*.

# 6. Implementacija softvera

U ovom poglavlju opisani su određeni zanimljivi elementi implementacije i tehničke karakteristike softvera.

## 6.1. Stvaranje korisničkih terena

Igrač koristeći editor terena može da stvori teren sa proizvoljnim brojem prepreka koje može da izabere iz ponuđene palete, podrazumevajući da su prepreke na validnim mestima. Editor omogućava dva načina postavljanja prepreka na teren, i to prevlačenjem, da bi se postavila samo jedna, i izborom, da bi se moglo postaviti više klonova izabrane prepreke za redom.

Paleta u editoru sadrži listu komponenti čijim se izborom stvara primerak šablona odgovarajuće komponente koji prati miš igrača. Postavljanjem komponente na moguće mesto na terenu, stvara se novi primerak komponente na osnovu šablona i postavlja na teren. Svaka komponenta ima šablon koji odgovara datoj komponenti u editoru i šablon koji odgovara toj komponenti u toku igre. Ovi šabloni razlikuju se po svojim 2D sudaračima koji se u editoru koriste za biranje postavljene komponente i otkrivanje sudara sa drugim komponentama, dok se u igri koriste za otkrivanje sudara sa lopticom.

Pri postavljanju komponente na teren proverava se sudar postavljane komponente sa drugim na terenu i, u slučaju sudara, proverava se vrsta komponente označena njenim atributom oznake i posmatra njena interakcija sa vrstom komponente sa kojom se sudarila. U slučaju da je utvrđeno da ne sme da se dešava sudar, šablon primerka komponente koji prati miša se zacrveni i postavljanje je onemogućeno.

## 6.2. Čuvanje podataka igre

Klasa *GameDataController* je Unikat klasa čiji se primerak stvara pri pokretanju igre i koji vrši obradu i čuvanje korisničkih podataka igre. Ona pristupa sačuvanom fajlu na računaru koji menja i čuva po potrebi. Pri prvom pokretanju igre ovaj fajl se stvara u folderu *C:\Users\Name\AppData\LocalLow\Elektrotehnicki Fakultet\Golf* pod imenom fajla „*data.save*“. U ovom fajlu sačuvani su podaci o imenu igrača, sadržaju rang liste, izabranim podešavanjima, poslednjoj odigranoj nezavršenoj igri i svim terenima napravljenim i sačuvanim u editoru terena.

Tereni napravljeni u editoru čuvaju se u formi niza objekata klase *CustomStage*. Ova klasa sadrži ime terena, broj žetona, sliku terena i listu svih komponenti koje su predstavljene klasom *StageObstacle*, koja sadrži naziv šablona komponente i poziciju u formi vektora.

Pri čuvanju terena u editoru, stvara se slika dela ekrana koji predstavlja teren koristeći klasu *ScreenShotController*. Ovaj proces izvršava se koristeći kameru odvojenu od one kroz koju igrač posmatra, kojoj se dodeljuje tekstura, nakon čega se kameri naloži da iscrta scenu. Rezultat je smešten u objekat klase *Texture2D* i koduje se u *base64* formatu, koji se čuva kao vrednost tipa *string*.

## 6.3. Tehničke karakteristike

Korisnička aplikacija postoji u formi *.exe* fajla koji se nalazi u folderu sa svim svojim pomoćnim fajlovima generisanim od strane *Unity* okruženja pri izgradnji (eng. *building)* projekta. Pokretanje aplikacije zahteva korišćenje *Windows* operativnog sistema i monitor minimalne rezolucije 1024x576 piksela.

U nastavku se nalazi tabela sa tehničkim karakteristikama projekta (Tabela 6.1).

Tabela 6.1 – Tehničke karakteristike

| | |
|---|---|
| Broj fajlova (*.cs* fajlova) | 27 |
| Broj klasa | 33 |
| Broj metoda | 164 |
| Broj linija koda | 3851 |
| Veličina aplikacije | 94.7 MB |
| Veličina izvornog projekta | 1.56 GB |

# 7. Zaključak

Proces razvoja igre *Golf* predstavljao je prvi ovakav izazov koji je autor imao sa okruženjem *Unity*. Kao posledica ovoga, veliki delovi projekta su vremenom bili izmenjeni ili potpuno zamenjeni usled sticanja iskustva i smišljanja boljeg rešenja. Sticanje iskustva je takođe uticalo i na dostizanje kvalitetnijeg i efikasnijeg način rada, koji je doveo do stvaranja video igre koja bi mogla na tehničkom nivou da se poredi sa igrama kojim je inspirisana.

U radu se opisuje razvoj igre *Golf*, u kojoj igrač gađa topom rupe na terenu, izbegavajući brojne prepreke. Aplikacija sadrži više menija kao što su glavni meni, meni za podešavanja, rang-lista, meni u pauzi, scenu igre na izabranom terenu sa izabranim topom, teren sa brojnim preprekama i ciljnim rupama, kao i editor u kojem igrač može kreirati nove terene.

Neke od većih mana igre su manjak kvalitetno dizajniranih terena, koji potiče od malog iskustva sa temom dizajna video igara i stilska konzistentnost izgleda igre prouzrokovanih korišćenjem različitih izvora prilikom nabavljanja i stvaranja slika za pozadine, prepreke i sl. Ove mane se mogu prevazići sticanjem iskustva u pomenutim oblastima ili angažovanjem dizajnera.

Određeni aspekti igre mogu se unaprediti, a mogu se i dodati novi koji bi učinili iskustvo igranja mnogo prijatnijim. Neki ovakvi dodaci su:

- Dodavanje varijacije zvučnih efekata i njihova primena sa ciljem stvaranja reaktivnijeg korisničkog interfejsa
- Dodavanje novih nivoa koji bi na teren postavili nove vrste prepreka
- Implementiranje onlajn (eng. *online*) rang liste, gde bi igrači mogli da vide rezultate najboljih igrača na internetu
- Dodavanje mogućnosti da igrač promeni tastere koje koristi za kontrolu igre u *Controls* meniju
- Dodavanje animacija za sve elemente igre kojim bi se postigao efekat realnosti. Primer ovakvih bio bi pojavljivanje dima pri ispaljivanju topa i povremeno lelujanje krošnji drveća.
- Podrška za druge platforme

# 8. Literatura

[1] *Peggle* video igra, Vikipedija stranica:
https://en.wikipedia.org/wiki/Peggle

[2] *Puzzle* video igre, Vikipedija stranica:
https://en.wikipedia.org/wiki/Puzzle_video_game

[3] *Unity* razvojno okruženje, zvanična stranica:
https://unity.com/

[4] *Unreal Engine* razvojno okruženje, zvanična stranica
https://www.unrealengine.com/en-US

[5] *Unity game engine*, Vikipedija stranica:
https://en.wikipedia.org/wiki/Unity_(game_engine)

[6] *Unity's interface*, *Unity* dokumentacija:
https://docs.unity3d.com/Manual/UsingTheEditor.html