УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: **Александар Стефановић**
Број индекса: **e12668**

Тема рада: **Једно решење Андроид апликације са подршком за више додира**

Ментор рада: **Пап др. Иштван**

Нови Сад, месец, година

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Завршни (Bachelor) рад	
Аутор, **АУ**:		Александар Стефановић	
Ментор, **МН**:		Пап др. Иштван	
Наслов рада, **НР**:		Једно решење Андроид апликације са подршком за више додира.	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2012	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/24/0/0/4/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		Андроид апликација, Игра	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		У овом раду је описана једна реализација мултитач Android aplikacije u vidu igre.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	др Јелена Ковачевић, доц.	
	Члан:	др Небојша Пјевалица, доц.	Потпис ментора
	Члан, ментор:	др Иштван Пап, доц.	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Bachelor Thesis	
Author, **AU**:		Aleksandar Stefanović	
Mentor, **MN**:		PhD Pap Ištvan	
Title, **TI**:		One solution of Android mutitouch application.	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2012	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/24/0/0/4/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Android application, Game	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper presents a solution of example Android multitouch application in form of a simple game.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Jelena Kovačević, PhD	
	Member:	Nebojša Pjevalica, PhD	Menthor's sign
	Member, Mentor:	Ištvan Pap, PhD	

SADRŽAJ

SPISAK SLIKA

SKRAĆENICE

OpenGL - *Open Graphics Library,* biblioteka za podršku u grafici.

API - *Aplication Programing Interface,* programska sprega.

1. Uvod

U ovom radu je realizovan jedan primer Android multitouch aplikacije u vidu igre. Aplikacija je realizovana za pametne telefone i tablet računare sa android operativnim sistemom.

U okviru rada je dat pregled kreiranja programske podrške za realizaciju aplikacije (eng. Engine) kao i njena realizacija.

Ovaj rad je sačinjen od sedam poglavlja.

- U prvm poglavlju je dat kratak opis rada.
- Drugo poglavlje daje teorijske osnove Android aktivnosti, kao i kratak opis OpenGL programske podrške za grafiku.
- U trećem delu su opisane ključne komponente ovog rada.
- Četvrto poglavlje sadrži detaljniji opis realizacije aplikacije kao i kratak opis svih paketa i klasa koji čine aplikaciju.
- U petom poglavlju je opisan način ispitivanja i verifikacije glavnih modula aplikacjie.
- Šesto poglavlje sadrži kratak pregled onoga što je urađeno u radu i kakvi su dalji pravci razvoja.
- U sedmom delu je dat spisak korišćene literature za izradu rada.

2. Teorijske osnove

U ovom poglavju dat je kratak opis Android komponente, aktivnosti, kao i opis grafičke sprege namenjene za ugrađene sisteme (OpenGL ES).

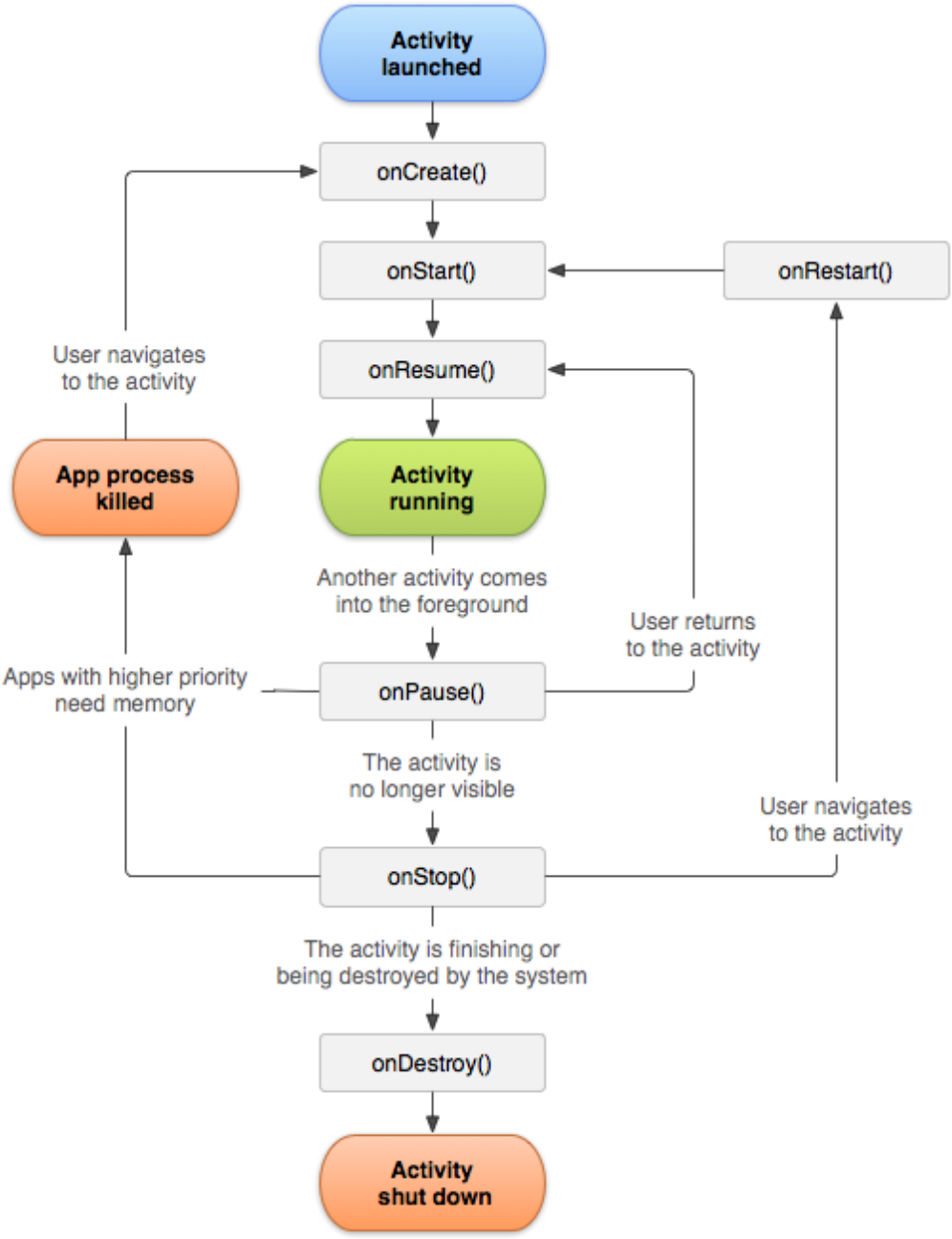
2.1 Android aktivnost

Aktivnosti su sastavni deo android operativnog sistema, i koriste se u izradi aplikacija. Android aplikacija može da se sastoji od više aktivnosti i aktivnosti se mogu prebacivati jedna sa druge u toku izvršavanja aplikacije. Aktivnosti se slažu u vidu magazinske memorije (steka). Kada je nova aktivnost pokrenuta ona se postavlja na vrh magazinske memorije i postaje aktivna, prethodna aktivnost uvek ostaje ispod nove, i neće biti aktivna sve dok se trenutno aktivna ne završi.

Aktivnost ima četiri stanja:

- Ako je aktivnost vidljiva na ekranu (na vrhu magazinske memorije) ona je aktivna odnosno u pokrenutom stanju.
- Ako je aktivnost izgubila fokus ali je i dalje vidljiva (u slučaju da nova aktivnost koja nije veličine celog ekrana ima fokus) ona je pauzirana. Pauzirana aktivnost je i dalje živa (i dalje čuva podatke o stanju i informacijama o članovima i ostaje zakačena sa menadzer prozora), ali može biti ugašena od strane sistema u slučaju ekstremno malo raspoložive memorije.
- Ako je aktivnost u potpunosti pokrivena nekom drugom aktivnošću ona je zaustavljena. Ona i dalje čuva sve svoje podatke ali nije više vidljiva na ekranu i često može biti završena od strane sistena ako je njena memorija potrebna negde drugde.
- Ako je aktivnost pauzirana ili zaustavljena, sistem može izbaciti aktivnost iz memorije zahtevajući od aktivnosti da se završi, ili je nasilno završiti. Ako je ta

aktivnost posle ponovo prikazana korisniku ona se mora u potpunosti restartovati i vratiti u prethodno stanje.



Slika 2.1 Životni ciklus aktivnosti.

2.2 OpenGL ES

OpenGL ES je deo OpenGL grafičke aplikativne programske sprege namenjen za ugrađene sisteme kao što su mobilni telefoni i konzole za video igre. Predstavlja podršku za 2D i 3D grafiku visokih performansi. OpenGL ES 1.0 i 1.1 su podržani još od verzije Androida 1.0. Počevši od Android verzije 2.0 (API nivo 8) podržan je Opengl ES 2.0.

Postoje dve fundamentalne kalse u android okruženju koje omogućavaju kreiranje i manipulisanje grafikom pomoću OpenGL ES API:

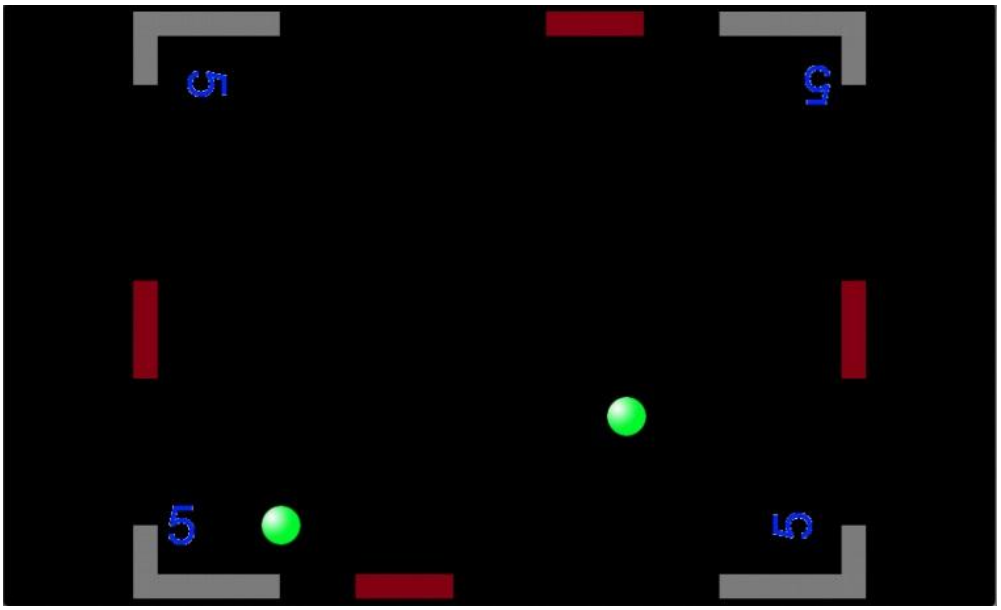
- GLSurfaceView - Pomoću ove klase se mogu iscrtavati i manipulisati objekti upotrebom OpenGL API poziva, i slična je po funkciji klasi SurfaceView. Klasa se može koristiti tako što joj se nakon kreiranja dodaje klasa Renderer, ili se može proširiti.
- GLSurfaceView.Renderer - Ova sprega definiše metode potrebne za iscrtavanje grafike u GLSurfaceView klasi. Potrebno je obezbediti implementaciju ove sprege kao posebnu klasu i dodati je primerku GLSurfaceView klasi.

3. Koncept rešenja

U ovom poglavlju su opisani detalji igre kao i njena pravila. Takođe opisane su ključne komponente aplikacije kao što su glavna petlja igre, sistem simulacije sudara dvodimenzionalnih objekata, fizika pomeranja objekata.

3.1 Opis aplikacije

U igri učestvuje četiri igrača, svaki od igrača upravlja jednim objektom. Svaki igrač ima svoju stranu (stranu ekrana) koju čuva. Na terenu (ekranu) se nalaze loptice koje inicijalno kreću iz nasumičnog ćoška ekrana, i svaki igrač svojim objektom mora da čuva svoju stranu, tj ne sme da dozvoli da loptica prođe van njegove strane ekrana.



Slika 3.1 Prikaz igre i rasporeda elemenata.

Ukoliko loptica prođe, igrač gubi život. Svaki igrač ima određen broj života na početku igre i pobednik je onaj koji poslednji ostane živ (ima broj života veći od nule). Kada neki od igrača izgubi sve svoje živote, njegova strana ekrana se zatvara, kako loptice nebi i dalje ispadale sa te strane ekrana. U početku igre na terenu postoje dve loptice, nakon izvesnog vremena broj loptica se povećava do maksimalnog broja loptica koji je 10.

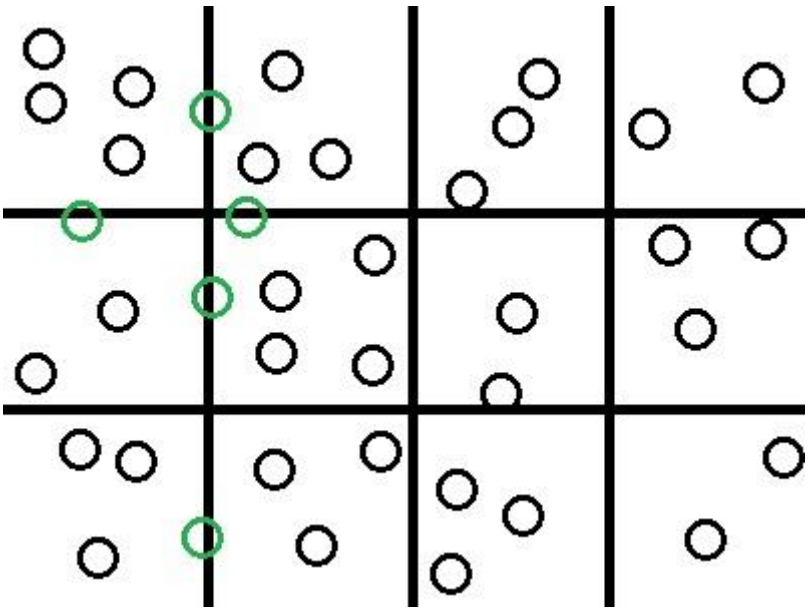
3.2 Glavna petlja

Aplikacije se sastoji iz jedne aktivnosti koja sadrži glavnu petlju igre. U svakoj iteraciji glavne petlje se posebno osvežavaju svi podaci igre, kao što su položaji loptica, detekcija sudara i njihovo razrešavanje, obrada ulaza. Pored ovoga se takođe u svakoj iteraciji ponovo iscrtavaju svi objekti na osnovu prethodno osveženih podataka.

3.3 Sistem sudaranja

Sistem sudaranja obuhvata detekciju sudara, i njegovu obradu. Objekti koji se obrađuju mogu biti raznih oblika, ali oblici koji se obrađuju prilikom detekcije i obrade sudara (kako bi sama detekcija bila jednostavnija i brža) su pravougaoni i okrugli. Svaki objekat u aplikaciji pored brojnih svojih podataka, koji će biti opisani u daljem tekstu, sadrže svoj ograničavajući oblik (eng. bounding volume) koji može biti pravougaonik ili krug.

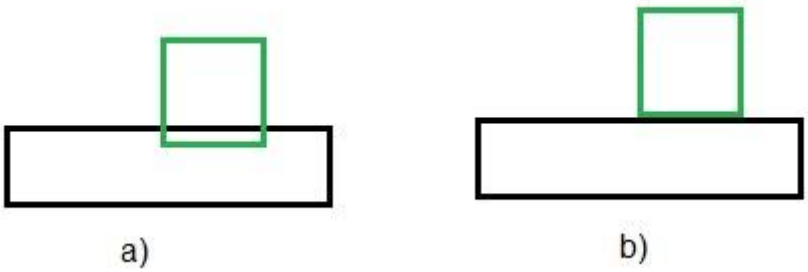
Kako bi se ubrzala sama detekcija sudara koristi se prostorna raspodela. Pod ovim se podrazumeva da je teren (ekran) izdeljen u vidu rešetke na NxM delova. Svaki pravougaonik dobijen ovim deljenjem predstavljen je jednom listom, tako da imamo NxM listi koje sadrže objekte. Na osnovu položaja objekta određuje se identifikator liste u koju će on biti dodat. Ako se objekat nalazi na granici između dva ili više susedna dela mape, on će biti dodat u sve liste preko kojih se preklapa.



Slika 3.2 Prikaz preklapanja objekata preko susednih podeoka.

U svakoj iteraciji glavne petlje ove liste se osvežavaju, i na taj način objekti koji se kreću bivaju prebačeni iz jedne u drugu listu. Pošto postoje dinamički (loptice) i statički (zidovi) objekti, prilikom osvežavanja se osvežavaju samo liste koje sadrže dinamičke objekte. Na ovaj način se izbegava nepotreban utrošak vremena procesora. Prilikom detekcije sudara međusobno se proverava postojanje preseka u ograničavajućim oblicima objekata koji se nalaze u istoj listi. Posebno se prvo proveravaju međusobno dinamički objekti, svaki sa svakim, a nakon toga dinamički sa statičkim iz odgovarajućeg prostora mape. Statički objekti se međusobno ne proveravaju.

Što se tiče rešavanja sudara potrebno je objekat vratiti unazad tako da prilikom njihovih iscrtavanja nema međusobnih preseka. Objekat se vraća unazad suprotno od njegovog pravca kretanja.



Slika 3.3 Prikaz rešavanja sudara. a) detektovan sudar, b) rešen sudar.

3.4 Objekti

U aplikaciji postoje statički i dinamički objekti. Pošto su dinamika sudaranja i kretanje loptica ključni deo aplikacije, potrebno je što realnije simulirati njihovo ponašanje.

Glavne osobine koje objekat poseduje su:

- pozicija
- brzina
- ubrzanje
- sila koja deluje na njega
- inverzna vrednost mase
- normala sudara
- ograničavajući oblik

Na osnovu ovih osobina se svake iteracije izračunava nov položaj objekta. Umesto stvarne vrednosti mase koristi se inverzna vrednost kako bi se u simulaciji lakse mogli predstaviti objekti beskonačne mase (kao što su statički objekti).

Normala sudara se računa u slučaju sudara okruglog objekta sa pravougaonim i koristi se za određivanje novog pravca loptica nakon sudara (loptice se odbijaju po principu refleksije).

Kao što je već pomenuto, ograničavajući oblik se koristi u detekciji sudara, i njegov položaj je isti sa položajem objekta kom pripada. Služi kako bi se pojednostavila detekcija sudara objekata kompleksnog oblika.

3.5 2D grafika

Za prikaz 2d slika odnosno grafike koristi se GLSurfaceView. Ova Android klasa predstavlja implementaciju SurfaceView klase koja koristi predviđenu površinu za iscrtavanje grafike korišćenjem OpenGL-a. Slike se iscrtavaju tako što se na poligon dodaje tekstura.

4. Programsko rešenje

Programsko rešenje se svodi na implementaciju glavne bazne funkcionalnosti (eng. engine), koja obuhvata iscrtavanje 2D grafike, detekciju sudara, glavnu petlju, obradu ulaza (dodir i tastatura), fiziku interakcije objekata. Upotrebom ove funkcionalnosti je implementirana igra.

4.1 Moduli engine-a

Implementacija je uradjena u javi pa su moduli organizovani u pakete. Engine se sastoji od sledećih paketa:

- bsc.engine - sadrži klase koje su u stvari interfejsi.
- bsc.engine.gl - sadrži klase koje koriste OpenGL.
- bsc.engine.impl - sadrži implementaciju interfejsa i glavne klase koje implementiraju glavnu petlju alikacije.
- bsc.engine.math - sadrži klasu za računske operacije sa 2d vektorima.
- bsc.engine.objs - sadrži klase objekata.
- bsc.engine.physics - sadrži klase koje su zadužene za detekciju i obradu sudara.
- bsc.engine.test - sadrži jednostavnu demonstraciju funkcionalnosti.
- bsc.engine.ui - sadrži klasu koja predstavlja dugme, koje moze da se aktivira pritiskom.

4.1.1 Paket bsc.engine

Ovaj paket sadrži klase koje su interfejsi, koji su implementirani u paketu bsc.engine.impl. Sadržane klase su:

- Game - sadrži get i set metode koje rukuju sa interfejsima iz ovog paketa, i služe za pribavljanje nekih komponenti kao što su ulaz, scena (predstavlja skup

komponenti koje su prikazane u okviru jedne scene na ekranu, odgovara klasi Screen).

- Input - sadrži funkcije koje barataju prihvatanjem ulaza.
- Screen - sadrži referencu na klasu Game, i metode zadužene za osvežavanje scene, tj svih objekata koji joj pripadaju, i metodu zaduženu za njihovo iscrtavanje.
- Pool - ovo je apstraktna klasa i služi za čuvanje objekata. Ova klasa predstavlja jedan vid optimizacije, jer se prilikom stvaranja i uništavanja objekata gubi procesorsko vreme. Objekti se instanciraju jedanput i nakon završetka njihove upotrebe smeštaju se u Pool klasu odgovarajućeg tipa, kako bi se prilikom potrebe kreiranja novog objekta preuzeo stari objekat, čije su vrednosti već pročitane i nisu više potrbne (informacija je zastarela ili već obrađena), i popunjavaju se njegova polja novom informacijom. Primer upotrebe ove klase je prilikom obrade ulaza, pri pritisku tastera ili ekrana, ne kreiraju se novi objekti već se stari samo menjaju i ponovo obrađuju.
- FileIO - sadrži funkcije za rukovanje datotekama.

4.1.2 Paket bsc.engine.gl

Ovaj paket sadrži klase koje koriste OpenGL pozive. Ove klase su zadužene za iscrtavanje 2D grafike:

- Texture - služi za učitavanje tekstura iz datoteke, kao i za njeno rukovanje.
- TextureRegion - predstavlja region teksture koji će biti iscrtan. Mogućnost da region bude manji od same teksture je korisna kada imamo jednu teksturu koja sadrži veći broj manjih slika koje predstavljaju razne objekte.
- Vertices - koristi se u klasi SpriteBatcher kako bi se iscrtavale teksture. Tekstura je predstavljena jednim pravougaonikom koji odredjuju četiri tačke koje pored svog položaja imaju i podatak o položaju u odnosnu na teksturu (koordinate teksture). Ovi podaci se smeštaju u niz koji se pri iscrtavanju prosleđuje grafičkoj jedinici pomoću OpenGL poziva.
- SpriteBatcher - ova klasa predstavlja vid optimizacije pri iscrtavanju velikog broja tekstura. Kako bi se jedna tekstura iscrtala potrebno pozvati nekoliko OplenGL funkcija kako bi se omogućilo njeno iscrtavanje, takođe je potrebno izračunati položaj tačaka koje predstavljaju dati pravougaonik na kom će slika biti iscrtana. Ova klasa je uvedena kako bi se automatizovalo računanje i kako bi se smanjio broj OpenGL poziva. SpriteBatcher može da iscrtava proizvoljan broj tekstura a da su pri tome grafički pozivi za njihovo omogućavanje obavljeni samo jednom.

4.1.3 Paket bsc.engine.impl

Ovaj paket pored implementacije prethodno pomenutih interfejsa sadrži i implementaciju glavne petlje u okviru Android aktivnosti. Klase sadržane u ovom paketu su:

- AccelerometerHandler - služi za preuzimanje podataka senzora.
- AndroidFileIO - služi za rukovanje datotekama.
- AndroidInput - klasa zadužena za kontrolu celokupnog ulaza, touch, tastatura i senzor. U zavisnosti od toga da li platforma podržava moultitouch, kao klasa za očitavanje pritiska se koristi SingleTouchHandler ili MultiTouchHandler.
- KeyboardHandler - služi za očitavanje pritiska tastature.
- SingleTouchHandler - služi za očitavanje pritiska ekrana.
- MultiTouchHandler - služi za očitavanje pritiska ekrana u slučaju mogućih višestrukih pritisaka.
- TouchHandler - intefejs koji nasleđuje OnTouchListener.
- TimerManager - upravlja vremenskim kontrolama.
- GLGraphics - sadrži instancu GLSurfaveView na kojoj se iscrtava grafika, i sadrži instancu objekta GL10 preko kog se pozivaju sve OpenGL funkcije.
- GLScreen - proširuje klasu Screen i sadrži referencu na GLGame i GLGraphics kako bi se mogli preuzimati podaci o ulazu, i kako bi se iscrtavala željena grafika u okviru jedne scene.
- GLGame - predstavlja glavnu klasu aplikacije koja proširuje Android klasu Activity i implementira klasu Game kao i klasu Renderer (kako bi se omogućilo iscrtavanje grafike pomoću funkcije onDrawFrame koja je abstraktna i potiče iz ove klase). GLGame sadrži instance svih prethodno navedenih klasa iz ovog paketa, takođe implementira glavnu petlju u kojoj se svake iteracije osvežavaju funkcije: update (položaj objekata, detekcija i rešavanje sudara, kretanje i dinamika objekata) i present (iscrtavanje grafike). Ovim funkcijama se kao parametar prosleđuje vreme u milisekundama koje je prošlo od prethodne iteracije. Takođe, u glavnoj petlji se i osvežava funkcija koja proverava dali je neka od postojećih vremenskih kontrola istekla.

4.1.4 Paket bsc.engine.math

U ovom paketu se nalazi klasa Vector2. Ona obuhvata skup metoda za rukovanje dvodimenzionalim vektorima.

4.1.5 Paket bsc.engine.objs

U ovom paketu se nalaze klase koje zajedno čine baznu klasu objekta (klasu Object2D). Objekat klasa kao člana sadrži abstraktnu klasu BoundingVolume, koja predstavlja osnovu za klase BoundingRect (ograničavajući oblik je pravougaonik) i BoundingCircle (ograničavajući oblik je krug), i koristi se u detekciji sudara i njenoj obradi. Glavna funkcija svakog objekta je funkcija:

```
public void integrate(float deltaTime)
```

Ova funkcija je zadužena za pomeranje objekta na osnovu vremena koje je prošlo od prethodnog poziva. Objekti se nalaze u dvodimenzionalnom prostoru pa se za računanje kretanja koriste vektori. Položaj objekta se osvežava tako što se prvo izračuna vektor ubrzanja:

a = F * m

gde je *a* - ubrzanje, *F* - sila koja deluje na objekat, *m* - masa objekta. Nakon čega se računa vektor brzine tako što se poveća za umnožak ubrzanja i vremena koje je prošlo od prošlog osvežavanja.

Nasleđivanjem bazne klase Object2D su dobijene dve nove, animirani objekat (Animated2DObject) i objekat sa statičnom teksturom (TexturedObject).

4.1.6 Paket bsc.engine.physics

U ovom paketu se nalazi implementacija sistema za detekciju i obradu sudar. Sistem se sastoji iz dve klase:

- Collision
- CollisionManager

Klasa Collision sadrži podatke o sudaru, odnosno objekte koji su u sudaru. Svaki objekat poseduje abstraktnu funkciju

```
public abstract void onCollision(Object2D obj)
```

koja se poziva od strane oba objekta koja učestvuju u sudaru i kao parametar funkciji se šalje referenca drugog objekta kako bi se, prilikom pisanja koda specifičnog za igru, mogla dodati dodtna reakcija na sudar, koja moze biti drugačija za sve vrste objekata. Ova klasa služi za proveru već detektovanih sudara kako se isti sudar nebi registrovao dva puta, pošto se svaki objekat u jednom pravougaonom delu mape proverava sa svim ostalim.

Klassa CollisionManager sadrži funkcije za samu detekciju i obradu sudara, takođe sadrži funkciju za registrovanje objekata u sistem kao i za njihovo uklanjanje. Sistem za sudaranje nije automatski uključen u svaku scenu, on se pri kreiranju scene inicijalizuje na osnovu parametara koji određuju veličinu i broj delova mape na koji je ona izdeljena, nakon čega je potrebno registrovati sve objekte koji učestvuju u sudaru.

Kako bi sistem što bolje funkcionisao potrebno je dobro odrediti veličinu ćelije (dela mape) tako da ona ne bude ni premala a ni prevelika. U slučaju da je premala može se desiti da je neki objekat veći od same ćelije u kom slučaju se gubi ceo smisao ovog pristupa detekcije sudara. U slučaju da je prevelika, gubi se na brzini detekcije jer se puno objekata međusobno poredi iako nisu u blizini jedni drugima, pošto je cilj ovog pristupa detekciji u tome da se porede samo objekti koji su blizu jedni drugima i postoji velika šansa de će se sudariti.

4.1.7 Paket bsc.engine.test

U ovom paketu se nalaze test scene, u kojima je testirana funkcionalnost na jednostavnim primerima. Testiran je sistem za sudaranje, i animirani objekat.

4.1.8 Paket bsc.engine.ui

U ovom paketu se nalazi klasa koja predstavlja dugme. Dugme je nasleđen objekat sa teksturom, i ono nema predefinisani izgled, već se tekstura mora posebno podesiti. Prilikom pritiska na dugme poziva se funkcija onTouch (koja pripada onTouchListener-u, koji je potrebno dodati prilikom kreiranja dugmeta) u koju je potrebno definisati reakciju na pritisak. Pored teksture potrebno je podesiti dodatne parametre kao što su položaj i veličina dugmeta.

4.2 Moduli igre

U ovom delu se nalaze moduli specifični za igru koju implementiramo, i ona se sastoji od sledećih paketa.

- bsc.game - sadrži opcije igre kao i podatke o igraču.
- bsc.game.mng - sadrži klasu koja kontroliše loptice.
- bsc.game.objs - sadrži klase svih objekata u igri.
- bsc.game.screens - sadrži sve scene (koji nasleđuju klasu GLScreen) koje čine igru.

4.2.1 Paket bsc.game

U ovom paketu se nalaze klase:

- GameInfo - sadrži podatke o početnom broju života svakog igrača i o vremenu posle kog se na teren dodaje još jedna loptica (ovo vreme se nakon isteka povećava za 50%). Broj života je moguće promeniti pre početka igre.
- Player - ova klasa sadrži podatke specifične za svakog igrača, zadužena je za iscrtavanje i upravljanje objekta kojim igrač brani svoju stranu ekrana i vodi računa o tome koliko igrač ima života.

4.2.2 Paket bsc.game.mng

U ovom paketu se nalaze klase koje kontrolišu broj loptica na ekranu. Kada neka loptica izađe sa ekrana, nakon pola sekunde stvara se nova loptica u nekim od ćoškova terena, takođe posle isteka određenog vremena dodaje se po jedna loptica sve do maksimalnog broja koji iznosi 10.

4.2.3 Paket bsc.game.objs

U ovom paketu se nalaze svi objekti koji se koriste u igri. Ovi objekti su dobijeni nasleđivanjem objekta sa teksturom. Objekti sadržani u ovom paketu su:

- Ball - predstavlja lopticu, ima specificirane parametre (kao i sve naredne klase u ovom paketu) kao što su tekstura, veličina, ograničavajući oblik, masa, trenje. Kako bi objekat bio potpun ima definisanu funkciju reakcije na sudar. U ovoj funkciji je implementirana reakcija u slučaju sudara sa nekom drugom lopticom i sa zidom ili objektom svakog igrača, koji su pravougaonog oblika.
- HorizontalWall - predstavlja statičan horizontalni zid.
- VerticalWall - predstavlja statičan vertikalni zid.
- Wall - bazna klasa koju nasleđuju vertikalni i horizontalni zid.
- PlayerBar - predstavlja objekat koji svaki igrač upravlja. Prilikom sudara loptice sa ovim objektom brzina loptice se povećava za 10%, i loptica se odbija po principu refleksije. Ova klasa takođe obrađuje podatke o dodiru ekrana na osnovu čega se pomera same levo - desno ili gore - dole u zavisnosti od položaja na ekranu koij može biti horizontalan ili vertikalan.

4.2.4 Paket bsc.game.screens

U ovom paketu se nalaze klase koje predstavljaju scene koje se u određenom trenutku prikazuju na ekranu:

- MainMenuScreen - scena koja je prikazana pri pokretanju aplikacije. Sadrži dugmad pomoću kojih se može početi igra ili promeniti opcije.
- GameScreen - glavna scena u kojoj se odvija sama igra.
- OptionScreen - sadrži nekoliko dugmadi pomoću kojih se bira broj početnih života.

5. Ispitivanje i verifikacija

Ispitivanje i verifikacija funkcionalnosti aplikacije je realizovana upotrebom ispisa na konzoli prilikom pokretanja aplikacije u Android emulatoru (LogCat), kao i upotrebom kalse GLScreen pomoću koje su kreirane određene scene kako bi demostrirale i omogućile proveru ponašanja sistema.

5.1 Registrovanje dodira i pritiska tastature

Ispitivanje registrovanja dodira i pritiska tastature je realizovano upotrebom ispisa konzole u kojoj se prilikom pritiska određnog dugmeta ispisuje odgovarajući kod, i prilikom svakog dodira ekrana se ispisuju koordinati mesta dodira.

5.2 Sistem sudaranja i kretanje objekata

Ispitivanje funkcionalnosti sistema za detekciju i rešavanje sudara, kao i samo kretanje objekata je realizovano upotrebom posebne scene u vidu demonstracije. Na sceni se nalaze četiri zida, po jedan na svakoj strani ekrana, i četiri loptice koje se kreću određenom brzinom. Ove lptice se sudaraju i međusobno i sa zidovima. Verifikacija je izvršena samim posmatranjem kretanja loptica pre i posle sudara.

5.3 Upravljanje lopticama u igri

Pod upravljanjem loptica se smatra njihovo stvaranje na terenu. Ovo je testirano upotrebom ispisa u konzoli. Prilikom stvaranje svake nove loptice ispisuje se ukupan broj loptica na ekranu, ovaj broj se ispisuje i prilikom nestanka loptica sa ekrana. Prilikom ispisa ovog podatka dostupna je i informacija o vremenskom trenutku. Na osnovu ovog vremenskog trenutka je verifikovano to da se svaka loptica nakon nestanka sa ekrana ponovo stvara za pola sekunde.

6. Zaključak

U ovom radu je ralizovana Android multitouch aplikacija u vidu igre.

Prilikom realizacije programskog rešenja aplikacije korišćena je knjiga [1] uz čiju su pomoć realizovane klase koje se nalaze u paketima bsc.engine, bsc.engine.gl, bsc.engine.impl.

Realizovano rešenje se može koristiti na svim platformama Android operativnog sistema koje podržavaju multitouch, različitih rezolucija ekrana.

Rešenje je ispitano na dve Android platforme, odnosno dva telefona sa Android operativnim sistemom, sa rezolucijama ekrana od 800x480 i 480x320.

U postojeću aplikaciju je moguće dodati opcije za proizvoljan broj igrača od 2 do 4. Takođe je moguće dodati i opciju za biranje drugog tipa igre koji bi se zasnivao na vremenskom trajanja jedne partije i gde bi pobednik bio onaj sa najmanje propuštenih loptica.

Moguće je poboljšati trenutnu grafiku, i ubaciti neke efekte u vidu animacije.

7. Literatura

[1] Mario Zechner: Beginning Android Games

[2] Ian Milington: Game Physics Engine Development