Univerzitet u Beogradu

Matematički fakultet

Master rad

Razvoj 3d igara za sistem Android pomoću okruženja Unity

Nikola Milojević

Mentor: **dr Filip Marić**

Septembar 2017.

Mentor:

dr Filip Marić
Matematički fakultet
Univerzitet u Beogradu

Članovi komisije:

dr Miodrag Živković
Matematički fakultet
Univerzitet u Beogradu

dr Dušan Tošić
Matematički fakultet
Univerzitet u Beogradu

Datum odbrane:

Apstrakt

Tema je inspiriana velikom ekspanzijom video-igara poslednjih godina, kao i činjenicom da je razvoj video-igara jedna od najprofitabilnijih grana u sektoru informacionih tehnologija. Posebno bitna oblast u razvoju video-igara tokom prethodnih godina su mobilni uređaji. Android zauzima 87,6% [10] tržišnog udela što ga čini vodećim operativnim sistemom za mobilne uređaje. Osim zabavnog karaktera, video-igre mogu biti i edukativne i savremenu tehnologiju moguće je upotrebiti u cilju razvoja obrazovanja.

Cilj rada je da se prikaže pristup razvoju video-igara za sistem Android kroz korišćenje okruženja za razvoj video-igara. Unity je okruženje *(eng. Game engine)* koji daje celovito rešenje za razvoj video-igara, koji je ujedno i vodeće okruženje sa naslovima koji su postizali svetske uspehe i dokazali da se kvalitet igara ne razlikuje od video-igara napravljenih bez pomoći nekog okruženja za razvoj video-igara. U radu će biti prikazan razvoj edukativne video-igre *"Solarni Sistem"* sa akcentom na prikazu osnovnih entiteta i komponenti ali i nekih naprednih tehnika okruženja Unity.

Sadržaj

Glava 1

Operativni sistem Android

1.1 Uvod u Android

Android je operativni sistem otvorenog koda, zasnovan na sistemu Linux napravljen za veliki spektar mobilnih uređaja. Android je originalno razvila kompanija *Android Inc.*, od koje je 2005. godine kao deo strategije za pristupanje tržištu mobilnih uređaja kupila kompanija *Google* i preuzela odgovornost za njihov dalji razvoj.

Android je trenutno najrasprostranjeniji operativni sistem za mobilne telefone, prilagođen je tako da se može koristiti na većini mobilnih uređaja, uključujući pored mobilnih telefona, i tablet računare, laptop računare, netbook računare, čitače elektronskih knjiga i ručne satove.

1.2 Arhitektura sistema Android

Najbolji način razumevanja sistema Android je prikaz njegovog rasčlanjenja na sekcije. Android je grubo podeljen na šest sekcija i pet slojeva:

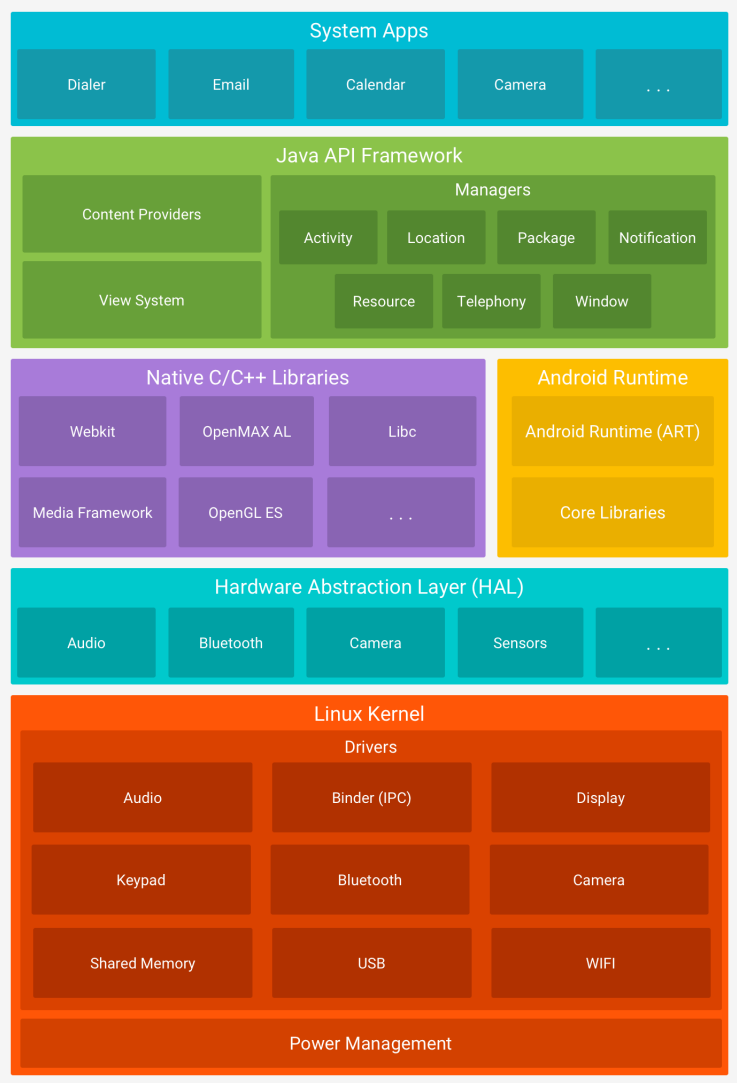
- **Linux jezgro** - osnova sistema Android. Ovaj sloj sadrži sve drajvere uređaja definisane na niskom nivou za različite hardverske komponente svakog pojedinačnog Android uređaja.
- **Apstraktni sloj hardvera (eng. HAL)** - definiše standardni interfejs koji omogućava korišćenje hardverskih komponenti uređaja na višem nivou pomoću Java API[1] okvira. Apstraktni sloj hardvera se sastoji iz više modula, gde svaki modul omogućava interfejs za specifičnu hardversku komponentu,kao na primer modul kamere ili BlueTooth-a. Pri pozivu pristupa se hardverskoj komponenti iz API okvira, sistem Android automatski učitava modul te komponente.
- **Biblioteke** - sadrže ključne komponente i servise sistema Android, kao sto su ART (Android okruženje za izvršavanje) i HAL (Apstraktni sloj hardvera) napisane u izvornom kodu koji za izvršavanje traže biblioteke napisane u C/C++.

[1] Aplikacioni programski interfejs, definiše načine na koje aplikacije mogu da zahtevaju usluge od biblioteka ili operativnih sistema.

Operativni sistem Android pruža okvir Java API za korišćenje nativnih biblioteka u aplikacijama. Na primer, biblioteka WebKit obezbeđuje funkcije koje se odnose na pregledanje veb sadržaja ili biblioteka SQLite koja obezbeđuje podršku za korišćenje baza podataka, tako da aplikacija može da se koristi za skladištenje podataka.

- **Android okruženje za izvršavanje (eng. ART)** - na istom nivou kao i biblioteke, okruženje Android obezbeđuje skup osnovnih biblioteka koje omogućavaju programerima da pišu Android aplikacije korišćenjem programskog jezika Java. Sistem Android sadržao je do verzije 5.0 *Dalvik virtuelnu mašinu (eng. DVM)* specijalno projektovanu za sistem Android i optimizovanu za mobilne uređaje koji koriste baterije pri radu i imaju ograničene memorijske i procesorske resurse. Dalvik je preciznije proces virtuelne mašine koji izvršavaju specijalno pripremljene fajlove za njih sa sufiksom *.dex*. Android aplikacije pisane u programskom jeziku Java su prevedene u bajtkodove [2] namenjene za Java virtuelnu mašinu, koji se kasnije prevodi na Davlik bajtkod i skladišti u *.dex* datoteku. Nakon verzije 5.0 ART nasleđuje DVM, koji u odnosu na DVM ima velike prednosti. ART koristi iste datoteke sa ekstenzijom *.dex* i unapređen je u odnosu na Dalvik kompilacijom unapred *(eng. Ahead of time, AOT)* i unapređenim skupljačem otpadaka koji značajno unapređuje performanse aplikacije.
- **Radni okvir apikacija (Java API okvir)** - predstavlja celokupan skup funkcionalnosti sistema Android preko više API okvira napisanih u jeziku Java. Skup API okvira čini gradivne jedinice potrebne za kreiranje aplikacija za sistem Android, pojednostavljenjem ponovnog korišćenja jezgarnih sistemskih komponenti i servisa. Neke od najvažnijih komponenta ovog sloja su: *Menadžer aktivnosti* (upravljanje životnim ciklusom aplikacije), *Menadžer paketa* (sadrži informaciju o tome koje su aplikacije instalirane na uređaju), *Menadžer prozora* (upravljanje prozorima aplikacija), *Menadžer telefonije* (upravljanje pozivima), *Provajder sadržaja* (omogućava da više aplikacija koristi iste podatke), *Menadžer resursa* (brone o resursima aplikacija).
- **Aplikacije** - na ovom nivou nalaze se aplikacije koje se isporučuju sa Android uređajima (Imenik, Email aplikacija, Kamera, itd.), kao i aplikacije koje se preuzimaju i instaliraju korišćenjem Android prodavnice.

[2] Bajtkod je "mašinski jezik" za Java virtuelnu mašinu.



Slika 1.1. Prikaz arhitekure podeljen na sekcije i slojeve.

1.3 Verzije sistema Android

Operativni sistem Android je otvorenog koda, upravo to omogućava da se razvija prilično brzo, nova verzija je uglavnom izlazila svake godine i donosila nove mogućnosti. U listi su navedene glavne verzije sistema Android i nove mogućnosti koje su dolazile sa verzijama:

Android, verzija 1.0 (septembar, 2008):

- Skidanje i ažuriranje aplikacije preko *Android prodavnice*,
- internet pregledač,
- podrska za Wi-Fi i BlueTooth,
- Media player,
- integracija i sinhronizacija Google sevisa (Gmail, Google Contacts, Google maps, Google Calendar, Google Talk).

"Eclair", verzija 2.0 (oktobar, 2009)

- Veća brzina hardvera,
- unapređeni korisnički interfejs (UI),
- mogućnost dodavanja više korisničkih naloga,
- poboljšana brzina kucanja na tastaturi,
- proširena podrška za rezolucije i veličine ekrana, sa boljim kontrastom,
- mogućnost živih (animiranih) pozadina ekrana.

"Gingerbread", verzija 2.3 (decembar, 2010)

- Unapređenje i pojednostavljenje korisničkog interfejsa sa poboljšanim odzivom,
- podrška ekstra-large veličina ekrana (WXGA i veće),
- unapređena podrška za SIP i VoIp internet telefoniranje,
- jedno-dodirna copy/paste funkcija,
- novi Download Manager,
- podrška više kamera na uređaju,
- poboljšan mehanizam uštede energije
- lak pristup aplikacijama, audiju i grafici.

"Honeycomb", verzija 3.0 (februar, 2011)

- Optimizovana podrška za tablete sa novim "highgraphic"korisničkim interfejsom,
- podrška za procesore sa više jezgara,
- pojednostavljen mutitasking,
- mogućnost otvaranja više kartica u internet pretrazivaču,
- mogućnost enkripcije svih korisničkih podataka,
- podrška za HTTP proksi.

"Ice cream Sandwich", verzija 4.0 (novembar, 2011)

- Soft dugmići iz verija 3.x omogućeni za navigaciju umesto hardverskih,
- mogućnost pristupa aplikacijama direktno sa zaključanog ekrana,
- mogućnos snimanja ekrana (screenshot),
- poboljšane copy-paste funkcionalnosti,
- uvedena potrošnja podataka kao nova sekcija u postavci telefona meri upotrebu podataka),
- ugrađen editor forografija, unapređena kamera,
- hardversko ubrzanje korisničkog interfejsa,
- Wi-Fi direkt.

"Lollipop", verzija 5.0 novembar, 2014)

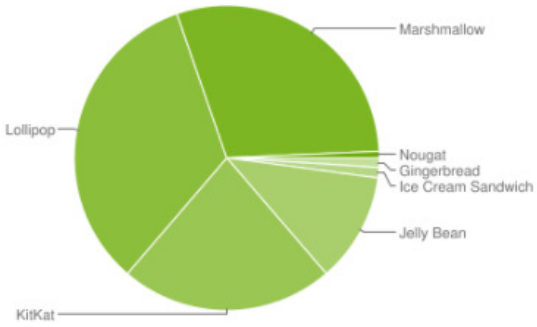
- ART sa AOT kompilacijom i poboljšanim garbage colection-om, menja Dalvik koji je kombinovao bajt-kod interpretaciju i JIT kompilaciju,
- podrška za procesore sa 64-bitnom arhitekturom,
- OpenGL ES 3.1,
- vektorska grafika, mogućnost skaliranja bez gubljenja kvaliteta,
- Material design, restilizovan korisnički interfejs usklađen sa Google aplikacijama,
- osvežen notifikacijski panel i panel brzog podešavanja,
- projekat Volta za poboljšanje i optimizaciju trošenja baterije,

- poboljšana zaštita.

"Marshmallow", verzija 6.0 oktobar, 2015)

- Predstavljen 'Dozen mode', koji smanjuje brzinu CPU dok je ekran isključen u cilju uštede baterije,
- čitač otiska prtiju,
- USB Type-C podrška,
- automatsko čuvanje i vraćanje svih aplikacijskih podataka,
- 4K režim prikaza za aplikacije.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.1%
4.1.x	Jelly Bean	16	4.0%
4.2.x		17	5.9%
4.3		18	1.7%
4.4	KitKat	19	22.6%
5.0	Lollipop	21	10.1%
5.1		22	23.3%
6.0	Marshmallow	23	29.6%
7.0	Nougat	24	0.5%
7.1		25	0.2%



Slika 1.2. Prikaz zastupljenosti android verzija, dana 9. januara, 2017.

1.4 Dobre strane sistema Android

Open Handset Alliance ističe nekliko bitnih karakteristika operativnog sistema Android:

- **Otvorenost** - Android je izgrađen tako da omogućava programerima stvaranje aplikacija koje u potpunosti koriste sve što uređaj nudi. Napravljen je da bude otvoren. Na primer, aplikacija može da poziva jezgarne funkcije mobilnog telefona kao što su pozivanje, slanje tekstualnih poruka, korišćenje kamere, dopuštajući programerima da stvore bogatije i složenije korisničke aplikacije. Temeljen je na otvorenom Linux jezgru. Koristi svoj virtualni proces koji je dizajniran da optimizuje memorijske i hardverske resurse uređaja. Android može da se proširuje i na taj način može da prihvati najnovije tehnologije. Platforma će nastaviti da raste sve dok programerska zajednica radi zajedno, i razvija inovativne aplikacije za mobilne uređaje. Android omogućuje proizvođačima laku integraciju hardvera sa sistemom, bez plaćanja autorskih prava.

- **Sve aplikacije su jednake** - Android ne razlikuje jezgarne aplikacije i aplikacije raznih programera kad je u pitanju pristup mogućnostima uređaja. I jedni i drugi imaju jednak pristup mogućnostima uređaja što dozvoljava korisnicima upotrebu širokog pojasa aplikacija i usluga. Uređaje koji su izgrađeni na platformi Android, korisnici mogu u potpunosti da prilagode svojim zahtevima. Na primer, mogu da konfigurišu postavku uređaj tako da koristi njihovu omiljenu aplikaciju za pregled slika.

- **Brz i jednostavan razvoj aplikacija** - Android pomera granice u stvaranju novih i inovativnih aplikacija. Android pruža pristup bogatoj bazi korisnih biblioteka i alata koji mogu da se koriste pri izradi aplikacija.

- **Dozvole** - Od 21.septembra 2008., Android je dostupa pod otvorenim kodom. Google je otvorio čitav izvorni kod, koji je pre bio nedostupan, pod licencom Apache. Sa licencom Apache, programeri mogu da dodaju svoja proširenja bez obaveze da ih daju zajednici.

1.5 Osnovne karakteristike Android aplikacija

Aplikacije za Android pisane su u programskom jeziku Java. Prevedeni Java kôd - zajedno sa svim podacima i datotekama resursa potrebnim za aplikaciju je arhiviran u datoteku obeleženu sufiksom *.apk*. Ova datoteka služi za distribuciju i instaliranje aplikacija na mobilnim uređajima. Datoteku korisnici preuzimaju na svoje uređaje. Sav kod u jednoj datoteci *.apk* smatra se jednom aplikacijom. Svaka aplikacija živi u svom svetu:

- Sistem svaku aplikaciju izvršava u posebnom procesu. Sistem Android počinje proces kad bilo koji deo koda aplikacije treba da se izvrši i isključuje proces kad više nije potreban.
- Svaki proces ima svoju specijalnu Java virtualnu mašinu (ART), pa kôd aplikacije radi u izolaciji od kodova svih ostalih aplikacija.
- Svakoj aplikaciji je dodeljen jedinstveni ID korisnika. Dozvole su postavljene tako da su datoteke aplikacije vidljive jedino samoj aplikaciji, iako postoje načini kako da se izvoze u druge aplikacije.

Većina aplikacija spada u jednu od sledećih kategorija:

- Aktivnost koja se izvršava u prvom planu *(eng. foreground activity)* - aplikacija bez pozadinske aktivnosti (npr. igre ili mape). Kad su izvan fokusa aplikacije ove vrste se uglavnom privremeno zaustavljaju.
- Pozadinska usluga (*eng. background service*) - aplikacije ograničene interakcije sa korisnikom koje se uglavnom izvršavaju u pozadini (npe. apikacije za upravljanje dolaznim pozivima).
- Aktivnosti sa prekidima - aplikacije koje podrazumevaju određeni stepen interakcije sa korisnikom, ali se uglavnom odvijaju u pozadini (npr. mp3 player).

Za razliku od većine aplikacija na drugim sistemima, Android-ove aplikacije nemaju samo jednu pokretnu tačku (konkretno znači da nemaju samo jednu funkciju `main()`).

Postoje osnovne četri komponente koje čine aplikaciju:

- **Aktivnosti** ***(eng. Activity)*** - predstavlja komponentu aplikacije koja se uglavnom poistovećuje sa jednim konkretnim prozorom aplikacije u kojem korisnik može da izvrši određenu radnju. Aplikacija može da sadrži jednu ili više definisanih aktivnosti, pri čemu je jedna od aktivnosti uvek definisana kao primarna aktivnost. Iako više aktivnosti stvara jedan kompaktni korisnički interfejs treba imati na umu da su one međusobno nezavisne. Svaka aktivnost implementira se kao posebna klasa koja nasleđuje klasu `Activity`, te je i sama odgovorna za čuvanje svog stanja u zivotnom ciklusu aplikacije.
- **Namera** ***(eng. Intent)*** - su poruke koje omogućavaju komponentama aplikacije da zahtevaju funkcionalnost od drugih komponenti. Omogućava komunikaciju sa komponentama iz iste aplikacije kao i sa komponentama drugih aplikacija. Namera je u osnovi pasivna struktura podataka koja sadrži apstraktni opis akcije koja se obavlja.
- **Usluga (*eng. Service*)** - predstavlja proces bez vidljive korisničke interakcije. Pošto usluga nema korisnički interfejs, ona nije vezana za životni ciklus neke aktivnosti. Uglavnom se izvršava u pozadini za nedefinisani period vremena. Usluge se koriste za ponavljajuće i potencijalno dugačke operacije, kao što su preuzimanje sa interneta, proveru novih podataka, obradu podataka, ažuriranje dobavljača sadržaja i slično. Svaka usluga nasleđuje klasu `Service`.
- **Prijemnik (*eng. BroodcastReceiver*)** - služi da prihvati poruke koje mogu da salju druge aplikacije ili čak sam sistem. Te poruke mogu da idu od sistemskih događaja kao što je uključivanje telefona ili kraj preuzimanja podataka sa interneta.
- **Dobavljač sadržaja (*eng. Content Providers*)** - pomažu aplikaciji prilikom pristupa sopstvenim podacima, podacima drugih aplikacija i prilikom deljenja podataka sa drugim aplikacijama. Dobavljači sadržaja su standardni interfejs koji povezuje podatke iz jednog procesa sa kodom koji se izvršava u drugom procesu. Dobavljači sadržaja nasleđuje klasu `ContentProvider`.

Aplikacija ne mora da sadrži sve navedene komponente, a isto tako može da sadrži i neke druge.

1.5.1 Korisnički interfejs

Postoje dva načina dizajniranja korisničkog interfejsa: *proceduralno* i *deklarativno*. Proceduralni dizajn odnosi se na pisanje Java koda, a deklarativno na pisanje XML (*EXtensible Markup Language*) koda. U praksi se za kreiranje korisničkog interfejsa uglavnom koristi XML, tj. deklarativni način. Kreiranjem interfejsa, aktivnosti dobijaju svoju funkcionalnost. Osnovne jedinice korisničkog interfejsa su objekti pogled *(eng. View)* i grupe pogleda *(eng. ViewGroup)*:

- **Pogled** - objekat čija struktura u sebi nosi zapis izgleda i sadržaj određenog pravougaonog područja na ekranu, te upravlja iscrtavanjem elemenata, pomicanjem sadržaja na ekranu (*eng. scrolling*) i ostalim faktorima koji utiču na izgled definisanog dela ekrana. Android raspolaže sa već gotovim skupovima objekata ove vrste, ovi objekti nasleđuju klasu `View`.
- **Grupe pogleda** - posebna vrsta objekata koja sadrži i upravlja skupom zavisnih objekata pogleda i grupe pogleda čime je omogućena kompleksnost prikaza korisničkog interfejsa. Objekti ove vrste su instance klase `ViewGroup`.

1.5.2 AndroidManifest.xml

Svaka aplikacija obavezno mora da sadrži datoteku *AndroidManifest.xml* koja je opisuje. Manifest datoteka sadži sve neophodne metapodatke za grupu pratećih datoteka koje su deo skupa ili koherentne jedinice. Ova datoteka nalazi se u korenskom direktorijumu paketa i sadrži sledeće informacije o aplikaciji:

- Naziv paketa - služi kao jedinstveni indentifikator aplikacije,
- opis komponenti - aktivnosti, usluge, dobavljač sadržaja, filter namena, prijemnik...
- odredbe o tome koji će proces sadržati programske komponente,
- deklaracija dozvola za pristup aplikacijskim komponentama od strane drugih aplikacija,
- minimalan nivo Android API-a koju zahteva aplikacija,
- lista biblioteka koje aplikacija koristi.

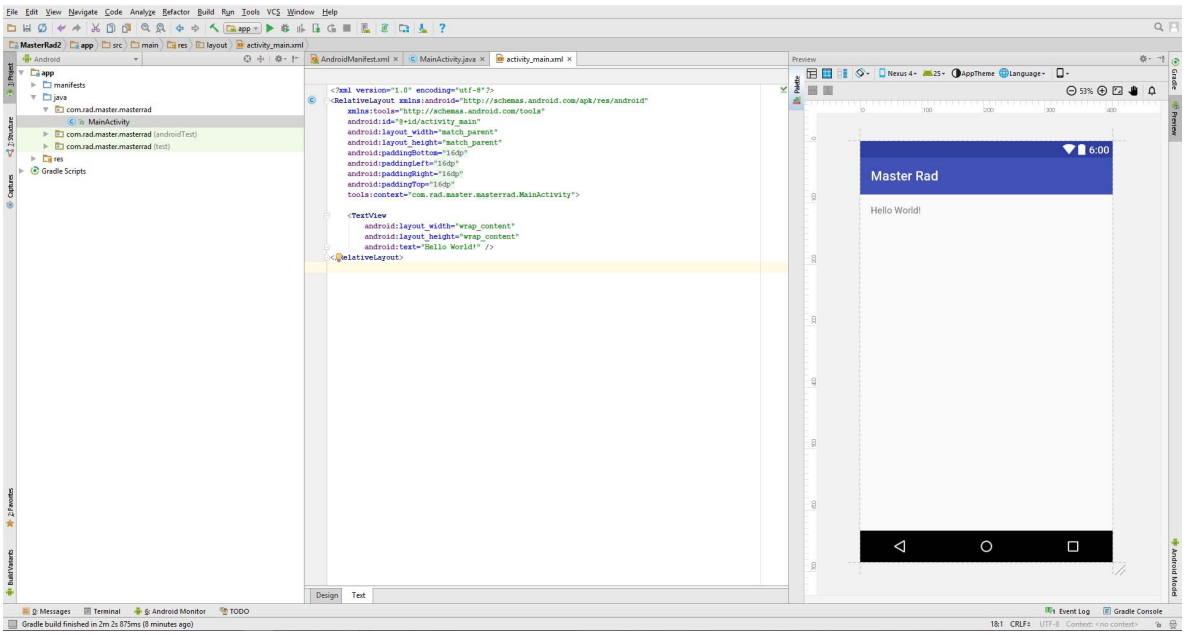
1.6 Android prodavnica

Jedan od glavnih faktora za utvrđivanje uspeha određene platforme pametnih telefona je broj aplikacija kreiranih za tu platformu.

Kompanija Google je u avgustu 2008. godine najavila Android Market, mrežno dostupnu prodavnicu aplikacija za Android uređaje, pristupanje je omogućio korisnicima dva meseca kasnije. Korišćenjem prodavnice korisnici mogu da preuzimaju aplikacije raznih autora direktno na svoje uređaje. U prodavnici se nude besplatne aplikacije kao i aplikacije koje se plaćaju. Aplikacije koje se naplaćuju dostupne su samo u određenim zemljama, zbog pravnih ograničenja. Slično tome, u nekim zemljama aplikacije sa Android prodavnice korisnici mogu da kupuju, ali programeri ne mogu da ih prodaju i obrnuto.

1.7 Alati za razvoj Android aplikacije

Android Studio je zvanični alat za razvoj aplikacija za Android platformu, objavljen je 2013. godine i postao zamena za do tad zvanični Eclipse alat sa ADT dodatkom.



Slika 1.3. Android Studio.

1.8 Razvoj igara za sistem Android

U sledećem poglavlju biće opisani načini na koje može da se razvija video-igra za sistem Android:

- **Android SDK** - Android SDK je robustan i potpuno funkcionalan skup API, koji omogućuje programerima da kreiraju aplikacije za platformu Android. Android SDK pruža dva API za programiranje grafike. Prvi pristup se zasniva na crtanju 2D grafike pomoću **Canvas API**, a drugi pristup se koristi za hardverski ubrzano crtanje 2D i 3D grafike pomoću **OpenGL ES API**. Za većinu projekta često Android SDK pruža programeru sve što je potrebno za razvoj jedne video-igre, Canvas API je dovoljan za prostije i manje zahtevne 2D igre dok se za 3D video-igre koristi OpenGl ES. OpenGL ES je grafički API koji određuje standardni softverski pristup za hardversku obradu 3D grafike. OpenGL ES je višenamenska grafička biblioteka koja podržava kreiranje 2D i 3D sadržaja, mehaničkog i arhitektonskog dizajna, simulaciju letenja, video-igre i još mnogo toga. Dizajniran je da prevodi pozive funkcija u grafičke komande koje se izvršavaju na grafičkom hardveru.
- **Izvorni razvoj** - Programeri koji žele da iskoriste svoje video-igre pisane u C/C++ mogu da izaberu Android NDK umesto prepisivanja svega. **Android NDK** je set alata koji omogućavaju primenu delova aplikacije pisane

u izvornom kodu, koristeći jezike C/C++. Na primer ukoliko imate razvijenu mehaniku igre pisane u C/C++ jeziku, možete da je iskoristite. Pomoću NDK se C/C++ kôd može u izvornu biblioteku, čije funkcionalnosti mogu da se koriste pomću poziv iz Java jezika. Često se koristi pri razvoju video-igara visokih performansi za sistem Android.

- **Okruženje za razvoj video-igara** - Okruženje za razvoj video-igara je programski okvir dizajiran za stvaranje i razvoj video-igara. Glavne funkcionalnosti pružaju grafički pokretač za 2D i 3D video-igre, pokretač fizike, audio, scripte, animacije, ai komponente, memorijsku kontrolu i lokalizaciju. Proces razvoja video-igre pomoću nekog od okruženja za razvoj video-igara je često ekonomičan pa se mogu iskoristiti i prilagoditi već napravljenje komponente. Okruženja za razvoj video-igara često imaju mogućnost multiplatformisanja što dodatno olakšava posao programerima.

  Neki od najpopularnijih i najboljih okruženja za razvoj video-igara su:

  - Unity
  - Unreal Engine
  - CryEngine
  - Source Engine

Glava 2

Video-igre

Video-igra je elektronska igra koja uključuje interakciju korisnika sa korisničkim interfejsom za generisanje vizuelne povratne informacije na video uređaj. Video-igra je igra koja se igra uz pomoć računara, mobilnih uređaja ili igračkih konzoli i spada u jedan od najpopularnijih oblika zabave. Sistemi koji se koriste za igranje video-igara nazivaju se platforme, primer platformi su računari i konzole. Ulazni uređaj za igranje video-igre se naziva kontroler i njegov oblik varira u zavisnosti od platforme. Najčesći konroleri su džojstici, miš, tastatura i ekran osetljiv na dodir. Uređaj za reprodukciju može da bude ekran (tv, monitor, ekran moblinog uređaja) ili uređaja za VR. Video-igre se suštinski razlikuju od aplikacija, jer aplikacijama pružamo određenu uslugu korisniku a video-igre imaju zabavnu ulogu.

2.1 Video-igre kao ozbiljna grana IT industrije

Video-igre spadaju u jednu od najprofitabilnijih grana kako IT industrije tako i zabavne industrije. U 2016. godini industrija video-igara dostigla je prihode od oko 91 milijardu dolara. Segment mobilnih igara je najzaslužniji za takav prihod sa udelom od 41 milijarde dolara (sa rastom od 18% u odnosu na prethodnu godinu), sa 26 milijarde dolara ih prate video-igre čija se izdanja nalaze u maloprodajnim prodavnicama (obično su to video-igre namenjene konzolama ili PC računarima, gde se kupuje CD sa video-igrom) i sa 19 milijardi dolara besplatne video-igre na internetu (gde se monetizacija odvija u samoj igri). Nove kategorije video-igara kao što su igre virtuelne stvarnosti *(eng. VR)*, e-sportovi *(eng. eSports)* i gledanje igranja video-igara *(eng. gaming video content)* su male u odnosu na najjače grane ali pokazuju konstantan trend rasta iz godine u godinu. [11].

2.2 Istorija Video-igara

Istorija video-igara kreće 1950-tih godina, prve video-igre razvijaju kompijuterski naučnici za potrebe svog naučnog istraživanja. Popularnost doživljavaju između 1970. i 1980. godine, kad se javnost upoznaje sa igrama i uređajima za upravljanje igrom, kao što su uređaji za ulazne podatke kontroleri, džojstici, dugmići i ekrani

kao uređaji za prikaz video-igre. Od 1980. godine video-igre postaju popularna forma zabave i deo moderne kulture u svim delovima sveta.

Među pionirima video-igara se posebno izdvajaju igre *Spacewar!* (svemirska borba), koju je razvila grupa studenata 1962. godine sa Massachusetts Institute of Technology, predvođena Stve Russell-om i igra *Tennis for Two* (simulacija tenisa) koju je razvio Američki fizičar William Higinbotham 1958. godine.

Slika 2.1. Na slici levo se nalazi video-igra Spacewar! na uređaju PDP-1, desno Tennis for Two na osciloskopu.

Pravi komercijalni dizajn i razvoj video-igara kreće 1970-tih, kad nastaju prve konzole i arkadne video-igre[1]. *Computer Space* (svemirska borba) je prva prodata komercijalna video-igra 1971. godine, koristila je crno-beli TV za prikaz i kontrolnu tablu za ulazne podatke na arkadnoj mašini. Definitivno video-igra koje je podigla polularnost video-igara i taj vid zabave je video-igra *Pong* (sportska arkadna video-igra) koja je izlazi iste godine, koju je razvila kompanija Attari. Zlatno doba arkadnih video-igara otvara igra *Space Invaders* koju je 1978. godine razvio Tomohiro Nishikado. Uskoro biva prilagođena za konzolu *Atari 2600* što dovodi do ekspanzije konzola. U tim godinama kreće i pojava kućnih računara na tržištu, što dovodi do novih video-igara koje su nastale zahvaljujući samostalnim programerima.

U ranim 2000-tim godinama , mobilne igre dobijaju na popularnosti, ali svoju pravu ekspanziju doživljavaju tek posle lansiranja Android prodavnice i Apple App prodavnice 2008. godine.

2.3 Klasifikacija video-igara na žanrove

Žanr video-igre je klasifikacija dodeljena na osnovu načina igranja video-igre *(eng. gameplay)* i nije vezana za vizuelne ili narativne razlike. Video-igra moze da spada u više žanrova. Deskriptivna imena žanrova uzimaju u obzir ciljeve igre, protagoniste pa čak i perspektive iz kojih se igra. Na primer beskrajno trčanje (eng. endless run) karakteriše dinamičnu video-igru sa karakterom u konstantnom pokretu ili pucačina

[1] Arkadna video-igra je mašina za zabavu instalirana na nekom javnom mestu, radi po principu žetona za igru koji se kupuju.

iz prvog lica (eng. first-person shooter) je video-igra koja se igra u perspektivi prvog lica (ljudska perspektiva) i uključuje gađanje vatrenim oružijem. Često žanrovi sadrže podžanrove, pa je primer pucačina iz prvog lica, u prvoj klasifikaciji samo pucačina a onda unutar klase, u podklasu pucačina iz prvog lica, paralela je pucačina iz treceg lica.

Video-igre se najčešće klasifikuju u sledeće žanrove[18]:

- **Akcione** *(eng. Action)* - Akcione video-igre naglašavaju izazove u kojima se od igrača zahteva kordinacija oka i ruke kako bi se oni prevazišli. Igrač je u prvom planu a sama dinamika igre je takva da od korisnika zahteva stalnu interakciju. Akcione igre se dele na mnoge podžanrove. Platformske igre *(eng. Platform games)* i igre borbi *(eng. Fighting games)* spadaju u najpoznatije podžanrove, dok su igre pucačine *(eng. Shooter games)* jedne od najdominantnijih podžanrova u video-igrama.
- **Avanturističke** *(eng. Adventure)* - Avanturističke video-igre su igre u kojoj igrač ima ulogu protagoniste u interaktivnoj priči gde je priča vođena istaživačkim i zagonetnim problemima.
- **Avanturističko-akcione** *(eng. Action-adventure)* - Avanturističko-akcione video-igre kombinuju elemente iz dva žanra akcionog i avanturističkog. Zahtevaju slične fizičke zahteve od korisnika kao akcione igre i glavne elemente avanturističkih igara, više karaktera, jednostavnije slagalice, inventar karaktera, dijalog i druge elemente avanturističkih igara. Dinamika igranja se razlikuje od klasičnih avanturističkih igara zato što uključuju i fizičke i konceptualne izazove.
- **Igranje uloge** *(eng. Role-playing)* - Igranje uloga je žanr u kojoj igrač kontoliše postupke svog karaktra (ili više njih) u izmišljenom svetu.
- **Simulacija** - Cilj ovog žanra je simulacija realnih aktivnosti. Obično u simulaciji nema definisanih ciljeva igre, već igrač ima punu kontrolu nad karakterom. Često se simulacije pored zabave koriste i za učenje, trening, analizu i predviđanje.
- **Strategija** - Strategijske video-igre spadaju u klasu u kojima je cilj i fokus na razmišljanju i veštini, kako bi se ostvarila pobeda. U igri je akcenat na strateškim, taktičkim i logičkim izazovima, često su tu uključeni i ekonomski i istrazivački izazovi.
- **Sport** - Sportske video-igre su video-igre u kome se simulira određen sport. Pored klasičnog igranja sporta u video-igri, neke igre naglašavaju i stratešku i menadžersku stranu. Ovakve igre imaju veliku popularnost i takmičarski duh kao i pravi sportovi.

2.4 Podela video-igara prema grafici

Grafika je uobičajeni naziv za vizuelnu prezentaciju okruženja video-igre, kao i za vizuelne reprezentacije svake pojedinačne komponente u njoj. Video-igru prema grafici možemo da klasifikujemo na **2D video-igru** ili **3D video-igru**.

2D označava dve dimenzije, tj. da se radnja dešava u dvodimenzionoj ravni. Video-igre u 2D možemo da posmatramo samo iz jedne perspektive. 3D video-igre imaju grafički prikaz takav da simulira realan svet, gde za razliku od 2D video-igara imamo i dubinu. U 3D video-igrama karakter možemo da vidimo iz više perspektiva, dok igrači imaju sećaj da sve što se dešava u domenu video-igre je u saglasnosti sa realnom fizikom i gravitacijom. Vizuelni sadržaj izgleda, ponaša se i oseća stvarno, čak i kad se radi o naučnoj fantastici. 3D video-igre se zasnivaju na modelima[2], dok se 2D igre zasnivaju na ravnoj grafici koja se zove sprajt[3] *(eng. Sprite)* koja uopste nema trodimenzionalnu geometriju. 2D video-igre često se nazivaju i platformske igre, gde reč platforma opisuje da se radnja video-igre odvija na nekoj platformi na kojoj igrač može da trči, skače, puca i skuplja moći. Pojavom 3D video-igara, 2D video-igre počinju da gube na popularnosti. Ponovo postaju popularne pojavom mobilnih uređaja, koji imaju potrebu za jednostavnijom grafikom.

Neke 2D video-igre koriste 3D sadržaje za okolinu i karakter, ali ograničavaju igru na dve dimenzije. Primer je kad je igra u bočnom pomeranju po strani, a karakter može da se pomera samo u dve dimenzije. Sama igra koristi 3D sadržaj i 3D perspektivu kamere bez funkcionalnosti, samo sa estetskim ciljem. Ovakva vrsta video-igara se naziva **2.5D video-igre**.

2.5 Razvoj video-igara

Razvoj video-igre je softverski proces koji za cilj ima pravljenje video-igre. Igru programira jedan ili više programera (ponekad i cela kompanija). Komercijalne igre za PC računare ili konzole razvija i finansira izdavač, obično proces razvoja traje po nekoliko godina. Za razliku od komercijalne igre, nezavisna igra *(eng. Indie-game)* je video-igra koja je kreirana bez finansijske podrške izdavača, i sa mnogo manjim timovima, neretko i pojedincima. Nezavisne video-igre su poslednjih godina u rastu, upravo zbog mobilnih i online platformi.

2.5.1 Uloge u razvoju video-igara

Video-igru moze da razvije jedan programer ili više njih, uloge u razvoju video-igre možemo da podelimo na:

- **Producent** *(eng. Producer)* - je osoba zadužena za nadgledanje i finansiranje razvoja video-igre. Producent je kontrolno telo u razvoju igre i sprega između razvojnog tima i izdavača.
- **Izdavač** *(eng. Publisher)* - izdavač video-igara je kompanija koja izdaje video-igru. Izdavač finansira razvoj video-igre, preko spoljnih programera ili preko svog studija za igre. Izdavač je takođe odgovoran i za ostale stvari prilikom izdavanja video-igre kao što su marketing, distribucija, način prihodovanja...

[2]Matematička reprezentacija nekog tridemnzionalnog objekta.
[3]Mala slika koja može da se kreće po grafičkom ekranu.

Programeri nezavisnih video-igara, razvijaju igre bez izdavača i često se odlučuju na digitalnu distribuciju svoje igre.

- **Razvojni tim** *(eng. Development team)* - moze da varira u broju, od malih timova koji razvijaju manje igre do nekoliko stotina koji razvijaju par velikih naslova.
  - **Dizajner** *(eng. Designer)* - je osoba koja dizajnira igrivost video-igre, koncipira i dizajnira pravila i strukturu igre. Dizajneri su glavni vizionari igre.
  - **Umetnici***(eng. Artist)* - kreiraju vidljiv sadržaj potreban za video-igru. Posao umetnika može biti orijentisan na 2D ili 3D sadržaj. 2D umetnici obično kreiraju teksture, pozadinsko okruženje, slike terena i korisnički interfejs. 3D umetnici kreiraju modele, animacije, 3D okruženje. Često umetnici rade obe uloge.
  - **Programer** *(eng. Programmer)* - primarno razvija video-igre ili srodne softvere, često alate za razvoj igara. Programeri su zaduženi za celukupni kod u video-igri. Obično je jedan ili više glavnih programera *(eng. Lead programmer)*, koji implementiraju osnovu i nadgledaju dalji razvoj i nove module.
  - **Dizajner nivoa** *(eng. Level designer)* - kreira nivoe, izazove, ili misije za video-igru uz pomoć specijalnog skupa programa. Obično taj skup programa razvijaju programeri za njihove potrebe.
  - **Inzinjeri zvuka** *(eng. Sound engineer)* - zaduženi za celokupan zvuk u video-igri, od zvučnih efekta, pozadinske muzike do naracije u potrebnim delovima.
  - **Ispitivači***(eng. Tester)* - garanciju kvaliteta sprovode ispitivači. Ispituju vizuelnu i zabavnu stranu igre, kao i celokupnog skupa karakteristika igre, kompatibilnost, lokalizaciju...

2.5.2 Proces razvoja video-igre

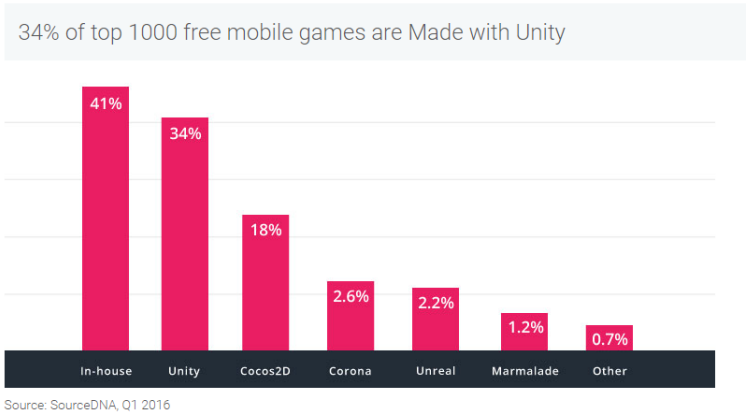
Razvoj video-igre, izrada ili dizajn je proces koji polazi od ideje ili koncepta. Često je ideja modifikacija postojećeg koncepta igre. Pri razvoju video-igara se često ne primenjuju klasične metode pri razvoju softvera. Jedan od najcešćih metodologija za razvoj video-igara je agilna metodologija Skram. Agilne metodologije predstavljaju alternativu tradicionalnom upravljanju projektima. Razvoj se odvija u iteracijama, gde se svaka iteracija ponaša kao projekat za sebe. Redovnim ispitivanjem i prilagođavanjem postiže se bolji kvalitet proizvoda. Skram je najpopularnija agilna metodologija, zahvaljujući svojoj jednostavnosti i fleksibilnosti. Skram nije jedinstven proces ili tehnika za pravljenje proizvoda, već okvir za kobinovanje više procesa ili tehnika, što je doprinelo njegovoj širokoj prihvaćenosti. Pored toga što olakšava razvojnim timovima proces organizacije, na kraju svakog ciklusa, ulagači dobijaju funkcionalan proizvod.

Glava 3

Unity

3.1 Uvod

Unity je višeplatformsko okruženje za razvoj video-igara *(eng. Game engine)* razvijan od strane kompanije Unity Technologies[5]. Služi za razvoj video-igara za PC, mobilne uređaje, igračke konzole i veb-sajtove. Unity predstavlja potpuno opremljen alat za razvoj video-igara i to je jedan od razloga njegove velike popularnosti. Poseban akcenat je stavljen na prenosivost, upravo zbog toga pokriva sledeće grafičke API-je: *Direct3D* i *Vulkan* na Windows i XBox 360 platformi, *OpenGl* na Mac, *OpenGL ES* na Android i iOS platformi i grafičke API-je konzola PlayStation, Wii U i Nitendo. Unity omugućava specifikaciju kompresije tekstura i rezolucije na svakoj platformi, takođe podržava grafičke tehnike za mapiranje reljefa *(eng. bump mapping)*, mapiranje refleksije *(eng. reflection mapping)*, mapiranje dubine *(eng. parallax mapping)*, aproksimaciju ambijentalnog svetla *(eng. SSAO)*, dinamičke senke, itd. Raznovrsnost pokretača grafičke platforme Unity-a omogućava prigrame za senčenje *(eng. shader)* sa više varijanti, čime automatskim detektovanjem najbolje varijante za graficki hardver izabira najbolju rešenje, uvek stavljajući akcenat na performanse ispred izgleda. Unity je takođe vodeće globalno okruženje za razvoj video-igara. Statistika pokazuje da su 34% od najtraženijih 1000 mobilnih video-igara razvijane upravo okruženjem Unity [16].



Slika 3.1. Procenat mobilnih video-igara napravljenih pomoću okruženja Unity.

3.1.1 Arhitektura i kompilacija

Okruženje za razvoj video-igara Unity je skup C/C++ biblioteka koje pomažu pri razvoju video-igre. Kôd se piše u C#, JavaScript (UnityScript) ili jeziku Boo. Sav napisan kôd se odvija u okviru Mono ili Microsoft.NET, koji je Just-in-Time (JIT)[1] kompajliran (izuzev iOS, koji ne dozvoljava JIT vec Ahead-of-Time (AOT)[2] kompilaciju). Okruženje Unity dozvoljava testiranje napisanog koda bez potrebe za eksportovanjem ili pravljenjem izvršne verzije. Prilikom pokretanja koda u Unity-u, koristi se okvir Mono, koji ima API grubo kompatibilan sa .NET okvirom.

3.2 Razvoj igara za razne platforme

Jedna od glavnih karakteristika Unity-a je razvoj igara za razne platforme, video-igre napravljene u njemu mogu da se izvršavaju na ukupno 27 različitih platformi. Platforme mozemo grubo podeliti u šest kategorija:

- **Mobilni uređaji** - Unity je najpopularnije okruženje za razvoj video-igara za mobilne uređaje. Podržava platforme Android, iOS, Windows Phone, Tizen i Fire Os.

Slika 3.2. Prikaz svih podržanih OS za mobilne uređaje.

- **Virtuelna stvarnost** *(eng. VR)* i **Proširena stvarnost** *(eng. AR)* - U korak sa vremenom, podrška za sledeće platforme virtuelne stvarnosti: Oculus Rift, Gear VR, Playstation VR, Microsoft HoloLens, Steam VR/Vive and Google Daydream.

Slika 3.3. Prikaz svih podržanih VR i AR uređaja i sistema.

- **Desktop** - Podrška za desktop operativne sisteme, izvršne verzije se mogu praviti za Mac OS X, Windows i Linux. Opciono je biranje između 32-bitne i 64-bitne arhitekture sistema.

[1] Pravovremena kompilacija se vrši za vreme izvršavanja programa, tj. za vreme dinamičkog punjenja klasa.

[2] Komplajliranje u kome se bajtkod kompajlira u mašinski kôd tokom instalacije na uređaj.

Slika 3.4. Prikaz svih podržanih desktop OS-a.

- **Igračke konzole** - Podrška za vodeće igračke konzole PS4, Xbox One, PlayStation Mobile, PlayStation Vita i Wii U, besplatno.

Slika 3.5. Prikaz svih podržanih konzola.

- **Veb** - Za izvršavanje u okruženju veb se koristi WebGl okvir. Prestala je podrška za Unity Web player (Flash player), ukidanjem podrške za Flash player od vodećih pretraživača.

Slika 3.6. Podržana veb tehnologija.

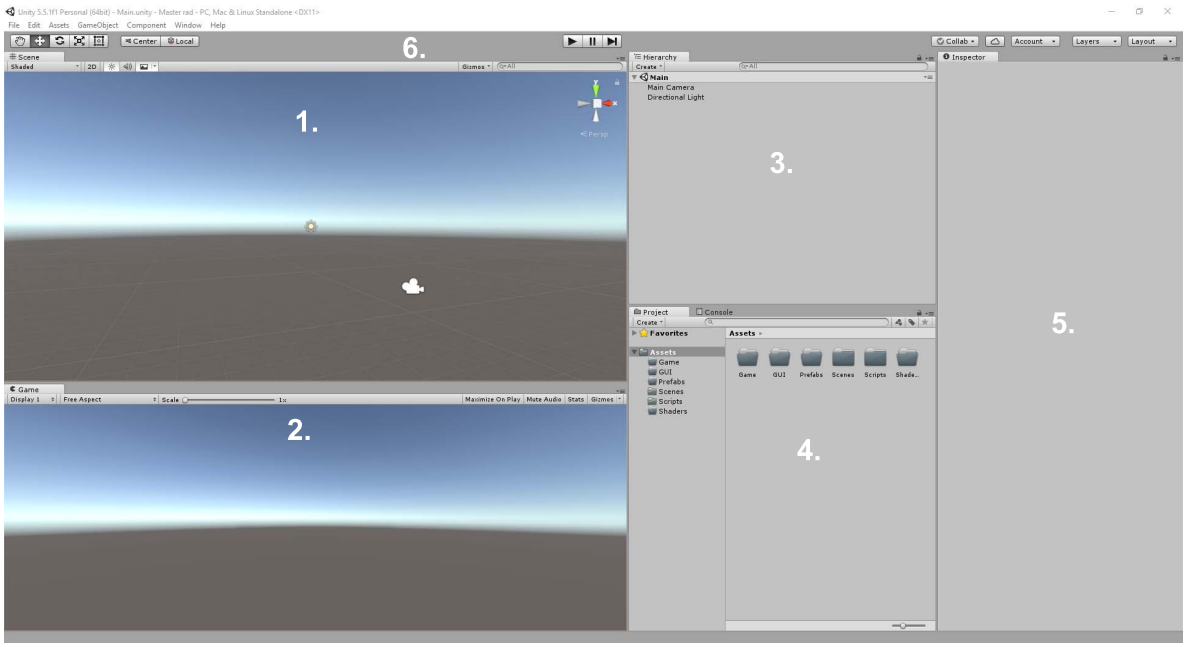
- **Pametni televizori** - Unity podržava izvršavanje i na platformama za pametne televizore tvOS, Android Tv i Samsung Smart TV.

Slika 3.7. Prikaz svih podržanih platformama za pametne televizore.

3.3 Razvojno okruženje

3.3.1 Unity editor

Unity razvojno okruženje se sastoji iz dve celine, Unity editora i razvojnog okruženja MonoDevelop. Centralna celina je Unity editor, gde programer ima integrisane sve osnovne delove neophodne za razvoj video-igre, kao što su alat za pravljenje scena, alat za reprodukciju video-igre, inspektori objekta, alati za animaciju (2D i 3D). Pored oruženja MonoDevelop Unity podržava i razvojno okruženje Visual Studio.



Slika 3.8. Prikaz Unity editora.

1. **Pogled scene** *(eng. Scene view)* - je prozor u kom se odvija vizuelna konstrukcija video-igre i odvija rad sa centralnim entitetima *(eng. Game Objects)*. U sebi ima mogućnost dodatnih podešavanja detalja scene.

2. **Pogled video-igre** *(eng. Game view)* - prozor gde video-igra oživljava posle pritiska na dugme *Play* u delu editora 6.

3. **Hijerarhija objekta u sceni** *(eng. Hierarchy)* - lista svih objekata video-igre *(eng. Game Objects)* u sceni, sortiranih abecedno. Objekti mogu u sebi da sadrže decu objekte i tako u dubinu. Redosled je jako bitan i utiče na redosled renderovanja objekta.

4. **Sadržaj projekta** *(eng. Project)* - sadrži sve komponente(skripte, teksture, 3d modele, audio snimke...) i foldere korišćene u projektu.

5. **Inspektor** *(eng. Inspector)* - prikazuje komponente izabranog objekta, deo u kom može da se barata sa pozicijom objekta, rotacijom i skaliranjem.

6. **Traka alata** *(eng. Toolbar)* - u levom uglu se nalaz dugmad za upravljanje scenom: *Pan*, *Rotate*, *Move*, *Scale* a u centru trake *Play*, *Pause*, *Advance Frame*. Klikom na dugme *Play* pokrećemo igru, bez potrebe da igru pre toga

izgradimo u izvršni fajl. Posebno bitan deo je dugme *Advance Frame* koje prikazuje igru kadar po kadar, što mnogo olakšava proces debagovanja.

3.3.2 MonoDevelop

MonoDevelop je integrisano razvojno okruženje otvorenog koda za Linuks, OS X i Windows. Primarni fokus je razvoj projekta koji koriste Mono ili .NET okvir. MonoDevelop integriše funkcije kao što su: automatsko izvrašavanje koda, izvorna kontrola, grafički korisnički interfejsi (eng. GUI) i web dizajner. Podržava jezike:

- **C**,
- **C++**,
- **C#**,
- **Java**,
- **Boo**,
- **CIL**,
- **D**,
- **F#**,
- **Oxzgen**,
- **Vala** i
- **Visual Basic.NET**.

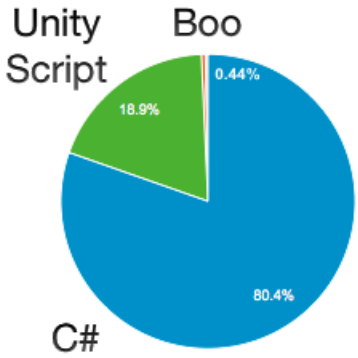
Nastao je iz razvojnog okruženja otvorenog koda SharpDevelop, sa ciljem da se okvir .NET zameni okvirom Mono. Danas ga razvijaju i podržavaju zajednice oko Xmarin i Mono projekta. U okruženju Unity je integrisana prilagođena verzija okruženja MonoDevelopa.

3.3.3 Jezici za razvoj

Okruženje Unity omogućava programerima da komfor nađu u nekom od tri podržana jezika ili mogućnost da kombinuju jezike, iako se to ne preporučuje. Podržani jezici su:

- **C#**,
- **JavaScript(UnityScript)**,
- **Boo**.

Razlike u performansi jezika su zanemarljive, tako da većina programera izabere jezik sa kojim ima iskustva i sa kojim se najudobnije oseća. Programeri se najčešće odlučuju za upotrebu C# jezika, pa se upravo za njega može naći veliki broj primera kodova i visoko razvijena zajednica, dosta manje se koristi JavaScript za koji se ređe nalaze pimeri kodova. Unity od verzije 5 ukida podršku za dokumentaciju jezika Boo, takođe skida opciju za kreiranje Boo skirpte iz menija. Podrška za izvršavanje Boo skirpti u postojećim projektima ostaje. Zbog ogromne popularnosti C# doživljava upravo suprotno, svi demo projekti su uglavnom pisani u njemu, uz nastojanje da se trenutna dokumentacija poboljša i dopuni. Svi dostupni jezici su objektno-orijentisani i imaju ugrađen mehanizam za skupljanje otpadaka.



Slika 3.9. Na slici je prikazana procentualna upotreba jezika.

U okruženju Unity dostupna su tri jezika za razvoj:

C#

C# je više-paradigman jezik koji sveobohvata strogo-tipiziranu, imperativnu, deklarativnu, funkcionalnu, objektno-orijentisanu i komponentno-orijentisanu programsku paradigmu. Široko se primenjuje za programiranje i u okruženju Unity i van njega.

```
using UnityEngine;
using System.Collections;

public class TouchTest : MonoBehaviour
{
    void Update ()
    {
        // Inicijalizacija statusa prvog dodira na ekranu
        Touch myTouch = Input.GetTouch(0);

        // Inicijalizacija svih statusa dodira na ekran
        Touch[] myTouches = Input.touches;
        for(int i = 0; i < Input.touchCount; i++)
        {
            // Petlja za prolazak kroz listu statusa dodira
        }
    }
}
```

Skripta 3.1 Primer koda u C# jeziku.

JavaScript (UnityScript)

UnityScript ili kako ga često zovu JavaScript je drugi najzastupljeniji jezik pri izboru programera. U odnosu na C# ima prostiju sintaksu, dinamički-tipiziran i sintaksno

je sličan JavaScriptu. UnityScript suštinski nema prednosti u odnosu na C#, koji je bolje dokumentovan. Autorovo misljenje je da je JavaScript pogrešan naziv, zbog fundamentalnih razlika između. JavaScript je prototipsko-orijentisan[3] jezik za razliku od UnityScripta, koji sadrži klase.

```
pragma strict

function Update ()
{
    // Inicijalizacija statusa prvog dodira na ekranu
    var myTouch : Touch = Input.GetTouch(0);

    // Inicijalizacija svih statusa dodira na ekran
    var myTouches = Input.touches;
    for(var i = 0; i < Input.touchCount; i++)
    {
        // Petlja za prolazak kroz listu statusa dodira
    }
}
```

Skripta 3.2 Primer koda u JavaScript jeziku.

Boo

Boo je jezik čija podrška prestaje sa verzijom okruženja Unity 5, čak se izbacuje i mogućnost pisanja skripti u Boo jeziku. Boo je jezik sa sintaksom sličnoj Python jeziku.

```
import UnityEngine
import System.Collections


public class TouchTest(MonoBehaviour):

    private def Update():
        # Inicijalizacija statusa prvog dodira na ekranu
        myTouch as Touch = Input.GetTouch(0)

        # Inicijalizacija svih statusa dodira na ekran
        myTouches as (Touch) = Input.touches
        for i in range(0, Input.touchCount):
            # Petlja za prolazak kroz listu statusa dodira
            pass
```

Skripta 3.3 Primer koda u jeziku Boo.

[3]Poseban stil objektno orijentisanog programiranja, rešava limitaciju jezika koji sadrži samo objekte, ne i klase, pa se nasleđivanje odvija preko ugrađenog mehanizma prototipskog-nasleđivanja.

3.4 Visual Studio

Visual Studio je integrisano razvojno okruženje, koje razvijeno od strane kompanije Microsoft. Služi za razvijanje aplikacija za Microsoft Windows OS, za veb-stranice, veb aplikacije, veb servise i mobilne aplikacije.

Visual Studio je jedan od najboljih i najkompletnijih razvojnih orkuženja na tržistu pa je rad u njemu definitivno više nego udoban, kako za početnike tako i za iskusne programere. U ovom radu, pri implementaciji video-igre kao razvojno okruženje umesto uobičajenog okruženja MonoDevelop koristiće se Visual Studio. Autorovo mišljenje je da početnicima i manje iskusnim programerima okruženje Visual Studio može da pruži udobniji rad.
Visual Studio podržava različite programske jezike, uključeni jezici su:

- C,
- C++,
- C++/CLI (Visual C++),
- VB.NET (Visual Basic .NET),
- C#(Visual C#),
- F#,
- XML/XSLT,
- HTML/XHTML,
- JavaScript i
- CSS.

Podrška za jezik Java (i J#) je uklojena. Postoji i mogućnost proširavanja postojeće podrške instalacijom dodatka za sledece programske jezike:

- Python,
- Ruby,
- Node.js i
- M

Kompanija Microsoft pruža besplatnu verziju okruženja Visual Studio pod nazivom Community edition, koji podržava dodatke i besplatan je za sve korisnike.

3.5 Prodavnica komponenti *(eng. Asset store)*

Unity prodavnica komponenti nastoji da osnaži programera video-igara sa komponentama i sredstvima koji su im potrebni, tako da mogu da se fokusiraju na kreativan proces izrade video-igre što efektivnije i efikasnije mogćue. Prodavnica komponenti je najveći komercijalni repozitorijum komponenti video-igara, podeljenih i struktuiranih u celine među kojima se nalazi kôd video-igara (skripte), 3D modeli, audio zvuci, shaderi, ekstenzije za editor, teksture i materijali kao i kompletni demo projekti.

2010. godine Unity lansira prodavnicu komponenti za zajednicu. Kako sami navode ideja je demokratizacija razvoja video-igara. Sama ideja prodavnice je da se

napravi jedinstveno mesto gde programeri, kreatori sadržaja, umetnici i dizajneri zvuka mogu da izlože svoje veštine i profitiraju na njima. Jedan od razloga za razvojem Unity prodavnice komponenti je i što u tom trenutku nijedna od postojećih prodavnica komponenti nije dizajnirana da odgovara Unity programeru i većina njihovog sadržaja nije bila pogodna za upotrebu u okruženjima za razvoj video-igara. Prodavnica video-igara je odlična simbioza gde iskusni specijalisti u konkretnim oblastima, popunjavaju sadržaj prodavnice i time manje iskusnim korisnicima omogućuju da lako svoje nedostatke popune, korišćenjem dostupnih resursa. Prodavnica igara je pored stranice na veb-u integrisana i u sam editor, pa je programeru nalaženje i preuzimanje komponenti olakšano. Sve komponente su pripremljene za upotrebu bez potrebe za dodatnim dograđivanjem ili podešavanjem. Većina komponenti su besplatne, oni koji žele mogu da prodaju svoj rad, pri čemu dobijaju 70% od cene komponente. Tok je takav da postavljenu komponentu proverava stručni tim prodavnice komponenti, ukoliko je komponenta prošla reviziju određujete cenu i komponenta postaje vidljiva na prodavnici komponenti.

3.6 Najpopularniji naslovi

Naslovi koji su dostigli svetsku popularnost a napravljeni su uz pomoću okruženja Unity, potvrđuju njegov kvalitet. U dole navedenim naslovima vidimo da okruženje Unity nije izbor samo mladim i manje iskusnim timovima već i velikim kompanijama koji u njemu vide kompletno rešenje za razvoj video-igre.

- **Hearthstone: Heroes of Warcraft** (Blizzard 2014, žanr - kartaška igra) - kolekcionarska video-igra sa kartama razvijena i publikovana od strane Blizzard Entertainment. Igra je brzo dostigla veliku popularnost i milionski auditorijum. Kroz popularnost video-igre Hearthstone okruženje Unity je dokazalo svoj kvalitet i pouzdanost, kao i da je često izbor velikih kompanija.
- **Temple Run** (Imangi Studios 2011, žanr - beskrajno trčanje) - video-igra beskrajnog trčanja razvijena i publikovana od strane Imangi Studios. Video-igra je razvijena za mobilne uređaje, doživljava ogromnu popularnost i biva skinuta sa prodavnica preko milijardu puta.
- **Assassin's Creed: Identity** (Ubisoft 2014, žanr - avanturistički-akciona) - video-igra je avanturistički-akcionog žanra, razvijena od strane Blue Byte i publikovana od strane Ubisoft-a. Video-igra je iz Assassins Creed kolekcije i razvijena je za mobilne uređaje.

Slika 3.7. Izgled video-igre Assasins Creed.

3.7 Podsistem za izračunavanje fizike

Podsistem za fiziku je najvažniji podsistem u okruženju Unity. Okruženje Unity korist podsistem za izračunavanje fizike Box2D u 2D-u i podsistem za izračunavanje fizike NVIDIA PhysX 3.3 u 3D-u. NVIDIA PhysX je moćan podsistem za izračunavanje fizike, omogućuje realnu fiziku i olakšava posao programerima nuđenjem osnovnih komponenti mehanike. Fizički pokret simulira ponašanje objekata kako reaguju jedni na druge kad se primeni sila na njih. Sile mogu biti konstantne poput gravitacije ili moment vozila, dok su druge kratke i moćne poput eksplozija. Osnovna komponenta fizike se naziva kruto telo *(eng. Rigidbody)* ona omogućuje fizičko ponašanje dodeljenom objektu. [4]

3.8 Licence(Dozvole)

Pro verzija sadrži dodatne grafičke funkcionalsnosti. Besplatna verzija sa druge strane prikazuje početni ekran *(eng. splash screen)* sa oznakom Unity-a i vodenim žigom u veb video-igrama koji ne moze biti izmenjen ili isključen.

Obe verzije uključuju razvojno okruženje, uputstva, demo projekte i sadržaj, podršku putem veb foruma, pristup wiki stranicama i budića ažuriranja.

U Unity-u postoje četri vrste licenci(dozvola) za programere:

- **Unity Free** - besplatna licenca, osim ako korisnik nije komercijalna organizacija sa godisnjim bruto prihodom od preko 100.000$ ili obrazovna, akademska, neporfitabilna ili vladina institucija sa ukupnim godisnjim budžetom koji prelazi 100.000$. Besplatna verzija prikazuje na početnom ekranu oznaku Unity-a i vodeni žig u veb video-igrama koji ne moze biti izmenjem.

[4] U fizici, kruto telo je čvrsto telo čija je deformacija nula ili tako mala da se može zanemariti.

- **Unity Plus** - licenca po ceni od 35$/mesečno. Nova verzija Unity-a specijalno dizajnirana za profesionalce i timove sa potrebom za efikasnijim. Minimali bruto prihod do 200.000$ za godinu dana, opcioni početni ekran.
- **Unity Pro** - licenca po ceni od 125$/mesečno obaveza pretplate na godinu dana. Sadrži sve funkcionalnosti kao i Plus verzija sa dodatnim poboljšanjima.
- **Unity Enterprise** - za velika preduzeća koja imaju potrebu za izvornim kodom i specijalnom podrškom za preduzeća.

Glava 4

Implementacija 3d edukativne video-igre

4.1 Uvod

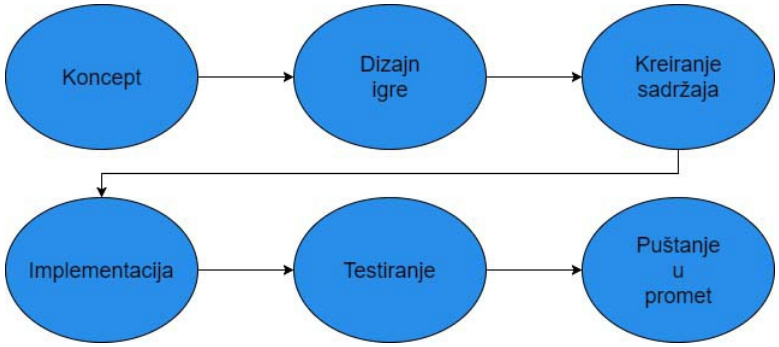
Cilj ovog rada je da se kroz razvoj edukativne video-igre *"Solarni Sistem"* prođu i pokažu osnovni koncepti okruženja Unity, osnovni entiteti i sam sistem komponenti na kome je zasnovana cela arhitektura. Kroz samu implementaciju prolaziće se kroz osnovne gradivne jedinice jedne video-igre, postojeće komponente i implementaciju komponenti zasnovanih na funkcionalnostima planiranim za video-igru. Zbog specifične platforme prikazaće se tehnika za rad sa ulaznim podacima sa mobilnih uređaja kao i implementacija prilagodljivog korisničkog interfejsa zbog širokog spektra hardverskih razlika mobilnih uređaja. U poglavlju 4.4.7 prikazana je tehnika i implementacija višejezičnosti za video-igru, tehnika je usvojena iz koncepta razvoja klasične Android aplikacije. Upravo je jedan od najmoćnijih mogućnosti okruženja Unity razvoj video-igara za razne platforme, gde je u tom delu opisano kako da video-igru distribuirate na više različitih platformi, uz pripremu svih sadržaja za konkretnu platformu, kao i rad sa ulaznim podacima.

4.2 Plan razvoja

Pre svake implementacije, prethodi faza planiranja koja omogućuje kasnije lakšu i bržu implementaciju. Samo planiranje je jako važno jer su nedovoljno planirani projekti često osuđeni na neuspeh. Upravo jasnim planiranjem i rasčlanjenjem na faze samu celinu cepamo na manje delove, lakše za manipulaciju. Svaka faza zavisi od predhodne faze i faze se ne mogu preskakati.

Faze razvoja su sledeće:

- **Koncept** - koncept je početna faza i postavlja okvire video-igre.
- **Dizajn video-igre** - potrebno je koncept pretvoriti u dizajn na osnovu kog se kasnije prikupljaju materijali. Za skicu dizajna video-igre korišćeni su besplatni veb alati.
- **Kreiranje sadržaja** - faza u kojoj se prikupljaju ili izrađuju svi materijali neophodni za video-igru. Modeli, teksture i elementi dizajna korisničkog interfejsa. Ova faza je rađena sa posebnom pažnjom kako bi prikupili što kvalitetniji materijali. Veliki deo tekstura planeta je obrađen i dorađivan kako bi sam prikaz planete bio što verodostojniji.
- **Implementacija** - sama implementacija se u početku bazira na apstrahovanju ključnih funkcionalnosti i pravljenje od njih ponovo iskoristive komponente. Sam pristup implementaciji je komponentno-orijentisan. U konkretnoj fazi se pored implementacije funkcionalnosti odvija i spajanje dizajn sadržaja sa komponentama funkcionalnosti.
- **Testiranje** - je faza u kojoj se obavljaju testiranja video-igre, video-igra prolazi kroz više test faza (alfa testiranje, beta testiranje) posle koje je video-igra spremna za zvanično puštanje u promet. Ova faza je uključena u plan razvoja kako bi dala celokupnu sliku ali neće biti razmatrana u daljem radu.
- **Puštanje u promet** - je faza u kojoj je video-igra dostupna svim potencijalnim korisnicima. Kao i faza testiranja ova faza neće biti razmatrana u radu.



Slika 4.1. Prikaz dijagrama faza razvoja video-igre.

4.3 Arhitektura bazirana na komponentama

Koncept komponenti i modularizacije idu zajedno. Grubo rečeno komponenta se može smatrati kao manji deo veće mašine. Svaka komponenta ima svoj specifičan posao i može generalno da ostvari svoj zadatak bez pomoći bilo kog spoljnog izvora.

Okruženje Unity je orijentisano prema komponentama, gde od apstraktnog objekta *(eng. GameObject)* dodavanjem komponenti dobijate finalnu celinu. Pre same implementacije, ključno je bilo apstrahovati i izdvojiti funkcionalnosti u ponovo iskoristive komponente.

4.3.1 Komponentno programiranje

Komponentno programiranje predstavlja pristup zasnovan na definisanju, implementaciji i komponovanja nezavisnih komponenti u labavo spregnute celine, pri čemu jedna komponenta obavlja određenu celinu posla. Ova praksa ima za cilj da dovede do širokog stepena iskorišćenosti komponente. Prilikom kreiranja novog entiteta, funkcionalnosti mu se dodaju preko komponenti koje su često napravljene univerzalno. Entitet u video-igri *(eng. Game Objects)* je u prost kontejner u kom se dodaju neophodne komponente za njegovo funkcionisanje.

4.3.2 Razlike u odnosu na OOP

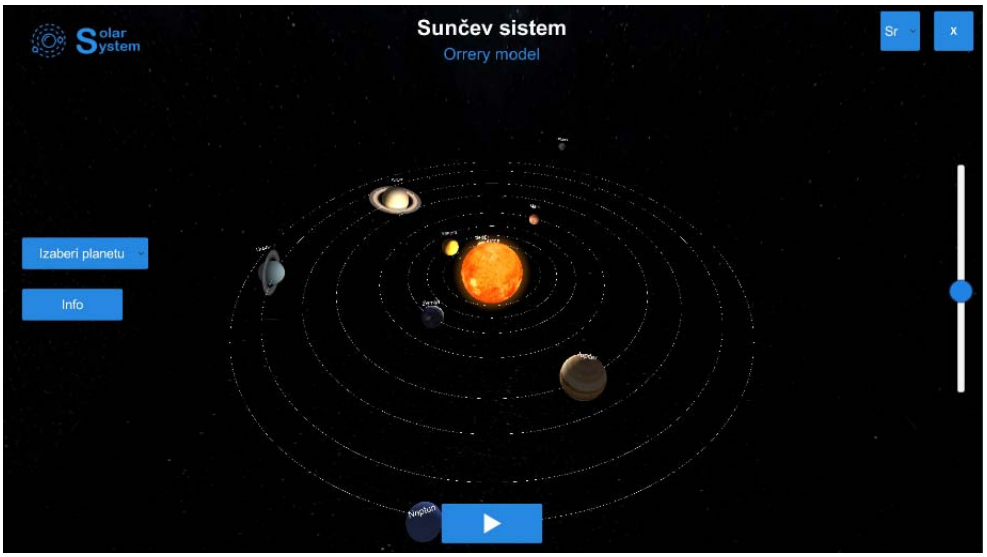
Objektno-orijentisana paradigma se trudi da modeluje objekat nalik na spoljašnji svet. Rešavanje problema objektno orijentisanim pristupom je vrlo slično ljudskom razmišljanju i rešavanju problema. Komponentno orijentisana razvoj softvera za razliku od objektno-orijentisane, nastoji da se softver izgradi od više manjih elemenata, slično kao u oblasti mehanike ili elektrotehnike. Uravo to olakšava lakše održavanje ili menjanje postojeće komponente drugom. Iako postoje razlike komponentno programiranje ne isključuje korišćenje objektno-orijentisanog pristupa, pa se tako u skript komponentama mogu koristiti sve pogodnosti objektno-orijentisanog pristupa.

4.4 Implementacija

U sledećim poglavljima biće opisan razvoj edukativne video-igre *"Solarni Sistem"*. Cilj same video-igre je da približi i dočara Solarni sistem u kome se i mi nalazimo kao sastavni deo. Video-igra je prvenstveno namenjena deci ali i svim zaljubljenicima u astronomiju, pa je pogodna da se koristi kao alat za upoznavanje i sticanje osnovnih saznanja o Solarnom sistemu. U video-igri je prikazano Sunce kao jedina zvezda i centralno telo u našem sistemu, stenovite planete Merkur, Venera, Zemlja i Mars, gasovite planete Jupiter, Saturn, Neptun i Uran kao i od skoro planeta patuljak Pluton. Pored planeta prikazani su i najvažniji prirodni sateliti, Mesec prirodni satelit planete Zemlje, Io, Evropa, Ganimed, Kalisto prirodni sateliti Jupitera, Dion,

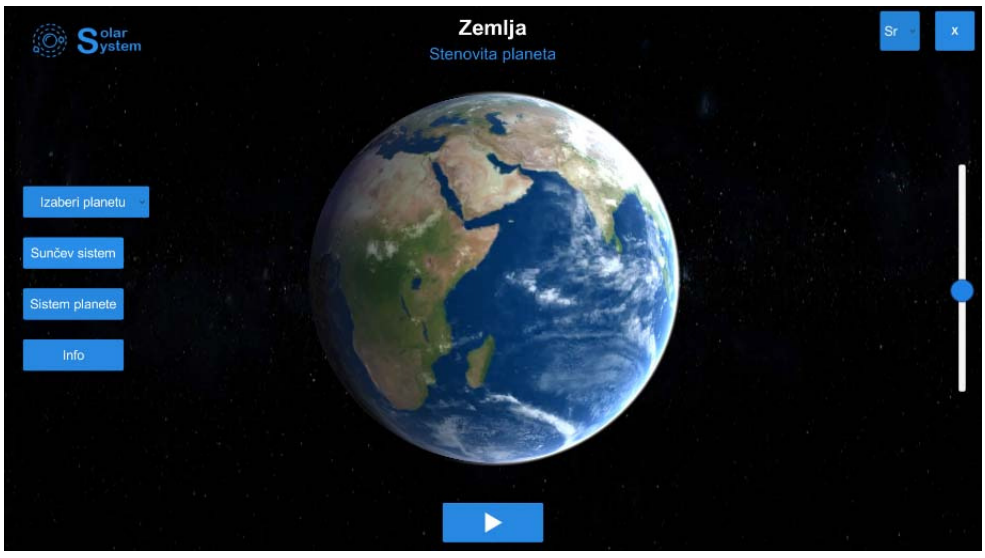
Rea, Iapetus, Titan prirodni sateliti Saturna i Titania prirodan satelit Urana i Triton satelit Neptuna.

"Solarni Sistem" je koncipiran tako da se sastoji iz više scena. Glavna u kojoj je prikazana centralna planeta Sunce i sve planete koje orbitiraju oko nje, postavljene tako da prikazuju realan redosled. Ujedno glavna scena je i interaktivni meni ka izabranoj planeti. Pritiskom na dugme *play* planete orbitiraju oko centralne planete Sunca i sopstvenih osa, brzina orbitiranja i rotiranja simulira približne razlike u brzinama planeta.



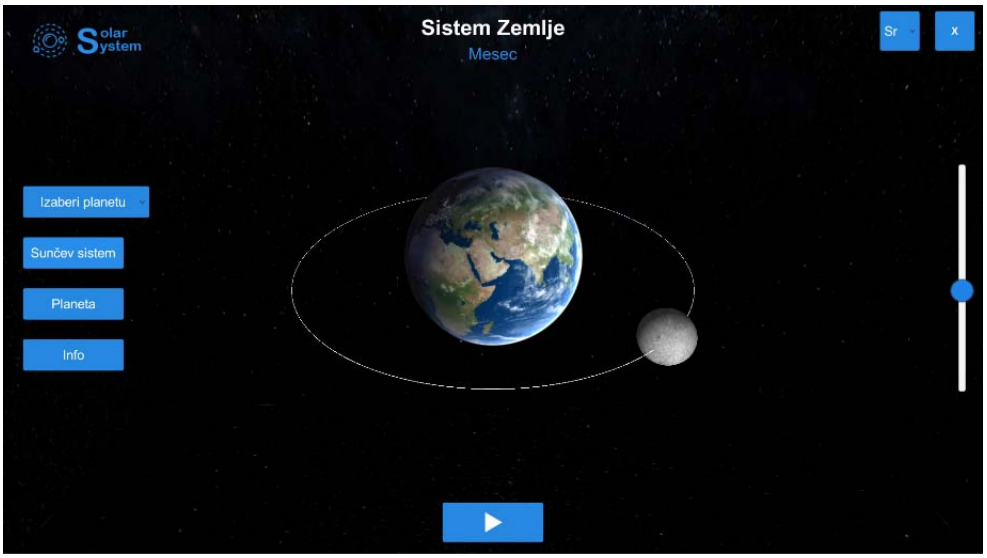
Slika 4.2. Glavna scena u video-igri *"Solarni Sisitem"*.

Prilikom odabira planete prelazi se na scenu izabrane planete gde pomoću dodira ekrana mobilnog uređaja odvija akcija kretanja oko same planete. Pritiskom na dugme *play* planeta rotira oko svoje ose.



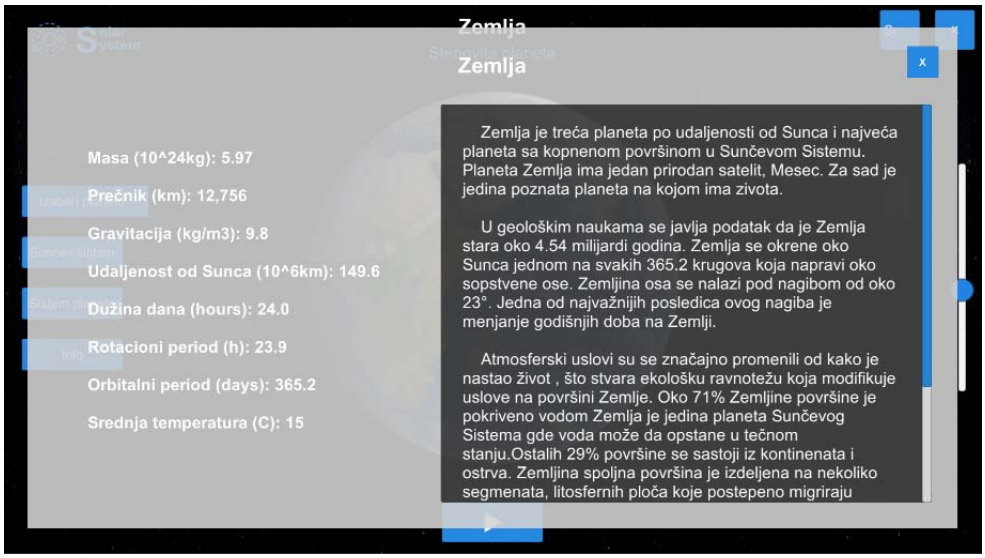
Slika 4.3. Pogled izabrane planete.

Sa planete se može preći na scenu sistema planete, gde se u sceni nalazi planeta i njen prirodni satelit/sateliti, čije se informacije nalaze u info meniju. Pritiskom na dugme *play* planeta rotira oko svoje ose dok prirodni satelit orbitira oko nje i rotira oko sopstvene ose.



Slika 4.4. Pogled sistema izabrane planete.

U svakoj sceni se pomoću pritiska na dugme *info* dolazi do informacija u zavisnosti koja je scena izabrana.



Slika 4.5. Pogled informacija planete.

4.4.1 Osnovni objekti i struktura

Scena (eng. Scene)

Sve što pokreće video-igru se nalazi u sceni ili više njih. Paket za izvršavnje video-igre, izgrađen za neku od platforma u sebi sadrži sve scene i sav zavistan kôd izabrane platforme. Scena je specifičan fajl koji u sebi sadrži sve vrste metapodataka o resursima korišćenim u projektu za tekuću scenu i njena svojstva.

Objekat video-igre (eng. Game Object)

Objekat video-igre je najvažniji koncept u okruženju Unity. Svaki objekat u igri je objekat video-igre, oblikovanjem i dodavanjem komponenti na apstraktni objekat video-igre dobijamo finalan objekat.

Prekonfigurisan objekat (eng. Prefab)

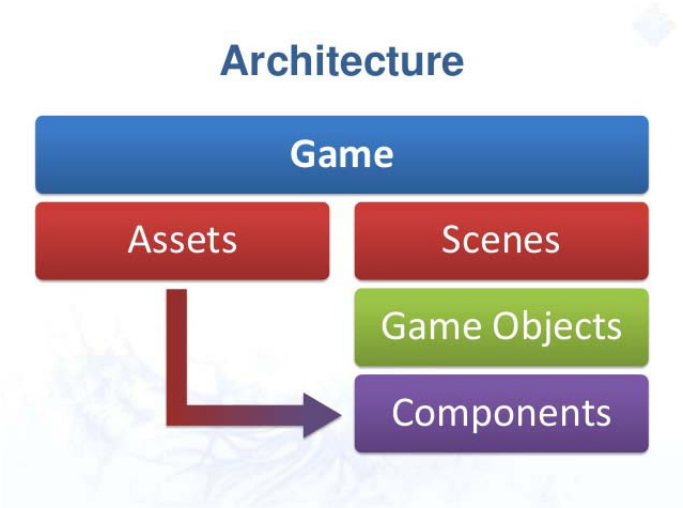
Prekonfigurisani objekti u okruženju Unity su objekti koji su kreirani u sceni, potom su dodate i konfigurisane sve potrebne komponente za taj objekat i nalaze se sačuvani u projektu. Oni mogu da generišu instance ili da se kloniraju u toku video-igre. Prekonfigurisan objekat se ponaša kao šablon iz kog može da se kreira nova instanca objekta u sceni. Svaka izmena nad prekonfigurisanim objektom se automatski reflektuje na sve instance.

Struktura projekta

Unity projekat sadrži Assets, Library, ProjectSettings, and Temp direktorijume, ali samo se jedan prikazuje u Unity editoru, Assets direktorijum.

Library direktorijum je lokalni keš za uvezene sadržaje (eng. assets), sadrži sve metapodatke za sadržaje. Project direktorijum sadrži svu konfiguraciju editora. Temp direktorijum služi za privremene fajlove iz okruženja Mono i okruženja Unity za vreme kreiranja izvršne verzije. Assets direktorijum sadrži sve potrebne komponente za video-igru:

- slike
- 3d modeli
- video sadržaj
- animacije
- fontovi
- zvuk
- skripte



Slika 4.6. Arhitektura video-igre u okruženju Unity.

4.4.2 Priprema okruženja za rad sa Android uređajima

Unity Remote[15] je aplikacija razvijena kako bi olakšala razvoj na mobilnim uređajima, dostupna na Android prodavnici. Aplikacija se povezuje sa okruženjem Unity prilikom izvršavanja video-igre u editoru. Vizuelni sadržaj iz editora se prenosi i na ekran uređaja, a ulazni podaci se prenose nazad u Unity. To omogućava da se dobije pravi utisak o tome kako video-igra izgleda za ciljani uređaj, bez potrebe za pravljenjem izvršnog fajla prilikom svakog testiranja na uređaju.

4.4.3 Kreiranja objekta planete

Objekat video igre planete je apstraktni objekat iz kog su specifične planete nastajale. Prilikom konstrukcije bilo kog objekta, inicijalno kreiranje je prazan objekat video-igre koji je suštinski kontejner za komponente koje želite da dodate. Objekat dolazi sa jednom obaveznom komponentom, komponentom Transformacije *(eng. Transform)* koja određuje poziciju objekta, njegovu rotaciju i mogućnost skaliranja. Neophodna komponenta da bi neki 3D objekat bio vidljiv je komponenta vidljivosti mreže *(eng. Mesh Renderer)* koja određuje kako nešto treba da se nacrta. Komponenta vidljivosti mreže je jedna od retkih zavisnih komponeti, ona zavisi od komponente filtera mreže *(eng. Mesh Filter)* koja oređuje šta treba da se crta.

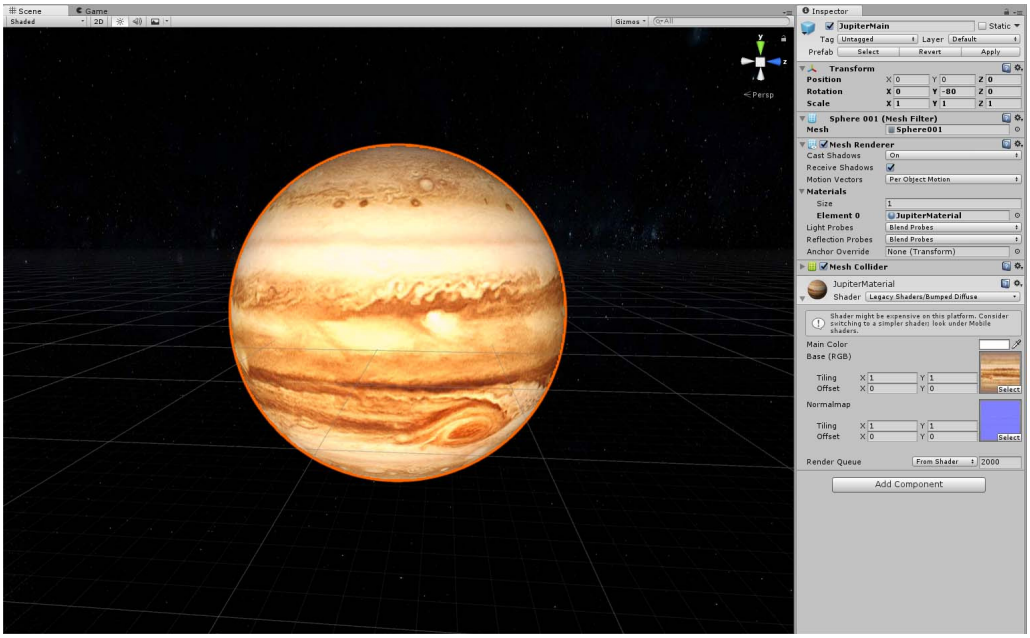
Unity omogućuje kreiranje ponuđenih 3D objekata kao što su: **kocka**, **sfera**, **kapsula**, **cilindar**, **ravan**, **kvadrat**. Kako je filozofija Unity-a takva da on nije alat za modelovanje, kompleksnije geometrije zahtevaju upotrebu nekih dodatnih alata. Model geometrije koji je korišćena za izgradnju planete je preuzeta sa prodavnice komponenti kao besplatna komponenta. Geometrija je sfera sa spljoštenim polovima i za razliku od ponuđene sfere u okruženju Unity boljeg je kvaliteta. Posle dodavanja modela geometrije planeta dobija oblike i strukturu izgleda. Kako bi geometrija dobila izgled neophodno je dodati joj komponentu Materijal.

Materijal *(eng. Material)* igra ključnu ulogu u tome kako će neki objekt da izgeda. Osobine komponente Materijal određuje program za senčenje koji materiajl koristi.

Program za senčenje *(eng. Shader)* je specijalizovana vrsta grafičkog programa koji određuje kako se teksture i informacije o osvetljenju kombinuju kako bi generisali prikaz objekta na ekranu.

Tekstura *(eng. Texture)* je rasterska slika (bit-mapirana slika). Materijal može da sadrži reference na teksture, tako da program da senčenje materijala može da koristi teksture pri izračunavanju boje površine objekta. Pored osnovne boje površine objekta, teksture mogu da predstavljaju i mnoge druge aspekte površine materijala kao što su refleksivnost ili hrapavost. U materijalu se izabira jedan specifičan program za senčenje za upotrebu, izbor programa za senčenje određuje koje su opcije dostupne u materijalu. Program za senčenje specifikuje mogućnost korišćenja jedne ili više očekivanih promenljivih za odabir tekstura, a Inspektor Materijala u okruženju Unity omogućuje da svoje teksture dodelite u te promenljive. Za većinu normalnih renderovanja, pod kojim mislimo na renderovanje karaktera, scenografije, okruženja, čvrstih i providnih objekata, standardni program za senčenje *(eng. Standard Shader)* je obično najbolje rešenje. To je visoko prilagodljiv program za senčenje koji je sposoban da renderuje mnoge tipove površina na vrlo realističan način. Okruženje Unity automatski prilikom izbora programa za senčenje stavlja performanse ispred kvaliteta grafike, pa često ukoliko je izabran program za senčenje skup za hardver na kom se izvršava, menja izabrani nekom optimizovanijom varijantom. Postoji mogućnost ukoliko neki od standardnih programa za senčenje ne odgovaraju vašim potrebama da napišete svoj program za senčenje, to je obično rešenje za neke grafički zahtevnije stvari kao što je tečnost, staklo, ogledalo, čestice, drugi umetnički ili specijalni efekti poput noćnog vida, toplotnog vida ili rengentskog vida.

Svaka planeta poseduje dve teksture, teksturu površine i teksturu reljefa *(eng. Bump map)*.



Slika 4.7. Konfigurisana planeta Jupiter.

4.4.4 Kreiranje skript komponenti

Skripte su suštinski sastojak u video-igrama. Čak i za najjednostavniju video-igru potrebne su skripte, kako bi se odgovorilo na ulazne podatke igrača i organizovali događaji u igri. Skripte se mogu koristiti i za grafičke efekte, fizičko ponašanje objekta ili čak implementacije prilagođenog sistema veštačke inteligencije (eng. AI) za karaktere u igri.

Skripta se povezuje sa unutrašnjim radom okruženja Unity preko klase koju nasleđuje `MonoBehaviour`. Ime klase u skripti se automatski kreira i glasi kao ime fajla skripte, ime fajla i naziv klase moraju da budu isti kako bi skripta mogla da se doda u objekat video-igre. Prilikom kreiranja skript fajla unutar klase se nalaze dve definisane funkcije Start i Update. `Update` je funkcija u kojoj treba da se nalazi sav kôd objekta video-igre koji treba da se ažurira u kadrovima. To može da uključuje kretanje, pokretanje nekih akcija ili odgovor na korisničke ulazne podatke, u suštini sve sa čime treba da se rukuje u mehanici igranja *(eng. Gameplay)*. `Start` funkcija se poziva pre nego što se video-igra pokrene i u njoj se odvija sva potrebna inicijalizacija.

```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour {

    // Start se poziva samo jednom na pocetku
    void Start () {

    }

    // Update se poziva jednom u frejmu
    void Update () {

    }
}
```

Skripta 4.1 Anatomija skript fajla.

Jedna od važnih komponenti za svaku planetu je skripta za rotiranje i translaciju. Skripta omogućuje da planeta orbitira oko objekta koji je izabran da bude centar orbite i rotaciju planete oko sopstvene ose. Samo orbitiranje i rotacija planete zavisi od toga da li je omogućena rotacija, tj. ukoliko je promenljiva `planetRotate` postavljena na tačno. Ukoliko nije izabran objekat oko kog se orbitira, planeta će ukoliko je to omogućeno samo da rotira oko svoje ose. Skripta je implementirana tako da pokriva jednu celinu vezanu za rotiranje i orbitiranje, pa se kao komponenta nalazi u sastavu svakog objekta planete u video-igri.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlanetRotTrans : MonoBehaviour {
    // Izbor ose rotacije planete
    public Vector3 rotateAxis = new Vector3(0, 1, 0);
    public Transform rotateAroundObject;
    public float planetRotationSpeed = 1.0f;
    public float rotationSpeed = 1.0f;
    public bool planetRotate = false;

    void Update () {

        if (rotateAroundObject && planetRotate) {
                // Orbitiranje planete oko izabranog objekta orbitiranja
                transform.RotateAround(rotateAroundObject.transform.position,
                    rotateAxis, rotationSpeed * Time.deltaTime);
        }

        if (planetRotate) {
            // Rotiranje planete oko izabrane ose
            transform.Rotate(new Vector3(0.0f,
                                planetRotationSpeed * Time.deltaTime, 0.0f));
        }
    }
}
```

Skripta 4.2 Skripta za rotaciju i translaciju planete.

4.4.5 Ulazni podaci mobilnih uređaja

Mobilni uređaji imaju svoj specifičan način prihvatanja ulaznih podataka. Za mobilne uređaje klasa *Input* omogućava pristup ulaznim podacima sa:

- **Ekrana osetljivog na dodir** ***(eng. Touchscreen)***
- **Akcelerometra** ***(eng. Accelerometer)***
- **Geografske lokacije** ***(eng. Geographical location)***

Ekran osetljiv na dodir na Android uređajima može da registruje više dodira na ekran simultano, stariji uređaji detektuju 2 dok noviji mogu da detektuju 5 i više dodira. Unity omogućava i simuliranje dodira putem miša, međutim samom prirodom ekrana osetljivog na dodir nemoguće je do detalja simulirati dodire pa je tako jedino moguće detektovanje jednog dodira. Iako Unity omogućava simulaciju dodira putem miša, uvek je bolje koristiti Android uređaj preko aplikacije Unity Remote kako bi se koristili ulazni podaci sa ekrana osetljivog na dodir.

U video-igri koriste se samo ulazni podaci sa ekrana. U skripti *MobileInputController* se nalazi kotrolor za ulazne podatke. Komponenta prepoznaje korisničke ulazne podatke i na osnovu njih izvršava oređene akcije. Prilikom prevlačenja prsta preko ekrana rotira se kamera oko centralnog objekta, zumiranje i odzumiranje sadržaja je moguće sa dva prsta pri čemu se akcija odvija u zavisnosti od pokreta kao i detekcija duplog dodira.

Skript komponenta *MobileInputController* prepoznaje broj dodira na ekran, ukoliko je prepoznala jedan dodir, prepoznaje da li se radi o dodiru ili prevlačenju prsta preko ekrana. Ukoliko je detektovao prevlačenje, ta akcija omogućava da se kamera rotira oko centralnog objekta u sceni, jačina i intenzitet zavise od jačine i intenziteta samog prevlačenja prsta preko ekrana osetljivog na dodir.

```
// Funkcija za rotaciju kamere oko tacke rotiranja
void RotateOnSwipe(Touch t)
{
    float touchTime = 0.0f;
    touchTime += Time.deltaTime;

    if (t.phase == TouchPhase.Began) {
        oldInputPos = new Vector2(t.position.x, t.position.y);
    }

    if (t.phase == TouchPhase.Moved || t.phase == TouchPhase.Stationary) {

        // horizontalna detekcija pravca povlacenja prsta levo/desno
        if (Mathf.Abs(oldInputPos.x - t.position.x) >
                        Mathf.Abs(oldInputPos.y - t.position.y)) {
            float diff = t.position.x - oldInputPos.x;
            if (!naturalMotion) { diff *= -1; }
            if (diff != 0) {
                rotCamParent.transform.Rotate(new Vector3(0, 1, 0) * diff
                        * rotSensitivity, Space.World);
            }
            oldInputPos = t.position;
        }
        // vertikalna detekcija pravca povlacenja prsta gore/dole
        if (Mathf.Abs(oldInputPos.x - t.position.x) <
                        Mathf.Abs(oldInputPos.y - t.position.y))
        {
            float diff = t.position.y - oldInputPos.y;
            if (naturalMotion) { diff *= -1; }
            if (diff != 0)
            {
                rotCamParent.transform.Rotate(0, 0, diff * rotSensitivity);
            }
            oldInputPos = t.position;
        }
```

```
    }

    if (t.phase == TouchPhase.Ended)
    {
        oldInputPos = t.position;
    }

    if (t.phase == TouchPhase.Ended && touchTime < 0.3f) {
        IEnumerator corutine = SingleOrDoubleTouch(t);
        StartCoroutine(corutine);
    }
}
```

Skripta 4.3 Funkcija za rotaciju, deo skript komponente *MobileInputController*.

Ukoliko skript komponenta *MouseInputController* prepozna dva dodira, prati njihovo kretanje i na osnovu toga zumira ili odzumira kameru. Ukoliko korisnik približava prste jedan drugom prilikom dodira kamera se udaljava centru scene, ukoliko korisnik prste udaljava jedan od drugog kamera se približava centru scene.

```
// Funkcija za zumiranje
void PinchZoom(Touch touchZero, Touch touchOne){
    Vector2 touchZeroPrevPos = touchZero.position - touchZero.deltaPosition;
    Vector2 touchOnePrevPos = touchOne.position - touchOne.deltaPosition;

    // Distanca izmedu dodira u kadrovima
    float prevTouchDeltaMag = (touchZeroPrevPos - touchOnePrevPos).magnitude;
    float touchDeltaMag = (touchZero.position - touchOne.position).magnitude;
    float deltaMagnitudeDiff = prevTouchDeltaMag - touchDeltaMag;

    camera.fieldOfView += deltaMagnitudeDiff * zoomSpeed;
    camera.fieldOfView = Mathf.Clamp(camera.fieldOfView, 20f, 85f);
    zoomSlider.value = Mathf.Clamp(camera.fieldOfView, 20f, 85f);
}
```

Skripta 4.4 Funkcija za zumiranje/odzumiranje, deo skript komponente *MobileInputController*.

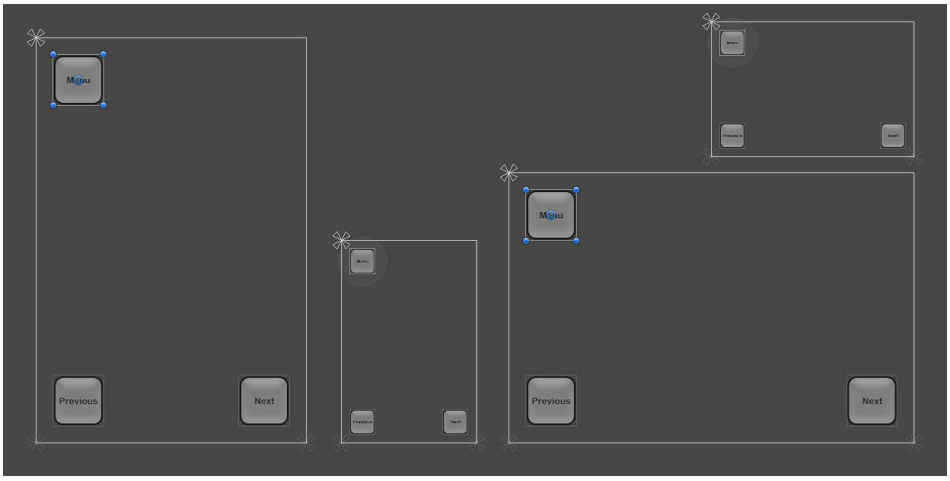
4.4.6 Prilagodljiv korisnički interfejs

Centralni i koreni objekat u sistemu korisničkog interfejsa je **Canvas**, svi elementi korisničkog interfejsa se kao deca objekti nalaze u njemu. Elementi korisničkog interfejsa su nacrtani po hijerarhijskom redosledu, prvo dete je nacrtano prvo, drugo sledeće i tako dalje. Ako se dva elementa preklapaju, redosled određuje koji element će pokriti ovog drugog.

Objekat Canvas poseduje Render Mode podešavanja koja određuju način renderovanja elemenata u prostoru ekrana *(eng. Screen space)* ili prostoru sveta *(endg. World space)*. Razlika iđu ova dva prostora što je prostor ekrana reprezentovan dvema osama X i Y, dok je prostor sveta prostor koji predstavlja realan prostor i predstavljen je sa tri ose X, Y, Z.

- **Prostor ekrana - prekriven** - način renderovanja gde se elementi korisničkog interfejsa prikazuju na vrhu scene. Ako ekran promeni veličinu ili rezoluciju, Canvas će automatski da promeni veličinu kako bi se poklapao.
- **Prostor ekrana - kamera** - slično prvom načinu, ali u ovom režimu renderovanja Canvas je postavljen na određenoj distanci u odnosu na specifičnu kameru. Elemente korisničkog interfejsa renderuje kamera, što znači da njena podešavanja utiču na izgled elemenata.
- **Prostor sveta** - u ovom režimu renderovanja, Canvas će se ponašati kao bilo koji drugi objekat u sceni.

Svi elementi umesto standardne komponente Transform imaju komponentu **Rect Transform**, koja poseduje poziciju, rotaciju, skaliranje elementa i dodatne parametre za visinu i širinu elementa. Video-igre namenjene za Andorid uređaje često moraju da podrže širok spektar različitih rezolucija ekrana pa je prilagodljiv korisnički interfejs obavezan. Parametar odgovoran za pozicioniranje elementa na ekranu je **sidro** *(eng. Anchor)*, prilikom kreiranja elementa on se uvek nalazi u sredini ekrana. Ukoliko je sidro elementa postavljeno u sredini ekrana i element je konfigurisan prema ekranu u vertikalnoj orijentaciji, prilikom promene orijentacije na horizontalnu može se desiti da se element ne vidi na ekranu, jer je dužina od sidra do njegove pozicije veća nego što je visina ekrana. Upravo zbog toga način postavljanja sidra omogućuje elementu da ne zavisi od širine i visine ekrana. Prilikom postavke elemenata najbolje je njegovo sidro vezati za najbliži košak ekrana, čime će element nezavisno od veličine ekrana uvek biti postavljen na istoj udaljenosti. Konfiguracija komponente **Canvas Scaler** je jako bitna, kako bi se veličina elementa smanjivala ili povećavala u odnosu na promenu ekrana za određeni procenat. U polju izbora vrste skaliranja *(eng. UI Scale Mode)* potrebno je izabrati opciju skaliranja prema veličini ekrana *(eng. Scale With Screen Size)*.



Slika 4.8. Prikaz prilagodljivog interfejsa prilikom promene orijentacije i rezolucija ekrana.

4.4.7 Višejezičnost

Višejezičnost je jako važna kako bi video-igra dostigla što bolje domete, ujedno i podigla kvalitet video-igre. U video-igri *"Solarni Sistem"* je implementiran srpski i engleski jezik. Tehnika implementacije koja se koristi se pokazala kao jako dobra i često je praksa prilikom implementacije u klasičnim Android aplikacijama. Sama ideja je da se sadržaj jezika čuva u XML fajlu, koji se parsira i čuva u strukturi ključ-vrednost. Parsiraje obavlja klasa `Language` koju poziva klasa `LanguageManager` Pomoću skript komponente *TextController* koja je vezana za tekstualni objekat interfejsa, učitavamo željeni deo teksta. Prilikom izmene jezika automatski se ažurira sadržaj i prikaz teksta.

Preference korisnika

Preference korisnika su najlakši način za čuvanje informacije korisnika između njegovih sesija. Preference korisnika *(eng. PlayerPrefs)* je klasa iz osnovne biblioteke u kojoj mogu da se čuvaju ključ-vrednost informacije. U video-igri se koristi kako bi se sačuvao korisnikov izabrani jezik i podešavanja.

```
<?xml version="1.0" encoding="utf-8"?>
<languages>
<En>
  <!-- Imena planeta -->
  <string id="sun">Sun</string>
  <string id="mercury">Mercury</string>
  <string id="venus">Venus</string>
  ...

  <!-- Oznake -->
  <string id="solar_label_name">
    <![CDATA[<b>Solar system</b> ]]>
    <![CDATA[<size=32><color=#288eea>Orrery model</color></size>]]>
  </string>
  <string id="sun_label_name">
    <![CDATA[<b>Sun</b>]]>&#xD;
<![CDATA[<size=32><color=#288eea>Star</color></size>]]>
  </string>
  ...

</En>
<Sr>
  <!-- Imena planeta -->
  <string id="sun">Sunce</string>
  <string id="mercury">Merkur</string>
  <string id="venus">Venera</string>
  ...
```

```
    <!-- Oznake -->
  <string id="solar_label_name">
    <![CDATA[<b>Sunčev sistem</b> ]]>
    <![CDATA[<size=32><color=#288eea>Orrery model</color></size>]]>
  </string>
  <string id="sun_label_name">
    <![CDATA[<b>Sunce</b>]]>&#xD;
    <![CDATA[<size=32><color=#288eea>Zvezda</color></size>]]>
  </string>
  ...

</Sr>
</languages>
```

XML 4.1 Deo izvornog koda fajl lang.xml koji sadrži tekst na engleskom i srpskom.

Poziv skript komponente *Language* se odvija prilikom pokretanja video-igre ili promene jezika i nju poziva izključivo skript komponenta *LanguageManager*. Skript komponenta *LanguageManager* poziva sa parametrom jezika skirpt komponentu *Language* koji je prilikom pokretanja video-igre izabrani jezik iz preference korisnika ili novoizabrani jezik iz opadajućeg menija za jezike na ekranu. Kako bi skirpt komponenta *Language* konstruisala izabrani jezik, učitava XML fajl koji sadrži jezike. Skripta prolazi kroz elemente jezika u XML fajlu i na osnovu njih konstruiše strukturu ključ-vrednost, gde je ključ identifikator te rečenice u elementu a vrednost tekst u elementu.

```
using System.Collections;
using UnityEngine;
using System.Xml;

public class Language  {

    private Hashtable Strings;

    public Language (string userLanguage) {
        setLanguage(userLanguage);
    }

    public void setLanguage(string language) {
        // ucitavanje xml fajla
        TextAsset textAsset = (TextAsset)Resources
                                    .Load("lang", typeof(TextAsset));
        var xml = new XmlDocument();
        xml.LoadXml(textAsset.text);
        Strings = new Hashtable();
```

```
        var element = xml.DocumentElement[language];
        if (element != null)
        {
            var elemEnum = element.GetEnumerator();
            while (elemEnum.MoveNext())
            {
                if (!(elemEnum.Current is  XmlComment)) {
                    var xmlItem = (XmlElement)elemEnum.Current;
                    Strings.Add(xmlItem.GetAttribute("id"), xmlItem.InnerText);
                }
            }
        }
    }

    public string getString(string name)
    {
        if (!Strings.ContainsKey(name))
        {
            return "";
        }

        return (string)Strings[name];
    }
}
```

Skripta 4.5 *Language* skript komponenta za ucitavanje željenog jezika iz XML fajla.

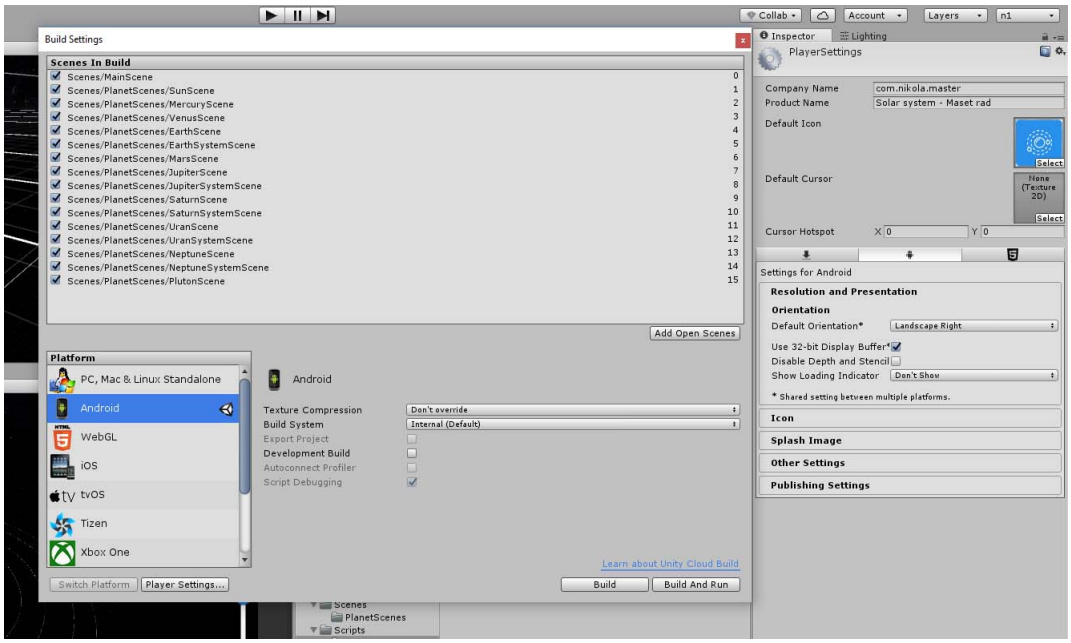
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class TextController : MonoBehaviour {
    private Text uiTxt;
    private Dropdown dropdown;
    private LanguageManager languageSelect;
    public Language language;
    public string getString;

    // Inicijalizacija teksta u izabranom jeziku
    void Start() {
        dropdown = GameObject.Find("LanguageDropdown")
                             .GetComponent<Dropdown>();
        languageSelect = GameObject.Find("LanguageDropdown")
                                   .GetComponent<LanguageManager>();
```

```
        uiTxt = gameObject.GetComponent<Text>();
        language = languageSelect.languageStruct;
        uiTxt.text = language.getString(getString);

        // Dodavanje funkcije za detektovanje promene jezika
        dropdown.onValueChanged.AddListener(delegate
        {
            UpdateLanguage();
        });
    }

    // Ažuriranje teksta prilikom reizbora jezika
    public void UpdateLanguage() {
        language = languageSelect.languageStruct;
        uiTxt.text = language.getString(getString);
    }
}
```

Skripta 4.6 *TextController* skript komponenta za odabir dela teksta iz jezika.

4.4.8 Kreiranje izvršne verzije

Kreiranje izvršnog fajla video-igre u okruženju Unity se odvija u prozoru za kreiranje izvršnog fajla *(eng. Build Settings)*, gde se odabira platforma za koju se kreira izvršni fajl. Pored prozora za kreiranje izvršnih fajlova, bitan je prozor postavke igrača prema platformi *(eng. Player Settings)*. U ovom prozoru se prikazuju postavke igrača specifične za Android platformu. Prozor postavke je sastavljen iz četri dela, deo za rezoluciju i prezentaciju gde se odabira orijentacija ekrana, deo za povezivanje ikonice za video-igru sa potrebnim rezolucijama za Android uređaje, deo za podešavanje početnog ekrana, deo za podešavanje načina crtanja *(eng. rendering)* i identifikacije video-igre kao i minimalno potrebne verzije Android sistema i dela za postavku objavljivanja gde je moguće uneti kriptografski ključ kako bi se video-igra dodatno zaštitila prilikom publikacije.



Slika 4.9. Prikaz prozora za kreiranje izvršnog fajla i postavke igrača.

U Unity-u je moguće napraviti izvršni fajl sa sufiksom .apk ili eksportovanje celokupnog projekta. Ukoliko je video-igra veća od dozvoljenih 100MB na android prodavnici, moguće je podeliti u manje pakete sa opcijom binarne podele *(eng. Split Application Binary)*.

Ukoliko teksture nisu pripremljene za Android platformu moguće je izabrati željenu kompresiju tekstura. Postoje dve opcije kreiranja fajla, opcija za kreiranje i pokretanje *(eng. Build and Run)* izvršnog fajla na povezanom uređaju ili samo opcija kreniranja *(eng. Build)* izvršnog fajla.

4.4.9 Razvoj video-igara za razne platforme - WebGl

Razvoj video-igara za razne platforme je jedna od najjačih karakteristika okruženja Unity, pa nudi mogućnost da video-igru razvijete i izvršavate na više platformi kako bi osigurali što bolju pokrivenost. Fundametalne razlike u hardveru zahtevaju da se za neke platforme video-igra prilagodi njima. Ukoliko video-igru razvijenu za sistem Android želimo da pokrenemo u veb pretraživaču, moramo da promenimo način prihvatanja ulaznih podataka zbog hardvera koji se razlikuje, ekran osetljiv na dodir sa jedne strane i tastatura i miš sa druge. Prilokom multiplatformisanja potrebno je uzeti u ozir memoriju i jačinu hardvera, pa bi tako na primer bilo jako teško zahtevnu video-igru razvijenu za desktop računare, prilagodili za rad na mobilnim uređajima upravo zbog razlike u jačini hardvera.

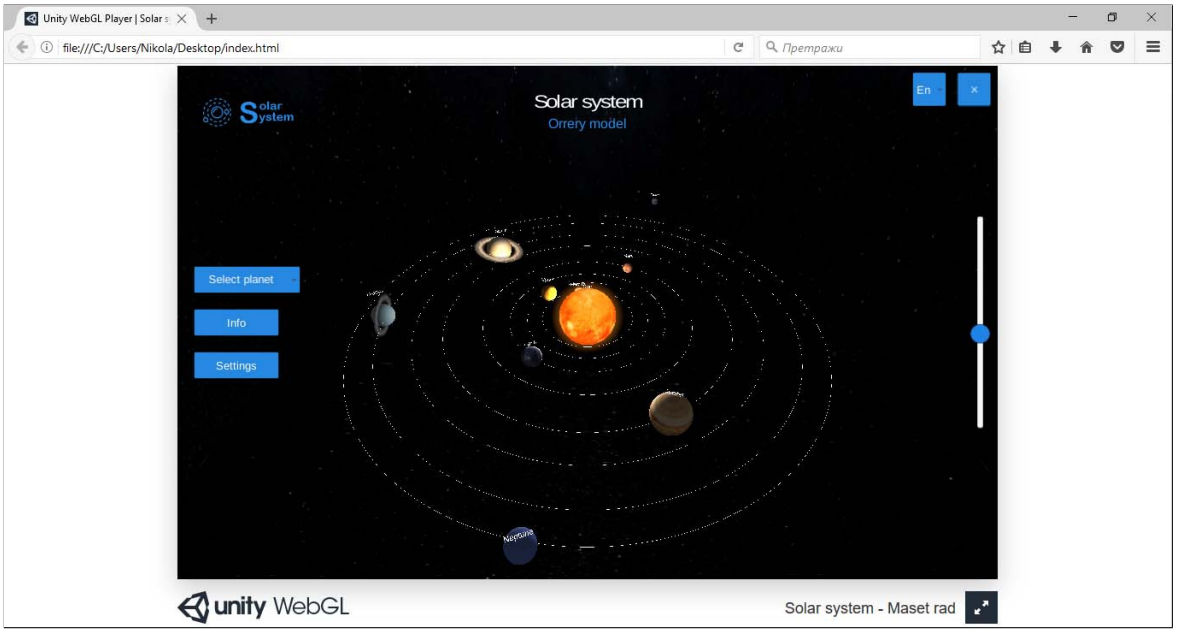
Jako bitna stvar koju okruženje Unity nudi je **kompilacija prema platformi**. Kompilacija prema platformi je funcija koja uz pomoć preprocesorskih direktiva omogućava da skripte podelite i izvršite deo koda isključivo za jednu od podržanih platformi.

```
using UnityEngine;
using System.Collections;
```

```
public class PlatformDefines : MonoBehaviour {

  void Start () {
    #if UNITY_ANDROID
      // deo koji se izvrsava za sistem Android
    #endif

    #if UNITY_WEBGL
        // deo koda koji se izvrsava za WebGl
    #endif
  }
}
```

Skripta 4.7 Primer preprocesorskih direktiva za izvršavanje koda za specifičnu platformu unutar skripte.

Prilikom pripremanja materijala za WebGl verziju, nije bilo potrebe da se teksture kompresuju kao za sistem Android zbog jačeg hardvera desktop računara. Opcija za WebGl kreiranje video-igre omogućava da se sadržaj pripremi u JavaScript programe koji koriste HTML5 tehnologije i WebGL API kako bi se video-igra izvršavala u veb pregledaču.



Slika 4.10. Prikaz video-igre u internet pregledaču.

Glava 5

Zaključak

Ovaj rad je koncipiran tako da uvede čitaoca u osnove pri razvoju 3D video-igre za sistem Android pomoću okruženja Unity. Okruženje Unity omogućava da se sve ono na čemu se radi vidi u realnom vremenu, testiranje video-igre preko prozora za pokretanje, mogućnost izmena u realnom vremenu i pregleda kako se izmene reflektuju. Moć koju okruženje Unity daje programeru je ogromna, što je po autorovom mišljenju verovatno suštinski aspekt savremenog razvoja video-igara. Sve to omogućava arhitektura zasnova na komponentama okruženja Unity. Sama kriva učenja u okruženju Unity je blaga pa je pogodno za programere sa manje iskustva ili programere početnike.

U radu se prikazuju osnove razvoja video-igara pomoću okruženja Unity zbog čega rad može da posluži kao dobra osnova za dalji proces učenja. U radu su prikazane i neke naprednije tehnike kao što je tehinka višejezičnosti i način kreiranja korisničkog interfejsa kako bi se pokrio širok spektar rezolucija ekrana.

Video-igra *"Solarni Sistem"* razvijena kao demonstracija funkcionalnosti okruženja Unity. Može da se koristi za edukaciju dece kao upoznavanje sa sunčevim sistemom jer ako slika govori više od hiljadu reči onda simulacija govori mnogo više.

Literatura

[1] Android
https://www.android.com/

[2] Android (operating system)
https://en.wikipedia.org/wiki/Android_(operating_system)

[3] Android developers
https://developer.android.com/index.html

[4] Open Handset Alliance
http://www.openhandsetalliance.com/

[5] Unity3d https://unity3d.com/

[6] Unity (game engine)
https://en.wikipedia.org/wiki/Unity_(game_engine)

[7] Unity 5.x Cookbook, Matt Smith.

[8] Unity 5 for Android Essentials, Valera Cogut.

[9] Beginning 3D Game Development with Unity, Sue Blackman.

[10] Smartphone market share
https://www.idc.com/prodserv/smartphone-os-market-share.jsp

[11] Game industry revenue
https://venturebeat.com/2016/12/21/worldwide-game-industry-hits-91-billion-in-r

[12] NASA planets
https://solarsystem.nasa.gov/planets/

[13] Unity3D Programming
https://www.slideshare.net/sasmaster/unity3d-programming-5725801

[14] Unity Manual
https://docs.unity3d.com/Manual/

[15] Unity Remote
https://play.google.com/store/apps/details?id=com.unity3d.genericremote

[16] Unity global game industry
https://unity3d.com/public-relations

[17] Video game
https://en.wikipedia.org/wiki/Video_game

[18] Video games genres
https://en.wikipedia.org/wiki/List_of_video_game_genres

[19] Earth 3D model object
https://www.assetstore.unity3d.com/en/#!/content/54914

[20] Agile Methodology
http://agilemethodology.org/