VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HRA PRO ANDROID OS

AN ANDROID OS GAME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **LUKÁŠ SALVET**
AUTHOR

VEDOUCÍ PRÁCE **Ing. IGOR SZŐKE, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Salvet Lukáš**
Obor: Informační technologie
Téma: **Hra pro Android OS**
An Android OS Game
Kategorie: Softwarové inženýrství

Pokyny:

1. Nastudujte základy implementace v Android OS.
2. Navrhněte a implementujte zvolenou hru.
3. Hru otestujte na vhodném vzorku beta-testerů.
4. Pokračujte v implementaci a změnách návrhu hry pro co nejlepší UX a herní zážitek. Získejte zpětnou vazbu od uživatelů. Zveřejněte aplikaci na Google Play.
5. Zhodnoťte výsledky a navrhněte směry dalšího vývoje.
6. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Podle pokynů školitele

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 ze zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szőke Igor, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L. S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá tvorbou 2D hry pro zařízení s operačním systémem Android. Popisuje možnost tvorby her v tomto prostředí. Hlavně pak využití knihovny OpenGL ES 2.0, kterou jsem pro implementaci použil. Dále je v práci popsáno renderování textu, použití Google play games services pro achievementy a leaderboard, testování hry, atd. Beta verze hry byla vydána na Google Play.

Abstract

This work deals with creating 2D games for devices running Android. It describes the ability to create games in this environment, especially the use of the OpenGL ES 2.0 library which I applied for implementation. In this work there are also described text rendering, Google Play Games Services for achievements and leaderboard, game testing, etc. Beta version of the game was released on Google Play.

Klíčová slova

2D hra, Android, OpenGL ES 2.0, Google Play Games Services, Vykreslování textu

Keywords

2D game, Android, OpenGL ES 2.0, Google Play Games Services, text rendering

Citace

SALVET, Lukáš. *Hra pro Android OS*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Igor Szőke, Ph.D.

Hra pro Android OS

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szőkeho, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Salvet
16. května 2017

Poděkování

Děkuji svému vedoucímu práce Ing. Igoru Szökemu, Ph.D, za aktivní přístup k vedení této práce a za jeho užitečné rady.

Obsah

Kapitola 1

Úvod

Přibližně rok před zapsáním bakalářské práce, se mi vnukla do hlavy myšlenka na vytvoření 2D hry určené pro mobilní zařízení. Byl jsem přesvědčen, že tuto hru budou hrát sta tisíce lidí a zároveň díky ní můžu zbohatnout. Proto jsem se tento nápad snažil zrealizovat, ale jako nezkušenému programátorovi se mi to příliš nedařilo. Chtěl jsem mít hru co nejdříve hotovou, a tak jsem hledal co nejjednodušší způsoby jak ji implementovat. Všechny víceméně selhaly. Ale já jsem svůj cíl nevzdal. Využil jsem možnost zapsat si takové téma bakalářské práce, abych v jeho rámci mohl hru zrealizovat a konečně dovést k cíli.

Hlavní princip mé hry, kterého jsem se držel již od samého počátku, sestává z kuliček odrážejících se od hran hrací plochy. Úkolem hráče bude tyto kuličky ničit. Aby se průběh hry nezakládal pouze na bezduchém mačkání kuliček na hrací ploše, musí existovat i kuličky, které hráč naopak ničit nesmí. Maximální počet kuliček na hrací ploše bude omezený. Hráč bude informován o aktuálním počtu kuliček na hrací ploše. Co by mělo hru udělat výjimečnou je fakt, že každá kulička bude mít určitý počet životů. Po prasknutí kuličky s $n > 1$ počtem životů z ní vypadnou dvě kuličky s $n - 1$ počtem životů. Tato mechanika by navíc měla ještě ztížit výběr kuliček, které je rozumné praskat. Jediný, a doufám, že dostačující cíl bude nahrát ve hře co nejvyšší skóre.

Jako první se v této práci zaměřím na průzkum trhu a stanovím cíle hry. V další kapitole popíši možnosti tvorby hry, a zdůvodním svůj výběr OpenGL ES 2.0. Následuje krátká kapitola s nutnou teorií, pro pochopení implementační části. V implementační části popisuji hlavně vykreslování a chování kuliček, dále renderování textu, tlačítek a použití Google play games services pro achievementy (úspěchy) a leaderboard (žebříček). V poslední kapitole se zabývám testováním hry, které jsem prováděl ve třech fázích hry.

Kapitola 2

Hry s podobnými principy mé hry

V době, kdy jsem dostal nápad na vytvoření mé hry, jsem se snažil zjistit, zdali již taková hra neexistuje. Žádnou jsem nenašel. To stejné jsem udělal i nyní. Navíc jsem se snažil vzít si příklad z již existujících her.

V této kapitole popíši několik úspěšných her založených na podobném principu, budu zamýšlet a hledat možný klíč jejich úspěchu.

2.1 Ant Smasher[1]

- Od vývojáře: Best Cool and Fun Games
- Instalace: 100 000 000 – 500 000 000
- Hra založena na zabíjení (mačkání prstem) mravenců

Mravenci chodí z horní hrany hrací plochy směrem dolů. Úkolem hráče je zničit je dříve, než se dostanou na spodní hranu hrací plochy. Kromě mravenců se občas objeví i nějaká včela, která se ovšem zabít nesmí. Hráč má omezený počet životů, který se odčítá pokud mravenec uteče (dosáhne spodní hrany hrací plochy), nebo při zabití včely. Ve hře je implementováno několik typů pohybů mravenců a včel. Hra nabízí několik herních módů, ale pouze jeden je zdarma. Jelikož jsem si ostatní módy hry nezakoupil, budu hodnotit pouze ten základní.

Obrovský počet stažení hry přisuzuji, tomu, že hráči hned ví, co budou dělat a znají to z reálného světa. Kdyby měl hráč ve hře mačkat nějaké abstraktní tvary, hra by nejspíše takový úspěch neměla.

Co se týče hratelnosti, tak mě tato hra nezaujala. Dle mého názoru je velmi pomalá a nezáživná (trvá dlouho, než se stane zabíjení mravenců obtížné). Hra implementuje možnost sdílet své skóre, ale jelikož je nutné při každé hře její pomalou částí, překonávání skóre mě nebaví. Hra nemá achievementy, což ale nevidím jako zásadní problém. Mám za to, že tato hra je na Google Play už hodně dlouho a svou uživatelskou základnu si získala v době, kdy se tam moc her ještě nenacházelo. Na dnešní dobu je hra zastaralá. Jako obrovský problém ve hře vidím neustálé podlé reklamy na antivirový program. Nevím, jestli za to můžou tvůrci hry. Ve hře se ale každopádně nachází tlačítko, u kterého není zřejmé, co dělá. Pokud uživatel na tlačítko klikne, bude přesměrován na reklamu nebo nějakou aplikaci v Google Play. V příloze A jsou screenshoty, z mého pohledu, nepřípustných reklam. Screenshot ze hry je na obrázku 2.1a

[1] Ant Smasher https://play.google.com/store/apps/details?id=com.bestcoolfungames.antsmasher

2.2 Balloon Smasher[2]

- Od vývojáře: Zabuza Labs
- Instalace: 1 000 000 – 5 000 000
- Hra založená na praskání balónků

Princip hry je založen na balóncích, které stoupají vzhůru. Úkolem hráče je prasknout je dříve, než odletí z hrací plochy. Kromě balónků se ve hře objevuje i různý hmyz a ptáci, kteří se naopak mačkat nesmějí.

Ve hře je kromě klasického módu i challenge mód, který je intenzivnější, ale stále se mi zdá být lehký. Podle recenzentů hry usuzuji, že hru hrají převážně děti a ženy, kterým může nižší obtížnost vyhovovat. Po ztrátě životu je možnost podívat se na reklamu, a tím se vrátit zpět do hry. Toto dle mého názoru degraduje váhu celého skóre, které bylo nahráno, tím pádem jsou uživatelé demotivováni v soutěžení mezi sebou. Osobně bych chtěl vyhnat obtížnost mé hry na maximum a dát všem hráčům stejné podmínky při hraní, ať jsou offline nebo mají zapnutý internet a zobrazují se jim reklamy. Co se mi na hře dále nelíbí jsou systémová upozornění, která uživatele lákají jít hru hrát. Já osobně jsem schopen kvůli tomuto hru okamžitě odinstalovat. Screenshot ze hry je na obrázku 2.1b

2.3 Piano Tiles 2[3]

- Od vývojáře: Cheetah Games
- Instalace: 100 000 000 – 500 000 000
- Hra na piáno

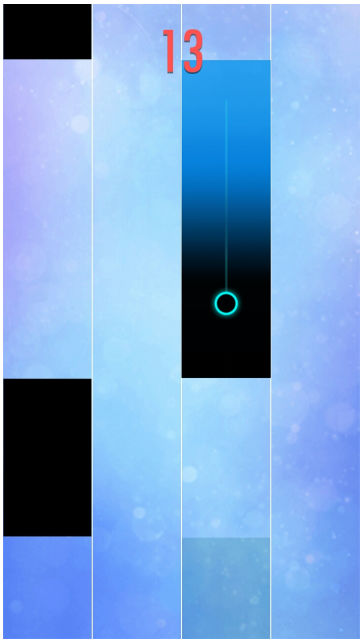
Ve čtyřech sloupcích jedou v rytmu hudby klávesy nebo slidery směrem dolů. Úkolem hráče je mačkat klávesy a držet slidery. Za každou zmáčknutou klávesu nebo podržený slider se dostávají body. Hra končí, jakmile nějakou z kláves nebo slider hráč nestihne zmáčknout. Podle počtu bodů hra odměňuje hráče hvězdičkami. Podle počtu hvězdiček uživatel vidí, jak dobře si vedl. Plný počet hvězdiček u některých skladeb dá zabrat i hodně zkušeným hráčům. Každá hra stojí hráče 1 ze 30 životů. Tyto životy se obnovují časem za zhlédnutí reklamy (hráč obdrží fiktivní měnu, za kterou si může koupit životy nebo skladby), nebo se dají koupit i za peníze. Hra dokáže natolik zaujmout, že si může tuto obchodní strategii s životy dovolit. Myslím, že u mé hry by tato strategie neměla moc smysl. Ve hře je na výběr nespočet skladeb, které se postupně odemykají. Hra nabízí i porovnávání skóre s přáteli přes sociální sítě. Nachází se zde i skladby, po jejichž dohrání uživatel vidí, o kolik % je lepší než ostatní. Této hře nemám co vytknout. Počet stažení je rozhodně zasloužený. Screenshot ze hry je na obrázku 2.1c

[2] Balloon Smasher https://play.google.com/store/apps/details?id=com.legendarygames.balloon_smasher

[3] Piano Tiles 2 https://play.google.com/store/apps/details?id=com.cmplay.tiles2

(a) screenshot ze hry Ant Smasher

(b) screenshot ze hry Ballon Smasher



(c) screenshot ze hry Piano Tiles 2

Obrázek 2.1: Screenshoty z popisovaných her

2.4 Předělávky počítačové hry Moorhuhn

- Instalace: do 1 000 000
- Hra založena na střílení krocanů

Obrázek 2.2: screenshot ze hry Chicken Shot

Úkolem hráče je za daný čas nasbírat co nejvyšší počet bodů. Body získává převážně střílením krocanů. Krocani letí nebo na krátkou chvíli vykukují ze svých úkrytů. Dále jsou zde „skryté“ objekty, za jejichž sestřelení hráč získává nebo naopak ztrácí body. Na to, za jaké objekty se body ztrácí, a za které se připisují, si musí každý přijít sám hraním.

Existuje spousta free to play verzí této hry, ale i pár zpoplatněných verzí. Hrál jsem 3 free to play verze této hry a na každé jsem našel z mého pohledu značné nedostatky. Například: Absence menu, špatný pohyb kamery, nepříliš dobrý hitbox krocanů, ... Existuje však verze, jež se dá koupit za „pár korun“[4]. Má 100 000 – 500 000 stažení, což je na takto jednoduchou hru opravdu hodně. Tuto verzi jsem nehrál, ale předpokládám (na základě zhlédnutí videí z této hry), že neobsahuje neduhy její free to play verze, a to je klíč k úspěchu.

[4]Moorhuhn Deluxe https://play.google.com/store/apps/details?id=com.Doyodo.Moorhuhn

2.5 Hry od studia ketchapp[5]

Toto studio má na svém kontě desítky her. Většina jejich her je velice jednoduchých a „originálních“. Hry se vyznačují jednoduchou grafikou a jednoduchou hratelností. V mnoha hrách stačí pouze jedním prstem rytmicky mačkat na display a tím měnit směr hada, kuličky, rakety, spouštět nějaké události, regulovat výšku letícího ptáka, apod. Průměrný počet stažení her dosahuje 5 000 000 – 10 000 000. V hrách implementují, dle mého názoru, dnes již standardní funkce: achievementy, vypnutí zvuku, vypnutí reklam (za poplatek), ohodnocení aplikace v obchodě, možnost porovnat skóre s přáteli nebo možnost nakoupit si za herní nebo reálnou měnu vizualizační balíčky. Studio prosazuje free to play model s reklamami, u některých her mají navíc mikrotransakce.

Po delším sledování trhu jsem si všiml, že většina jejich her jsou kopie více či méně úspěšných originálů. Jelikož ale studio Ketchapp sleduje na sociálních sítích velký počet lidí, jejich hry jsou mnohdy více stahované, než i lépe zpracované originály. Na internetu jsem našel velké množství stížností od herních vývojářů na nekalé praktiky firmy Ketchapp, bohužel s tím nejspíš nikdo nic nezmůže. Na konci roku 2016 Ketchapp koupila firma Ubisoft, takže i kdyby Ketchapp postupoval protiprávně, bude finančně náročné vyhrát právní spor proti takovému gigantu.

2.6 Zhodnocení ostatních her a cíle mé hry

Hodilo by se ještě popsat hru Flappy Birds, která zažila obrovský boom a odstartovala vývoj jednoduchých free to play her, ale originál jsem nehrál (na marketu již není) a její klony popisovat nechci, protože si myslím, že jsou stahované z velké části kvůli jménu.

Jak už jsem částečně napsal u popisu hry Balloon Smasher, chci aby má hra byla těžká a jednotlivé hry netrvaly moc dlouho. Obtížnost hry chci využít jako prostředek, díky němuž se bude hra šířit mezi lidmi. Hra Flappy Birds byla také extrémně těžká a myslím, že to byl hlavní důvod její popularity. Ve hře chci mít možnost sdílet skóre a implementovat achievementy. Tyto dvě funkce by taky měly pomoct s šířením hry. Jak už jsem psal v úvodu, na hře chci taky něco vydělat, takže do ní dám reklamy. Nechci ale, aby reklamy ovlivňovaly gameplay, tudíž nebudu používat možnost vrátit se díky reklamám do hry. S největší pravděpodobností nepoužiju ani bannery, které konstantně zabírají část plochy displaye. Vybízí se ale otázka, zdali s reklamami počkat na nějakou uživatelskou základnu, nebo je do hry dát hned při vydání. Pro větší motivaci hrát hru dlouhodobě chci vytvořit více barevných vzorů a textur, kterými budu hráče odměňovat za splněné achievementy.

Věc, která mě na Google Play zaráží, je množství klonů úspěšných her. Navíc tyto klony nevytváří pouze nezávislí vývojáři, ale mnohdy taky velká studia, viz. Ketchapp. Abych ostatním vývojářům neulehčoval klonování mé hry, budu se snažit zabezpečit zdrojové kódy.

[5] ketchapp http://www.ketchappstudio.com/

Kapitola 3

Možnosti tvorby her

Hlavním rozhodnutím u tvorby her je, jakým způsobem vykreslovat grafiku. Valná většina herních vývojářů používá univerzální herní enginy, které vývoj drasticky usnadňují. Pak jsou tu takoví, kterým žádný herní engine na trhu nevyhovuje, chtějí se odlišit od ostatních her (engin do jisté míry ovlivňuje vzhled hry), nebo nechtějí být odstíněni od nízkoúrovňových věcí a chtějí mít plnou kontrolu nad tím, jak se jejich hra chová.

Pokud se rozhodneme nepoužít herní engine máme tři základní způsoby jak vykreslovat grafiku:

1. Vykreslovat na View [3] (canvas).
2. Vykreslovat na SurfaceView [3] (canvas).
3. OpenGL ES [9]

3.1 On a View

Pro kreslení je potřeba zdědit třídu `View` a implementovat její callback metodu `View.onDraw()`, která se stará o vykreslení námi definovaných objektů. Vykreslování se provádí voláním metody `invalidate()`. Tato metoda požádá android o zavolání metody `onDraw()`. Není garantováno, že android metodu `onDraw()` zavolá okamžitě. Tento způsob kreslení je nejjednodušší, ale poskytuje nejmenší možnosti. Nedoporučuje se pro složitější hry.

3.2 On a SurfaceView

třída `SurfaceView` je podtřída třídy `View`. Vykreslování pomocí třídy `SurfaceView` má výhodu v tom, že není závislé na `View` hierarchii. To znamená, že aplikace nemusí čekat, než je systém `view` hierchie schopný kreslit, ale jelikož se kreslení odehrává v separátním vlákně, určuje si tempo kreslení třída `SurfaceView`.

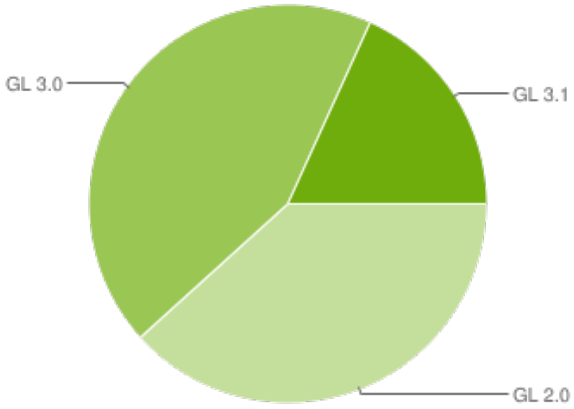
Pro kreslení je potřeba dědit z třídy `SurfaceView` a implementovat `SurfaceHolder.Callback`. Tato podtřída informuje o stavu aplikace. Má metody `surfaceChanged`, `surfaceCreated` a `surfaceDestroyed`.

Počínaje Androidem 3.0 (API 11), je Android 2D vykreslovací řetězec hardwarově akcelerován. To znamená, že všechny operace, které kreslí na `canvas`, používají GPU. Díky této vlastnosti a vlastnímu vláknu se dají pomocí `SurfaceView` udělat i náročnější 2D hry.

3.3 OpenGL ES

OpenGL ES je multiplatformní aplikační rozhraní pro tvorbu 2D a 3D grafiky pro vestavěné systémy. Skládá se z podmnožiny funkcí pro desktopové OpenGL. Vytváří flexibilní a mocné nízkoúrovňové rozhraní mezi softwarem a hardwarem. OpenGL je spravováno neziskovým technologickým konsorciem Khronos Group, jehož členové jsou technologičtí giganti jako Intel, NVIDIA Corporation, Broadcom Corporation, Nintendo Co.,Ltd., QUALCOMM, atd.

OpenGL ES existuje několik verzí. Verze 1.0 se již nějakou dobu nedoporučuje používat z důvodu omezené funkcionality. Na obr. 3.1 je vyobrazeno na kolika zařízeních jsou podporovány určité verze OpenGL. U zařízení funguje zpětná kompatibilita.



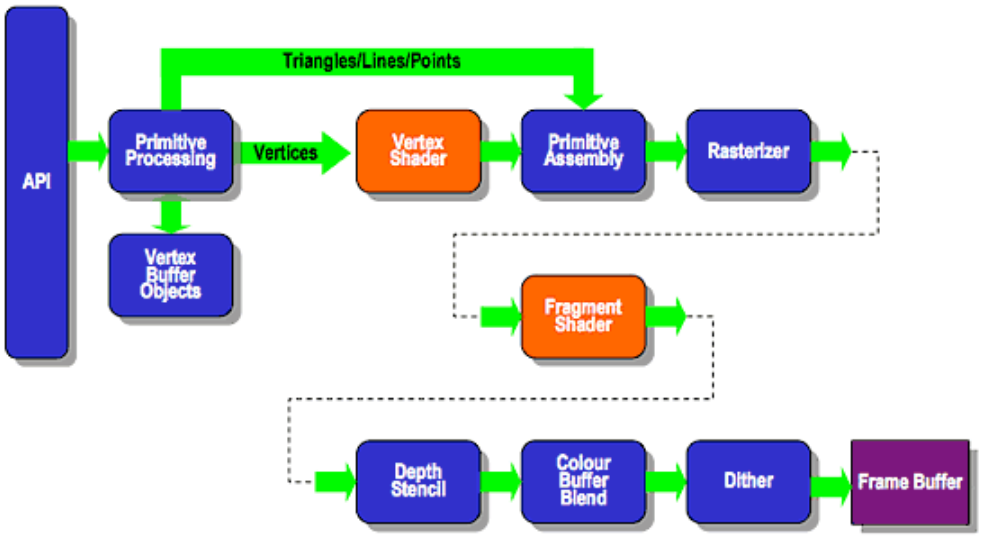
Obrázek 3.1: Zastoupení verzí OpenGL ES na počtu zařízení
převzato z https://developer.android.com/about/dashboards/index.html#OpenGL

OpenGL ES 2.0 vykreslovací řetězec

Vstupem vykreslovacího řetězce je Vertex Buffer Object, což je balíček vrcholů (vertexů). Výstupem je frame buffer, který pro každý pixel obsahuje informaci o jeho barvě. Každý vrchol může obsahovat souřadnice, barvu, texturovací souřadnice a normálový vektor.

- Souřadnice jsou typicky čtyři, čtvrtá souřadnice je homogenizační faktor.
- Barva je třírozměrný (RGB) nebo čtyřrozměrný (RGBA) vektor.
- Texturovací souřadnice mohou být jednorozměrné, dvourozměrné nebo trojrozměrné.
- Normálový vektor je typický vektor o čtyřech souřadnicích. Homogenizační faktor je nastaven na 0 (aby se na něj neaplikovala translace).

1. **Vertex Shader** – jelikož je grafická karta vysoce paralelní zařízení, je vstupem i výstupem jediný vrchol (vertex). Mezi nejčastější operace Vertex shaderu patří transformace vrcholu (umístění vrcholu do globální scény, umístění kamery a projekce) a per vertex osvětlení.
2. **Sestavení primitiv (Primitive Assembly)** – trojúhelníky, úsečky a body.



Obrázek 3.2: OpenGL ES 2.0 vykreslovací řetězec
převzato z oficiální dokumentace [9]

3. **Rasterizace** – vytvoří z primitiv množinu fragmentů.

4. **Fragment Shader** – vstupem i výstupem je jeden fragment. Jeho úlohou je spočítat výslednou barvu fragmentu a zahazovat nepotřebné fragmenty. Počítání barvy obnáší: nanášení barev, textury, per fragment osvětlení, environment mapping (odraz z okolí) atd.

5. **Per-Fragment Operations** – Každý fragment musí projít sadou testů na jejichž základě se rozhodne jestli bude uložen do frame bufferu (může jen ovlivnit hodnotu ve frame bufferu, například při blendingu), nebo zahozen. Mezi testy patří test hloubky, stencil test, blending, dithering atd.

v OpenGL ES 2.0 je programovatelný vertex shader a fragment shader. Shadery se programují v jazyce GLSL. Dále je možné vypínat/zapínat některé Per-Fragment operací. Ostatní operace jsou pevně dány.

OpenGL ES 2.0 na Androidu

Android umožňuje volat OpenGL funkce skrze framework [7] v jazyce java, nebo v nativním jazyce C (NDK). The Native Development Kit (NDK) je soubor nástrojů, které umožňují implementovat část kódu v jazyce C a C++. NDK je vhodné použít u aplikací, u kterých potřebujeme co nejvyšší výkon nebo když se nám hodí kód napsaný v jazyce C (multiplatformní vývoj). Cena, kterou platíme za vyšší výkon a „znovupoužitelnost“, je náročnější vývoj.

Pokud si vybereme možnost implementace OpenGL pomoci Android framewok interfaces musíme implementovat dvě základní třídy `GlSurfaceView` a `GLSurfaceVies.Renderer`.

`GlSurfaceView` je View pro zobrazování OpenGL grafiky. Můžeme v něm nastavit, zdali chceme vykreslovat neustále maximální rychlostí, nebo pouze při vytvoření a na vyžádání. `GLSurfaceVies.Renderer` je rozhraní definující metody potřebné pro kreslení grafiky na `GlSurfaceView`. rozhraní vyžaduje implementaci těchto metod:

- **onSurfaceCreated()** tato metoda je zavolána jednou, když je vytvořeno GLSurfaceView. Měla by by být použita pro operace, jež je potřeba provést jednou, a to při spuštění aplikace.
- **onSurfaceChanged()** Tato metoda je volána, pokud se změní rozměry GLSurfaceView, například přetočíme-li mobil z landscape do portrait polohy. V této metodě by jsme se tedy měli vypořádat se změnou rozměrů GLSurfaceView.
- **onDrawFrame()** Tato metoda je volána pokaždé, když se má GLSurfaceView překreslit. Tudíž tady je to místo, kde se volá většina kódu pro kreslení.

3.4 Herní Enginy s podporou vývoje her pro Android

Herní engine není nic jiného, než sada knihoven implementující znovu použitelný kód pro celou řadu her. Typicky herní enginy poskytují kreslení grafiky, detekci kolizí, fyzikální vlastnosti, zvuky, základní umělou inteligenci atd. Ke kreslení grafiky používají grafickou kartu, tudíž knihovnu OpenGL (pokud kreslí za pomoci procesoru, nenazýval bych je v dnešní době herním enginem). Za výhody herních enginů považuji: multiplatformní vývoj, který většina enginů nabízí, usnadnění vývoje (rychlejší vývoj) a odstínění od většiny nízkoúrovňových věcí. Za hlavní nevýhodu považuji omezenost, tím co umožňují vytvořit (u enginů vyvíjených jako open source je možné si chybějící vlastnosti doimplementovat, ale u jiných s největší pravděpodobností ne).

Krátký popis vybraných herních enginů:

- **Unity**[1] V dnešní době nejpoužívanější herní engine (minimálně pro vývoj her na mobilní zařízení). Má modulární Systém, kdy základní funkce je možné používat zdarma do výdělku $100k za rok. Pokud potřebujeme pokročilejší funkce, je nutné si je měsíčně platit. Velkou výhodou pro nováčky je kvalitně zpracovaná dokumentace, mnoho tutoriálů a mnoho zodpovězených dotazů na vývojářských fórech. Pro programování se používají tyto jazyky: C#, UnityScript a Boo
- **LibGDX**[2] V tomto herním enginu jsem se snažil mou hru vytvořit ještě před zapsáním bakalářské práce. Jelikož jsem ale moc nerozuměl grafice a materiály, ze kterých jsem čerpal, určitou znalost vyžadovaly, práci jsem po krátké době vzdal. Engine je zdarma a je vyvíjen jako Open source. Engine má dobré ohlasy u komunity vývojářů. Je k němu vydáno i několik knih. Přesto ale není pro nováčky doporučován. Programuje se v Javě.
- **Cocos2D**[3] Až 25% her pro iOS je vytvořeno za použití Cocos2D. Pro Android jsem konkrétní čísla nedohledal, ale předpokládám, že to bude méně. Jak již název napovídá je optimalizován pro 2D grafiku, ale podporuje i 3D vykreslování. Je open source a má MIT licenci, takže je zdarma k použití. Programuje se v C++, Lua a JavaScript.

[1] Unity https://unity3d.com/
[2] LibGDX https://libgdx.badlogicgames.com/
[3] Cocos2D http://www.cocos2d-x.org/

- **Monkey X**[4] Používá vlastní programovací jazyk `Monkey X`, který se kompiluje do nativního jazyka několika platforem, které podporuje. Tudíž není potřeba vlastní virtual machine, jako například u Unity. Jako velkou výhodu oproti výše zmiňovaným enginům vnímám možnost v kódu `Monkey X` volat nativní funkce a platformě závislé funkce.

3.5 Moje volba a její zdůvodnění

Rozhodl jsem se použít OpenGL ES 2.0, které má dnes podporu na valné většině zařízení 3.1. Pro OpenGL jsem se rozhodl, protože chci do hloubky rozumět tomu, jak kreslení probíhá. Předpokládám, že v budoucnu pro implementaci mé další hry použiji již existující herní engine, z důvodu výhod uvedených v sekci o herních enginech 3.4.

NDK jsem nepoužil kvůli tomu, že když jsem s vývojem začínal, měl jsem z NDK strach. Bylo to logické, v té době mi přišlo složité implementovat OpenGL kód v Javě, natož s použitím NDK, u něhož jsem se dočetl, že je pro nováčky těžce přijatelné. Po implementaci MVP (Minimum viable product) hry, přečtení knihy [2], několika článků, tutoriálů [10, 4, 1, 8] a fór týkajících se vývoje her na Android, bych si již na NDK troufnul, ale v tu dobu jsem měl již implementovaný slušný základ čítající více než 2500 řádků kódu, který jsem již nechtěl razantně měnit. NDK bych nechtěl použít ani tak kvůli výkonu, ale hodil by se mi kód napsaný v jazyce C, v případě že by hra byla úspěšná a já bych chtěl implementovat verzi pro iOS.

[4] Monkey X https://www.monkey-x.com/

Kapitola 4

Implementace

Hru jsem tvořil v Android Studiu, což je oficiální vývojové prostředí pro Android. Postupně zde popíši nejdůležitější části hry.

4.1 Kostra programu

Jako každá jiná android aplikace, musí mít i moje hra třídu `Activity`, jež je nastavena v `AndroidManifest` jako spouštěcí bod aplikace. Jelikož chceme použít OpenGL, musíme vytvořit třídu dědící z třídy `GlSurfaceView` a nastavit jí v třídě `MainActivity` jako `view`, které bude vykreslováno. V třídě `GlSurfaceView` se nastavuje, kromě jiných věcí, taky režim vykreslování. Tento režim mám nastaven na `Continuously`, což znamená, že se vykresluje co nejrychleji to jde. Dále v této třídě vytvářím instanci třídy implementující rozhraní `GLSurfaceVies.Renderer` (dále nazývanou jako `Renderer`).

Jak již bylo avizováno v sekci 3.3, v třídě `Renderer` je potřeba implementovat tři základní metody:

- **onSurfaceCreated()** V této metodě vytvářím shader programy, nastavuji blending, načítám textury, vytvářím data pro některé míčky, vytvářím některé objekty a inicializuji proměnné.

  Vytváření shader programů obnáší: načíst kód shaderů v jazyce GLSL do proměnné typu string, kompilovat shadery, spojit (link) vertex a fragment shader do jednoho programu a vrátit ID takto vytvořeného shader programu. Další zde zmíněné věci popíšu dále v textu.

- **onSurfaceChanged()** Jelikož se mi zdá zbytečné pro mou hru podporovat landscape orientaci displaye, nastavil jsem v `AndroidManifest` napevno portrait režim. Tím pádem bude tato metoda volána stejně často jako metoda `onSurfaceCreated`, proto by se mohlo zdát, že je pro mě zbytečná. Jenže tato metoda dostává jako parametr rozměry `GlSurfaceView`, které využívám pro výpočet aspectRatio.

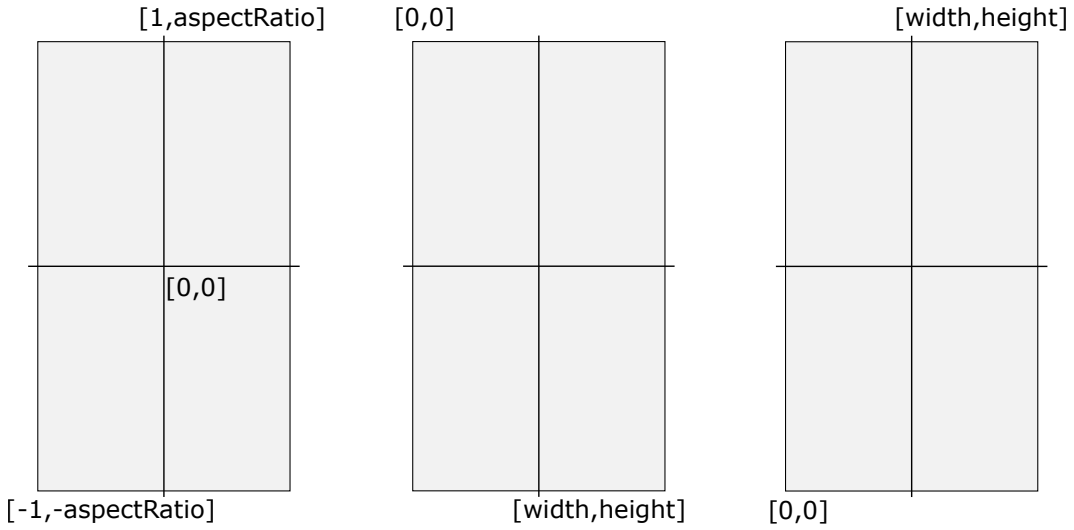
  Ve hře mám objekty, které potřebují znát aspectRatio již při své inicializaci. Jelikož v době volání metody `onSurfaceCreated` aspectRatio ještě neznám, vytvářím je zde. Z jakého důvodu potřebují znát aspectRation je popsáno v sekci 4.2. Na konec ještě nastavuji projekční matici, pro jejíž nastavení potřebuji znát taktéž aspectRatio.

- **onDrawFrame()** Zde probíhá kreslení scény na základě stavu hry.

4.2 Různé souřadné systémy

Při implementaci jsem se setkal se třemi souřadnými systémy. Pro lepší představu souřadných systémů jsem text podložil obrázky 4.1.

- **Souřadný systém scény (world space)** je teoreticky nekonečný. Programátor si volí kamerou, kam se do něj dívá. Já jsem si kameru umístil pevně tak, aby směřovala do bodu [0,0] a zabírala plochu ohraničenou body: levý dolní roh [-1, -aspectRatio] a pravý horní roh [1, aspectRatio]. AspectRatio je podíl výšky displaye šířkou displaye. Slouží k zachování poměru stran vykreslovaných objektů. Kdyby kamera zabírala prostor [-1,-1] až [1,1], nebyl by na nečtvercovém display zachován poměr stran vykreslovaných objektů (čtverec by se například vykresloval jako obdélník).
- **Souřadný systém fragment shaderu (glFragCoord)** souřadnice [0,0] se v tomto souřadném systému nachází vlevo dole a souřadnice [Xmax,Ymax] vpravo nahoře. Souřadnice značí jednotlivé pixely zařízení. Souřadnice jsou přímo mapovány do framebufferu. Když jsem potřeboval předávat fragment shaderu souřadnice objektů, musel jsem je mapovat z world space do glFragCoord.
- **Souřadnice, které dostáváte při onTouchEvent** Při dotyku na display dostaneme souřadnice [x,y], kde bod [0,0] je v levém horním rohu a souřadnice [Xmax, Ymax] vpravo dole. Xmax a Ymax je v pixelech šířka (width) a výška (height) displaye zařízení. Při zjišťování zdali byl při dotyku aktivován nějaký míček, musíme mapovat souřadnice dotyku na souřadnice scény.



(a) world space (b) glFragCoord (c) onTouchEvent souřadnice

Obrázek 4.1: Různé souřadné systémy

4.3 Vše co se týká kuliček

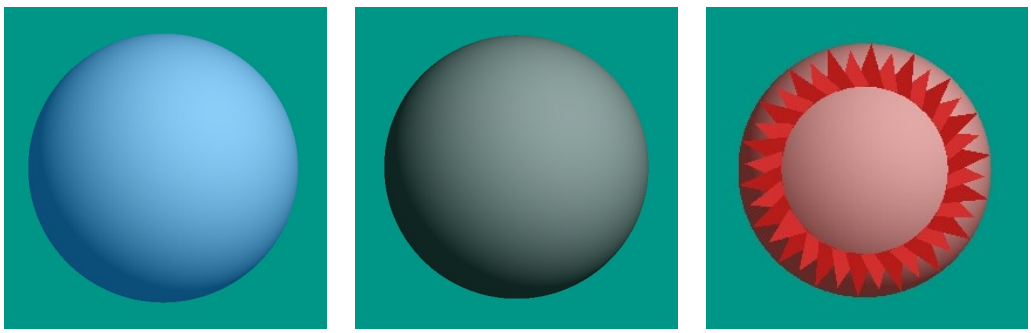
Kuličky jsou hlavními objekty mé hry. Každá kulička má jiné chování 4.3.1. Kuličky se na hrací ploše objevují na základě určitého algoritmu 4.3.6. Pohyb kuliček je lineární 4.3.4. Při nárazu do hrany hrací plochy se od ní odrážejí 4.3.5.

4.3.1 Typy kuliček

Ve hře mám tři typy kuliček:

- **základní kuličky (Normal ball)** Tyto kuličky mají $1 - 5$ životů. Počet životů je znázorněn odstínem barvy. Praskáním těchto kuliček hráč získává skóre. Při prasknutí kuličky s $n > 1$ počtem životů se na jejím místě objeví dvě kuličky s $n - 1$ počtem životů, deadly ball, nebo thorns ball.
- **černá kulička (Deadly ball)** Pokud ji hráč praskne, prohraje hru. Aby kulička na hrací ploše nezůstávala do nekonečna, má omezenou dobu života, po jejíž vypršení zmizí. Dobu života kuličce hlídá vlákno, které si sama vytvoří. Deadly ball má po celou dobu hry 10% šanci na vypadnutí z normal ball a stejnou šanci na vygenerování generátorem.
- **kulička s trny (Thorns ball)** Tato kulička se liší od ostatních již na první pohled. Má kolem sebe rotující prstenec trnů, které jsou ale do její aktivace neškodné. Po jejím prasknutí se prstenec začne zvětšovat a praskat všechny kuličky, jež se trnů dotknou. Jelikož může thorns ball zásadně ovlivnit hru, má pouze 1% šanci na vypadnutí z normal ball. Aby byly hry více vyrovnané, přidělal jsem po testování garanci jedné této kuličky na 80 bodů nasbíraného skóre.

Je tu prostor ještě pro další typy kuliček. Přemýšlel jsem například nad kuličkou, která by na krátkou dobu zastavila čas. Zastavit čas ve hře by na implementaci nebyl žádný problém, ale důvod, proč jsem tuto kuličku neimplementoval, byl fakt, že neumím udělat pěknou animaci zastavení času (například zamrznutí ostatních kuliček). To, jestli by se tato kulička do hry hodila, bych musel rozhodnou na základě testování.



(a) Normal ball (b) Black ball (c) Thorns ball

Obrázek 4.2: Obrázky kuliček ze hry

4.3.2 Kreslení kuliček

Jelikož OpenGL umí vykreslovat pouze trojúhelníky, úsečky a body, musí se jiné tvary vyskládat z těchto pro grafiku elementárních tvarů. Jsou dva rozumné způsoby jak kuličky vykreslit:

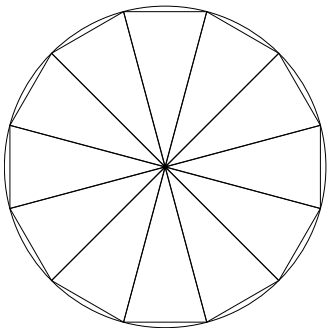
1. **Vyskládat kruh z trojúhelníků** – Na obrázku 4.3a je kruh vyskládaný z dvanácti trojúhelníků. Pro oku dokonalý kruh by jich ale bylo potřeba daleko více. Při této metodě kreslení kruhu je taky dobré brát v potaz velikost kruhu a rozlišení displaye na který se vykresluje. Pokud kreslíme kruh malý, bude nám stačit méně trojúhelníků, než když bude kruh veliký a v popředí.

2. **Vykreslit kruh z pixelů za pomoci fragment shaderu** – Vytvoříme čtverec ze dvou trojúhelníků a ve fragment shaderu rozhodneme, které pixely vykreslíme. Jsou dva způsoby, jak rozhodovat o pixelech, které vykreslit.

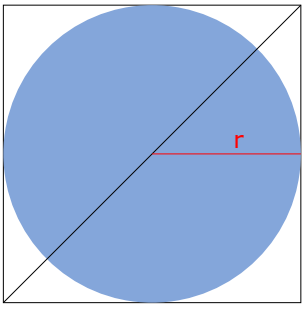
   (a) **Pomocí Textury** – Čtvercovou texturu namapujeme na čtverec popsaný vertexy. Pixely, které nemají být vykresleny musí mít alphu nastavenou na 100%. Pokud nenastavíme čtverci, na který texturu mapujeme, žádnou barvu, bude barva záviset pouze na textuře, a tak dosáhneme tíženého výsledku, tj. vykreslení kruhu. Kvalita vykresleného kruhu závisí na velikosti textury. Pokud bude mít textura malé rozlišení, bude u velkého kruhu vidět raster. Velká textura zabírá více místa v paměti a při kreslení příliš malých obrázků je problém rozhodnout, které pixely z textury vykreslit, tomuto jevu se říká aliasing. Pro minimalizaci aliasingu se často, kromě jiných metod, používá mipmapping.

   (b) **„Matematická“ metoda** – na základě rovnice kružnice ve fragment shaderu rozhodneme, které pixely vykreslit a které zahodit. Toto je znázorněno na obrázku 4.3b. Modré pixely se vykreslí, ostatní se zahodí. Zde bohužel taky vzniká aliasing, který ale pochopitelně nejde řešit mipmappingem, a pro vyhlazení je tedy potřeba použít jinou metodu, např. multisampling.

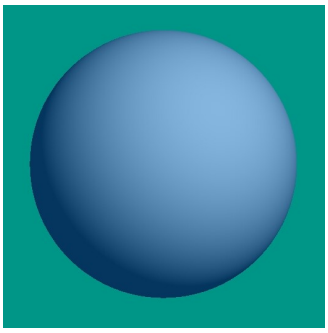
Já jsem na začátku používal pro kreslení kuliček Texturu, ale žádná, kterou jsem našel, nebo vytvořil, se mi nelíbila. Proto jsem začal kreslit kuličky „matematickou“ metodou s použitím per fragment osvětlením, pro které jsem použil difuzní složku z Phongova osvětlovacího modelu. Výsledek je na obrázku 4.3c.



(a) kruh vyskládaný z dvanácti trojúhelníků



(b) kruh vykreslený za pomoci fragment shaderu



(c) kulička s osvětlením

4.3.3 Data prstence thorns ball

Pro potřebu kreslit prstenec s trny pro thorns ball jsem si vytvořil třídu jež mi ho na základě zadaných parametrů vygeneruje. Výsledný vzhled prstence ovlivňují tyto parametry: počet vrstev trnů (druhů), počet trnů v jedné vrstvě, velikost základny trnu a délka trnů. Prstenec počítám pro všechny stavy jeho animace (jak bude vypadat při aktivaci thorns ball). Veškeré data prstence počítám při zapnutí aplikace. Data by se dala exportovat do souboru a při zapnutí hry je z něj načítat, ale jelikož těch dat není mnoho, nechal jsem to v programu.

4.3.4 Pohyb

Kulička při inicializaci dostane hodnotu o niž se bude v ose X a Y posunovat. První, teoreticky ideální metoda, kterou jsem pro pohyb kuliček použil, bylo aktualizovat pozici kuliček vždy před jejich kreslením. Pozice kuličky se posunula o vzdálenost závislou na uplynulém čase od posledního renderování. Ovšem z mně neznámého důvodu se pohyb nepatrně sekal.

Proto jsem přešel na jinou metodu, která pozici kuliček posouvá nezávisle na vykreslování. Pro posun kuliček jsem si vytvořil nové vlákno, které běží po celou dobu života hry a aktualizuje pozici všech vykreslovaných kuliček. Musel jsem najít vhodnou intenzitu aktualizací poloh kuliček, tak aby pohyb vypadal plynule. Fakt, se kterým jsem musel počítat byl, že renderovácí vlákno kreslí 60 FPS (frame per second).

Pokud by frekvence počítání nové pozice byla < 60, stávalo by se, že některá kreslení by se kulička vůbec neposunula. Toto je znázorněno na obr. 4.4a. U všech tří obrázků 4.4 platí, že černé čáry znázorňují renderování a modré s červenými čarami počítání nové pozice. Na obrázku 4.4a v elipse je možno vidět, že pozice zvýrazněna červenou barvou bude použita pro kreslení dvakrát.

Na obrázku 4.4b je četnost počítání pozice v intervalu $(1, 2)$. Červené vzorky se nekreslí, jelikož se mezi renderováním stihne spočítat pozice dvakrát. Je tedy možnost si všimnout, že na místech zvýrazněných elipsami jsou mezi modrými, vykreslovanými vzorky „velké“ skoky.

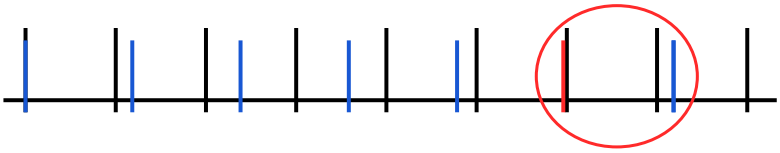
Těchto skoků se zbavíme, budeme-li počítat pozici frekvencí > 2, což je znázorněno na obrázku 4.4c. Při renderování se berou červené vzorky. Platí, že čím rychlejší bude aktualizace pozice míčků, tím plynulejší pohyb kuliček bude. Musíme taky brát v úvahu rychlost pohybu kuliček. Po testování jsem frekvenci aktualizací pozice zvolil na 180krát za sekundu.

4.3.5 Odraz

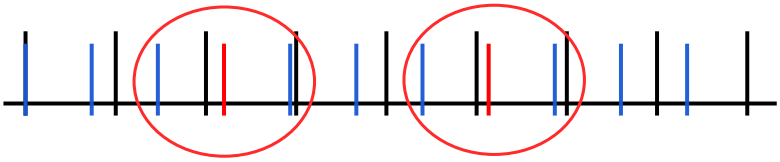
U odrazu kuliček je jediný problém, že nově spočtená souřadnice může být za hranicemi plochy, v níž se kulička pohybuje. Pokud se tak stane, musím provést korekci polohy, jinak by se kulička neodrážela od stěny, ale až za stěnou. Korekce obnáší návrat druhým směrem o ten kus, co jsem za plochou, v níž se kuličky pohybují. Ve skutečnosti téměř nikdy nevykreslím kuličky ve fyzikálním místě odrazu (u stěny), ale okem to nejde poznat.

4.3.6 Generování

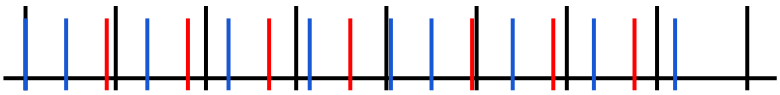
Pro generování kuliček vytvářím každou hru nové vlákno. Typ generovaných kuliček je volen náhodně s určitou mírou pravděpodobnosti pro každou kuličku. Po každé vygenerované kuličce se generující vlákno uspí. Doba, na kterou se uspí, je závislá na druhu kuličky



(a) četnost aktualizací pozice < 1



(b) četnost aktualizací pozice $(1, 2)$



(c) četnost aktualizací pozice > 2

Obrázek 4.4: Četnost aktualizací pozice, oproti rychlosti vykreslování. Renderování je znázorněno černými čarami. Aktualizace pozice modrými a červenými čarami. Elipsy slouží pro zvýraznění kritických momentů, které při kreslení vznikají.

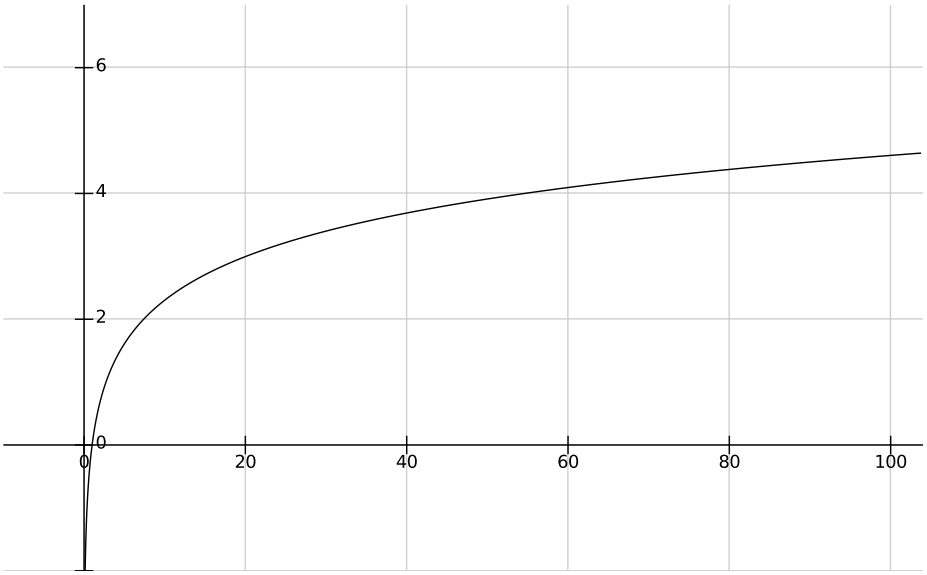
a fázi hry. Fází hry mám na mysli hodnotu, kterou dělím základní čas pro uspání vlákna. Základní čas získám na základě typu vygenerované kuličky. Hodnotu, kterou dělím základní čas získám funkcí $ln(x)$, kde x je proměnná inkrementující se na základě typu vygenerované kuličky. Na obr. 4.5 je zmiňovaná funkce $ln(x)$. Nutno podotknout, že parametr x je na začátku hry nastaven na hodnotu 20, aby se na začátku nemuselo na kuličky čekat. Po spočtení $y = ln(20)$, nám vyjde, že dělitel základního času je na začátku hry číslo 3. Pozice generovaných kuliček je učena náhodně.

Nově vygenerovaná kulička se nějakou chvíli nehýbe a nelze ji prasknout. Toto je implementováno proto, aby nedocházelo k jejímu nechtěnému prasknutí. Nesmrtelnost kuličky je znázorněna jejím nízkým jasem, který se postupně dostává do normálních hodnot.

4.3.7 Kolekce pro ukládání kuliček

Jelikož se kuličky rychle vytvářejí a zanikají, je nutné je mít uloženy v nějaké kolekci. Pro kolekci mám tyto nároky:

- Chci aby hloubka, ve které se kuličky vykreslují odpovídala pořadí, ve kterém se vygenerovaly. Proto potřebuji kolekci, která bude zachovávat pořadí vložených prvků.
- Potřebuji nad kolekcí iterovat v obou směrech. Při vykreslování potřebuji iterovat od začátku (v případě, že kuličky budu vkládat na konec kolekce), protože nejnovější kuličky mají překrývat kuličky starší. Při dotyku chci kuličky vykreslované na popředí testovat na dotyk jako první. Takže iterovat od konce.
- Jelikož do kolekce přistupuji z více vláken (renderovací, UI – user interface, vlákna počítajícího pozici a vlákna generujícího kuličky), potřebuji, aby byly operace nad kolekcí thread-safe.



Obrázek 4.5: rychlost generování kuliček, $ln(x)$

Kolekce, která všechny tyto vlastnosti splňuje je `LinkedBlockingDeque`[1], tuto kolekci jsem tedy použil.

Platí, že pokud je míček prasknut, smažu ho z `LinkedBlockingDeque`, tím pádem na něj neexistuje žádná reference a GC (garbage collector) ho může smazat z paměti. Jelikož je ale vytváření nových objektů drahá operace, mohl bych míčky, které byly prasknuty ukládat do jiné kolekce a při potřebě nového míčku je znovu inicializovat a přidat do `LinkedBlockingDeque`. Tato technika je známá jako návrhový vzor object pool. Je hojně používán, nejen při tvorbě her, ale třeba i pro dočasné uchovávání spojení s databází.

4.4 Renderování Textu v OpenGl ES 2.0

Ve hře potřebuji zobrazovat aktuální počet kuliček na ploše, aktuální skóre a text v menu. Knihovna OpenGl ES 2.0 však nenabízí přímý způsob renderování textu. Existují však tři způsoby, jak text na display dostat:

1. **Umístit TextView nad GLSurfaceView.** Tato metoda je rozhodně nejsnazší na implementaci, ale nejméně flexibilní. Text bude vždy umístěn nad scénou hry a transformace s textem budou omezeny možnostmi `TextView`. Metoda se navíc nehodí pro rychle měnící se text, jelikož vykreslování je závislé na view hierarchii.

2. **Připravit si text jako texturu a tu vykreslovat.** Texturu s textem namapujeme na obdélník. Pokud chceme zachovat rozlišení textu, musí být poměr stran obdélníku shodný s poměrem stran textury. Pokud takto chceme renderovat více druhů textu, je dobré vytvořit si sprite. Sprite je textura obsahující více textur. V době mapování textury se určí, jakou část spritu použít. Sprite se používá hlavně z důvodu, že přepínání textur v grafické kartě je extrémně drahá operace. Tato metoda vykreslování textu je výhodná použít pro malé množství pevně daného textu.

[1] LinkedBlockingDeque https://developer.android.com/reference/java/util/concurrent/LinkedBlockingDeque.html

3. **Naprogramovat si skládání textu z jednotlivých znaků.** Tento způsob může být náročnější na implementaci, ale nabízí nejvyšší flexibilitu. Je potřeba načíst si do paměti sprite obsahující nejlépe všechny znaky, které budeme chtít renderovat. Dále je potřeba:

   - Popsat velikosti písmen a velikost mezer mezi nimi ve spritu.
   - Implementovat mapování ascii hodnot znaků na pozice znaků ve sprite.
   - Pro každý znak popsat 4 vertexy (každý vertex obsahuje tyto složky: pozice, barva, textura) a přidat je do Vertex Buffer Object.
   - Pokud potřebujeme například zarovnávání textu, musíme si ho implementovat sami.

   Jakmile implementujeme funkce pro práci s textem, je již vykreslování libovolného textu snadné. Stačí vytvořit objekt, jenž uchovává samotný text a jeho vlastnosti (pozice, barva, ...) a ten vykreslovat.

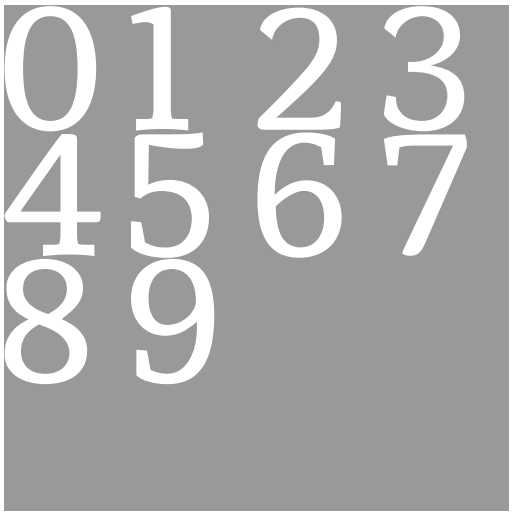
   Existuje několik knihoven implementujících toto vykreslování textu, které je možné použít.

Já jsem se rozhodl zobrazovat čísla metodou 3, protože se v mé hře čísla často mění. metodu 1 nemůžu použít, protože číslo zobrazující počet kuliček na herní ploše chci vykreslovat v pozadí. Pro text v menu, což jsou 3 sousloví, jsem zvolil metodu 2, jelikož je tento text neměnný a není ho mnoho.

Nepoužil jsem žádnou knihovnu, všechno jsem si implementoval sám. Na začátku jsem se inspiroval v tomto tutoriálu [8]. Pro čísla používám dva různé fonty, takže mám vytvořeny 2 sprity. Text v menu mám také v spritu z optimalizačních důvodů.

Sprity jsem si vytvářel v programu Inkscape[2]. Použil jsem fonty s open font licencí. V spritu s textem viditelném na orb. 4.6a jsou barevně vyznačeny oblasti, které poté mapuji na obdélníkové objekty. Sprity jsem udělal preventivně čtvercové, protože jsem se dočetl (na neoficiálních zdrojích), že u nečtvercových textur se na některých zařízeních nemusí povést vygenerovat mipmapu.

(a) sprite s textem použitým v menu



(b) sprite s čísly, font je Tienne

[2]Inkscape https://inkscape.org/en/

4.5 Achievementy a leaderboard

Pro achievementy (úspěchy) a leaderboard (žebříček) Google poskytuje Google play games services [6]. Google play games services poskytuje API pro achievementy, leaderboard, questy (úkoly), uložení hry a propojování hráčů pro multiplayer. API za vás řeší hromadu věcí, od ukládání dat na servery googlu, přes komunikaci s nimi, až po zobrazování uložených dat.

I když toho za vás dělá API opravdu hodně, není pro začátečníka snadné jeho služby do hry dostat. Musíme si vytvořit projekt pro games services v Google play console a provázat ho s aplikací. Google play console je místo, ze kterého vývojář spravuje své projekty.

Nebudu zde popisovat vše, co je nutné pro implementaci achievementů a leaderboardu udělat. To je možné najít v příručce [6]. Pouze zde zmíním problém, na který jsem narazil a zároveň jsem si ho nezpůsobil vlastní nepozorností. Při implementaci jsem postupoval dle materiálů od Googlu [6] a vzorových příkladů, k nim poskytnutým.

Krátce po rozchození achievementů a leaderboardu jsem si dělal pravidelný test na memory leaky, a k mému údivu jsem jich několik měl. Chvíli mi trvalo, než jsem zjistil, že je způsobuje něco kolem Google play games services. Trvalo mi to, protože jsem je tam nečekal, přece jen jsem Google play games services implementoval podle oficiálních materiálů dodaných společností Google. Po přesné lokalizace kamene úrazu jsem si spustil vzorový příklad dodaný k tutoriálu. Zjistil jsem, že příklad také leakuje paměť, a v dokumentaci o této nepříjemnosti není ani zmínka. Přišel jsem na to, že memory leaky způsobuje fakt, že `GoogleApiClient` si uchovává referenci na mou Activitu (context), i po ukončení aplikace a odhlášení se z `GoogleApiClient`. To stejné dělají i `ConnectionCallbacks` a `OnConnectionFailedListener`. U `ConnectionCallbacks` a `OnConnectionFailedListener` to lze po skončení aplikace vyřešit odregistrací těchto služeb. Problém s uchováváním contextu (Nadřazená třída, mimo jiné i třídy Activity) v `GoogleApiClient`, jsem vyřešil tak, že místo activity contextu, jí předávám application context. Application context je vázán na životní cyklus aplikace, takže pokud aplikaci ukončíme, context je vždy uvolněn.

Vše se zdálo, že funguje, ale v průběhu testování jsem si všiml, že se nezobrazují pop-up okna, informující o dosažení achievementů. Chyba vznikla při opravě memory leaků. Ukázalo se, že `GoogleApiClient` potřebuje context, v kterém má pop-up okna vykreslovat. Hledal jsem tedy jak to vyřešit a v Api [5] jsem našel metodu `setViewForPopups`, která chce jako parametr context. Chyba s okny se vyřešila, ale znovu jsem leakoval paměť a tentokrát jsem již nenašel způsob jak problém odstranit. Jelikož chci informovat hráče o splnění achievementu, budu prozatím memory leaky ignorovat. Hra leakuje po každém ztraceném spojení s Google play games services cca 3kB paměti.

Problém s memory leakem je již druhým rokem nahlášen googlu[3], ale do dnešního dne (8.5.2017) nebyl opraven.

4.5.1 Ukládání dat lokálně

Pokud se mi nepodaří odeslat zprávu o splněném achievementu, nebo o překonaném skóre, ukládám si data lokálně do souboru. Jakmile se hráč úspěšně připojí k Google play games services, posílám na server data, která jsem nashromáždil. Kromě dat indikujících stav achievementů a skóre si ukládám stav tlačítka ovládajícího zvuky, největší dosažené skóre, které je zobrazováno v menu, a počet zamítnutých připojení k Google play games services.

[3] Memory leak https://github.com/googlesamples/android-play-location/issues/26

Pokud počet zamítnutých připojení přesáhne určitou mez (aktuálně stačí jedno), nepřipojuji automaticky při startu hry hráče k Google play games services.

Abych zamezil podvrhnutí dat v souboru, potřeboval jsem je zašifrovat. Hledal jsem způsob jak to udělat, ale dozvěděl jsem se, že pokud nechci mít šifrovací klíč uložen někde na serveru, nebude nikdy šifrování 100% účinné. Toto platí, protože neexistuje způsob, jak zabezpečit zdrojové kódy od čtení. Pokud bude mít uživatel rootnutý mobil (administrátorská práva), může se dostat k apk souboru mojí hry, z kterého v pár krocích získá zdrojové kódy. Získání a čtení zdrojových kódů lze znepříjemnit použitím nástroje ProGuard[4]. Nástroj optimalizuje a hlavně obfuskuje kód, tudíž je těžko čitelný, ale pokud není útočník amatér, ví co hledat (v mém případě šifrovací funkci a šifrovací klíč) a umí používat debugger, tak si poradí.

Pro zabezpečení dat jsem tedy implementoval triviální šifrovací funkci, obfuscoval jsem klíč (je poskládaný z několika proměnných nad kterými jsou použity logické funkce) a použil jsem ProGuard.

4.6 Menu hry a stavy hry

Menu hry 4.7a slouží, jako rozcestník pro několik funkcí. Tou hlavní funkcí je samotná hra. V menu hry je kromě textu a tlačítek taky pohybující se kulička. Hru je z menu možno začít dotykem kdekoliv mimo tlačítka. Aby hráč nemusel zbytečně přemýšlet, jak hru zapnout, je v menu pulsující nápis „Tap to start“. Po dotyku hry se neprodleně vymění menu za scénu hry. Kulička, která se nacházela v menu, zůstává do samotné hry. Pokud byla kulička v menu prasknuta, projeví se to ve hře a místo jedné kuličky jsou na herní ploše kuličky dvě.

Hra končí, přesáhne-li počet kuliček na herní ploše určité číslo, nebo je-li prasknuta černá kulička (deadly ball). Počet kuliček vedoucích k prohře jsem stanovil na 16. Samozřejmě jsem testoval, zdali je toto číslo pro hru vhodné, jestli je ještě přehledná, popřípadě jestli se dají pohodlně praskat kuličky s pěti životy (které v sobě skrývají dalších 30 kuliček).

Ukončení hry je reprezentováno animací přechodu ze hry do menu. Animace spočívá ve vygenerování miniatury menu a jeho postupném zvětšování, až do plné velikosti. Pokud je přesáhnut maximální počet kuliček, animace začíná ve středu herní plochy 4.7b. Pokud je prasknuta černá kulička, animace začíná z místa, v kterém byla prasknuta 4.7c.

4.6.1 Stavy hry

Jelikož jsem se rozhodl pro přechod ze hry do menu tímto způsobem, musím při menu animaci vykreslovat dvě hrací plochy. To znamená, že objekty které lze vidět jak v menu, tak zároveň ve hře, musí být duplicitní. Týká se to objektu reprezentujícího pozadí a kolekce s míčky. Abych věděl, které pozadí a kterou kolekci s míčky kreslit, vytvořil jsem si třídu pro uchování a manipulaci s proměnnou určující, jestli jsem v sudé nebo liché hře. Tuto proměnou překlápím vždy na konci menu animace.

Hra může být ve 3 stavech:

1. `menu` – vykresluje se pozadí, aktivní kolekce s míčky, score separátor (čára mezi Top Skóre a aktuálním skóre), tlačítka a menu text

[4]ProGuard https://stuff.mit.edu/afs/sipb/project/android/sdk/android-sdk-linux/tools/proguard/docs/

(a) menu hry

(b) menu animace při přesáhnutí 16 kuliček

(c) menu animace při prasknutí Deadly ball

Obrázek 4.7: Screenshoty z mé hry

2. `hra běží` – vykresluje se pozadí, estetický kruh ve středu na pozadí, počet míčků, skóre a na popředí aktivní kolekce s míčky.

3. `menu animace` – v tomto stavu se vykresluje vše ze stavu `hra běží` a nad tím všechny animované objekty ze stavu `menu`. Animovaná kolekce s míčky se kreslí ta, která není právě aktivní (aktivuje se po skončení menu animace). Jak je realizována menu animace je popsáno v podsekci 4.10.4.

4.7 Zvuky

O přehrávaní zvuků se mi stará třída `SoundsManager`. Jelikož přehrávám pouze krátké zvukové stopy a velmi často používám pro přehrávání `SoundPool`[5]. Kromě `SoundPool`, se pro přehrávání zvuků a celkově medií používá `MediaPlayer`[6]. `MediaPlayer` umí přehrávat pouze jednu zvukovou stopu, kdežto `SoundPool` jich umí na jednou přehrávat tolik, kolik si jich uživatel nastaví.

Ve hře mám dva zvuky, prasknutí kuličky a konec hry. Zvuk prasknutí kuličky, jsem si vytvořil sám v programu Audacity[7]. Druhý použitý zvuk jsem si stáhl (licence mi umožňuje jeho použití v mé hře). Hledal jsem pro hru ještě nějakou vhodnou hudbu, ale nenašel jsem.

4.8 Tlačítka v menu a jejich funkce

Vzhled použitých tlačítek je možno vidět na obrázku 4.8. Čísla u tlačítek odpovídají bodům ze seznamu:

[5]SoundPool https://developer.android.com/reference/android/media/SoundPool.html
[6]MediaPlayer https://developer.android.com/reference/android/media/SoundPool.html
[7]Audacity http://www.audacityteam.org/

1. **tlačítko pro přihlášení se a odhlášení se z Google play games services** – tlačítko mění vzhled a funkci na základě stavu, ve kterém je hráč vůči Google play games service. Pokud je hráč přihlášen, tlačítko slouží pro odhlášení. Pokud naopak přihlášen není, tlačítko slouží k přihlášení se. Dokud hráč nepřeruší přihlašovací rutinu, je do Google play games services přihlašován automaticky při startu hry. Pokud ji přeruší není přihlašován, dokud se nepřihlásí sám pomocí tlačítka. Tato funkcionalita je silně doporučena v příručce k Google play games services [6].

2. **tlačítka pro zobrazení achievementů a leaderboardu** – pokud je hráč přihlášen k Google play games services, zobrazí se mu view s achievementy nebo leaderboard. Pokud přihlášen není, je mu nabídnuto přihlášení.

3. **tlačítko pro ohodnocení hry v Google play** – stiskem tlačítka se zobrazí dialog s krátkou zprávou vybízející k ohodnocení hry. Můžeme přijmout, odmítnout, nebo nechat rozhodnutí na jindy. Výběrem poslední zmíněné možnosti se dialog znovu objeví po určité době. Dobu pro znovu zobrazení dialogu jsem nastavil na jeden den. Implementaci dialogu jsem převzal z `https://github.com/hotchemi/Android-Rate`.

4. **tlačítko ovládající zvuk** – stejně jako tlačítko sloužící pro přihlášení se k Google play games services, mění toto tlačítko svůj vzhled na základě toho, jestli jsou zvuky zapnuty nebo vypnuty. Hra si pamatuje stav tohoto tlačítka i po vypnutí hry.

5. **tlačítko zobrazující stručnou nápovědu ke hře** – text nápovědy je zobrazován v info dialogu[8]

6. **tlačítko s nápisem noAds** – funkce tohoto tlačítka dosud nebyla implementována, protože je spjato s reklamami, a ty ještě ve hře nemám. Pomocí tlačítka si bude moci hráč koupit odstranění reklam.

4.9 Vzhled hry

Jak si je možno všimnout ze screenshotů na obr. 4.7, po každé hře měním vzhled objektů. Při každém kreslení objektu volám třídu, která má na starost správu vzhledu. Ta mi vrátí jak má objekt vypadat. V současné době mám pouze dva barevné styly, ale logiku starající se o vzhled jsem implementoval s důrazem, že do budoucna bude stylů více a uživatel si mezi nimi bude moci vybírat.

Co se týká rozložení textu, je to problém, protože na trhu jsou mobilní zařízení s poměrem stran od 4:3 až po 18,5:9. Udělal jsem tedy dvě rozložení, pro displaye s poměrem stran $\leq 1,5$ a pro ty s poměrem větším.

Dlouho jsme experimentoval se zobrazením nejvyššího dosažené skóre a aktuálního nahraného skóre. Zůstal jsem u varianty, která je momentálně k vidění ve hře. Jsou to dvě nad sebou, čarou oddělená čísla, z nichž vrchní zobrazuje nejvyšší nahrané skóre a spodní aktuálně nahrané skóre. Nejvyšší dosažené skóre má ještě větší velikost. Při testování se zdálo, že všichni ví, co ta dvě čísla znamenají.

Tlačítka jsem stejně jako veškerou jinou grafiku vytvářel v programu inkscape. Miniatury na tlačítkách jsou již dlouho používány v praxi, takže by hráč neměl mít problém určit jejich funkci. Jediné tlačítko, nad jehož vzhledem jsem dlouho přemýšlel, je tlačítko sloužící

[8] Info dialog `https://developer.android.com/guide/topics/ui/dialogs.html`

Obrázek 4.8: Tlačítka použité ve hře. Čísla u tlačítek odpovídají jednotlivým bodům z číslovaného seznamu v sekci 4.8.

pro odhlášení se z Google play games services. Společnost Google nařizuje, že na tomto tlačítku musí být, stejně jako na tlačítku pro přihlášení, miniatura gamepadu. Tlačítko by dle mého názoru bylo nejintuitivnější, kdyby byl vedle miniatury gamepadu vyobrazen text „odhlásit se“, anglicky pak „sign out“. Tento text by se mi však do kulatého tlačítka nevešel, tím pádem by tlačítko muselo mít jiný tvar. Jelikož ale trvám na jednotném stylu tlačítek, ustoupil jsem od intuitivnosti. Tlačítko je tedy kulaté a je na něm vyobrazen slabě přeškrtnutým gamepad.

4.10 Popis animací použitých ve hře

4.10.1 Rotace trnů u thorns ball

Při každém kreslení se thorns ring otočí o úhel závislí na čase uplynulém od posledního renderování. Funguje to na stejném principu, jako metoda, kterou jsem chtěl použít pro pohyb 4.3.4. U pohybu jsem pozoroval drobné záseky, zde to funguje bezchybně.

4.10.2 Aktivace thorns ball

Po aktivaci thorns ball si kulička zaznamená, že byla aktivovaná. Po aktivaci se má kulička jinak chovat a vykreslovat. S každým kreslením po aktivaci kuličky se berou pro kreslení thorns ring jiné data. Jsou to data, které jsem si na začátku hry vypočítal 4.3.3. Po určitém počtu vykreslení (50) kulička završí svou animaci a zmizí. Při každém kreslení kuličky je volána metoda, jež praská kuličky, které se dotknou trnů. Tento způsob animace má nevýhodu, že pokud nebude nějaké zařízení schopno renderovat 60 FPS, bude animace trvat déle. Ale pokud bych chtěl udělat rychlost animace nezávislou na FPS, musel bych data pro thorns ring počítat vždy před kreslením, tedy v reálném čase, což mi za to nestojí (nenašel jsem zařízení, které by nestíhalo kreslit).

4.10.3 Pulsující Tap to start

Abych dosáhnul pulsujícího nápisu v menu „Tap to Start“, musím každé kreslení měnit jeho velikost. Velikost měním periodicky na základě výsledku absolutní hodnoty funkce sinus. Funkci sinus předávám jako parametr úhel v rozsahu 0 – 360° (uvědomuji si, že by stačilo počítat do 180°). Tento úhel počítám pomocí funkce vracející reálný čas a funkce modulo.

4.10.4 Menu animace

Při menu animaci se před každým kreslením počítají dvě transformační matice (matrix), které se využívají k umístění a zmenšení objektů při animaci. Je to translační a scale matice. Scale matrix se aplikuje na všechny objekty, translační pouze na ty, na které ještě nebyla použita jiná translace (na všechny kromě kuliček a textu v menu). Pozice kuliček a textu v menu se při animaci dopočítá jinak.

Kapitola 5

Testování hry

Hru jsem testoval ve třech fázích. První testování proběhlo na MVP. Ve hře jsem ještě neměl thorns ball, ukazatel skóre ani funkce v menu. Druhé testování byl uzavřený Alpha test, kdy jsem testoval hlavně Google play games services a hratelnost hry. Třetí test je otevřená Beta, která běží do teď.

5.1 Testování minimum viable product

Cílem tohoto testování bylo zjistit názor na tuto hru, zdali o ni budou mít lidé po vydání zájem a získat návrhy na vylepšení. S některými testujícími jsem diskutoval i o finální verzi hry.

Testování proběhlo na dvanácti osobách (kamarádech a členech rodiny).

5.1.1 Průběh testování

Testoval jsem metodou pozorování uživatele. Testujícím jsem dal do ruky mobil se zapnutou hrou s tím, že chci slyšet jejich názor na hru. Hlavní věc, kterou jsem sledoval, bylo, jak se jim ve hře daří, z čehož jsem se snažil usoudit, jak je nastavena obtížnost hry. Během hraní jsem testujícím kladl otázky: jaké další kuličky by do hry přidali, kam by umístili aktuální skóre, zdali by využili sdílet své skóre s přáteli, jak se jim líbí přechod do menu, ...

5.1.2 Vyhodnocení testování

Co se týče obtížnosti hry, potvrdil se mi předpoklad, že ji budu muset zvednout. I když dotazovaní hráči udávali, že tempo jim vyhovuje. Já si myslím, že jelikož jsem jim neměl jak v době testování tempo upravit, má jejich názor malou váhu. Většině hráčů nedělalo problém hrát jednu hru (kolo) více jak 3 minuty. Což se mi zdá být hodně na to, že hru hráli poprvé. Chtěl bych, aby hráč neměl šanci stihnout zničit nově příchozí kuličku do doby, než přijde další. Tudíž aby se kuličky na hrací ploše hromadily. K vyčištění plochy by měl poté sloužit thorns ball, který ničí všechny kuličky ve svém okolí.

Na otázku, kam umístit aktuální skóre, jsem dostával rozdílné odpovědi: levý horní roh, pravý horní roh, na hoře ve středu a nejzajímavější: ve středu, pod číslem zobrazující aktuální počet kuliček na ploše. Přemýšlel jsem také o tom, jestli nezobrazovat aktuální skóre jiným způsobem. Například změnou barvy pozadí. Ovšem pokud bych se rozhodl pro proměnlivou barvu pozadí, eventuálně barvu míčků, už by asi nemělo smysl odměňovat hráče za dosažené achievmenty balíčky měnícími vzhled hry.

Na otázku, zdali by hráči využili možnost sdílet své skóre s přáteli, jsem se nejspíš ptal moc brzo (skóre jsem v době testování ještě neměl implementované). Tudíž jsem dostával odpovědi, že nevidí důvod pro sdílení. To mě utvrdilo, v tom že hra momentálně není pro hráče výzvou. A jelikož se chci držet prvotního plánu, udělat hru těžkou, vedoucí hráče k soutěžení, musím razantně zvednou obtížnost.

Jelikož byla hra v době testování v rané fázi vývoje, dostal jsem spoustu námětů, co do hry přidat a jak by podle testujících měla vypadat. Ovšem s většinou návrhů jsem již počítal a ostatní mě nezaujaly.

5.2 Uzavřený alpha test

V době tohoto testování jsem již byl s obtížností hry spokojen, ale chtěl jsem na ni znát názor ostatních lidí. Funkce v menu jsem již měl hotové, takže fungoval leaderboard, achievementy, hodnocení hry i zvuky.

5.2.1 Průběh testování

Jelikož jsem potřeboval vyzkoušet, zdali funguje hra a Google play games services i na jiných než virtuálních zařízeních a mém mobilu, vydal jsem hru v uzavřeném alpha testu na Google play. Uzavřený test znamená, že si hru můžou stáhnout pouze lidé, kterým to ručně povolím (whitelist). Otevřený test jsem nechtěl pouštět, protože bych musel vydat achievementy, které by poté již nešly změnit.

O testování jsem poprosil znovu kamarády a členy rodiny. Jelikož jsem se se všemi setkal i osobně, mohl jsem si na vlastní oči prohlédnout hru na jejich zařízeních.

5.2.2 Vyhodnocení testování

Po tomto testování se neobjevily žádné závažné problémy závislé na typu zařízení. Achievementy, které jsem do hry vložil byly všechny splnitelné. Leaderboard fungoval. Obtížnost se zdála být rozumně nastavena. Našlo se pár lidí, kterým se moc nelíbily barvy ve hře, tudíž jsem je změnil. Byl jsem překvapen, že někteří lidé hráli hru pro zábavu, a příliš se nesnažili překonávat své skóre. Měli maximální skóre kolem 120, a zdálo se, že se dále nezlepšují. Kdežto já a jiní soutěživěvější hráči jsme měli top skóre nad 400. To značí, že by hra mohla zaujmout větší okruh lidí.

5.3 Otevřený beta test

V otevřeném testu si již aplikaci může zahrát kdokoliv, kdo má na to zařízení a najde hru na Google Play. Dokud ale hra není vydaná, nezobrazuje se aktivně uživatelům, ale ti si ji musí sami vyhledat. Aby si hru někdo stáhl, musel jsem napsat na sociální sítě prosbu vyzkoušení a ohodnocení mé hry.

5.3.1 Vyhodnocení testování

Hru si v otevřeném beta testu stáhlo 38 lidí a dostal jsem více jak 10 zpětných vazeb. Lidé si stěžovali, že neví co mají ve hře dělat, že jim trvalo dlouho, než na to přišli, nebo že jim nejsou jasné určité principy hry (jaké kuličky praská thorns ball atd). Myslel jsem, že to, co mají hráči dělat, pochopí z názvu hry, který zní: Don't Go Over 16 a na zbytek si sami

rádi přijdou. Čekal bych, že tyto problémy by měli mít už lidé, kteří hru testovali ve dvou předešlých fázích, ale nikdo se o tom nezmínil. Pravděpodobně, pokud ho měli, tak se to styděli přiznat. Každopádně jsem usoudil, že do hry bude potřeba přidat nápovědu.

Další výtka byla k tomu, že tester neví, jakým způsobem prohrál. Radil mi po prohře zobrazoval okno, ve kterém bude zhodnocení hry a důvod, proč prohrál. Tento způsob lze vidět u většiny her, ale mě osobně se nelíbí, proto zůstanu u mého (myslím si, ve hrách revolučního řešení) animovaného přechodu ze hry do menu. Je ale v mém zájmu, aby hráč věděl, na základě čeho prohrál, proto jsem zpomalil animaci přechodu do menu.

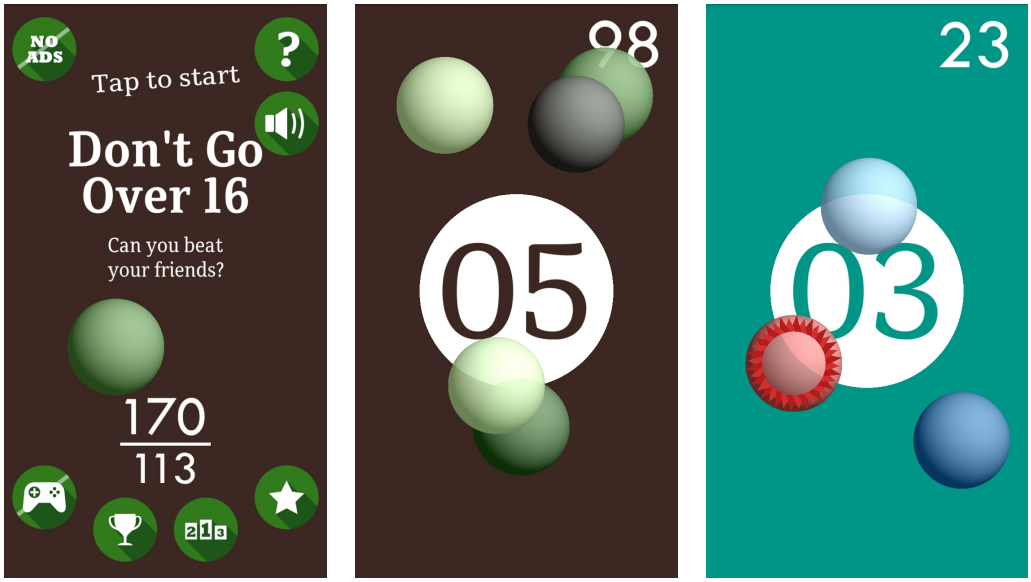
Bohužel jsem nenasbíral dostatek dat pro zobrazení statistik z Google play games services. Do Google play games services se přihlásilo 22 lidí z 38, kteří si ji stáhlo. Což pro zobrazení statistik nestačí. Větší reklamní kampaň jsem pro hru v beta fázi dělat nechtěl, protože si ji šetřím na ostré vydání, kdy se bude rozhodovat o tom, jaké hodnocení hra dostane a kolika lidem se bude zobrazovat.

5.4 Změny provedené na základě testování

Hlavní změnou provedenou na základě testování bylo přidání provizorní nápovědy, jelikož plnohodnotnou nápovědu jsem přidat nestihl. Za provizorní nápovědu považuji nápovědu zobrazitelnou po kliknutí na otazník v menu. Nápovědou, kterou bych v budoucnu chtěl implementovat je, že při prvním spuštění hry se při každé důležité události pozastaví hra a uživateli bude zobrazeno malé okno informující o dané události (když se objeví černá kulička, thorns ball, hráč prohraje atd.).

Ve všech fázích testů jsem taktéž měnil věci ovlivňující vzhled a chování hry. Jednalo se například o rychlost generování kuliček, velikost kuliček, barvu objektů, rychlosti animací, zobrazování skóre atd.

Obrázek 5.1: Screenshoty z fáze alpha testu



Obrázek 5.2: Screenshoty z finální verze hry

Kapitola 6

Závěr

Mým cílem v této bakalářské práci bylo vytvořit co nejkvalitnější hru specifikovanou v úvodu 1. To se mi do jisté míry povedlo. Vykreslil jsem a implementoval jsem chování hlavních objektů hry, konkrétně kuliček. Jednou z největších částí práce bylo implementovat efektivní renderování textu. U textu nešlo pouze o vykreslování, ale také o jeho pozicování a zarovnávání. Část, která mě na implementaci nejvíce bavila, byla animace přechodu ze hry do menu 4.10.4. Toto řešení ukončení hry jsem navíc dosud v žádné jiné hře neviděl. Pro implementaci achievementů a leaderboardu jsem použil API Google play games services 4.5. Věc, kterou jsem musel v souvislosti s achievementy a leaderboardem udělat, bylo ukládání dat lokálně 4.5.1 a jejich zabezpečení proti snadnému podvrhnutí.

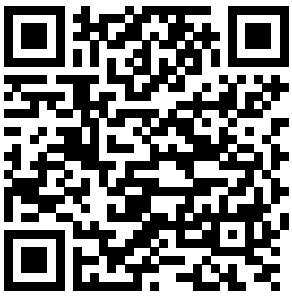
Hru jsem vydal na Google Play, ale pouze v beta verzi. Rozhodl jsem se ji nevydat v produkční verzi, dokud s ní nebudu naprosto spokojený, protože je velice důležité, jaké hra dostane na začátku hodnocení. Na hodnocení a počtu stažení se odvíjí, kolika lidem bude hra v Google Play zobrazována.

Počas testování se našli lidé, které hra bavila a hráli ji dlouhodobě. Tito lidé mě ujišťovali, že to, co dělám, má smysl a hra může být opravdu úspěšná. Na druhou stranu se našli i lidé, kteří si hru nainstalovali, nevěděli, co mají dělat, nebo je hra pouze nezaujala a vzápětí ji odinstalovali. Skutečnost, že nevěděli, co je ve hře jejich úkolem, mi byla sdělena ve zpětné vazbě. Na tu jsem reagoval implementací nápovědy. Že někoho hra nebaví jsem vydedukoval z výše nahraného skóre některých hráčů.

Věci, které jsem nestihl udělat, nebo je chci před vydáním udělat lépe jsou:

- Lepší nápověda (tutoriál), jak jsem již referoval v sekci 5.4.
- Přidat do hry více vzhledů, kterými budu hráče odměňovat za splněné achievementy. To je popsáno v sekci 4.9.
- V poslední řadě bych chtěl do hry přidat reklamy.

Hra je dostupná na Google play pod názvem Don't Go Over 16. Je možno ji vyhledat pomocí názvu hry, dále přes následující odkaz: `https://play.google.com/store/apps/details?id=com.games.smashthemall`, nebo přes QR kód 6.1.



Obrázek 6.1: QR kód vedoucí na hru Don't Go Over 16

Literatura

[1] Brothaler, K.: *Learn OpenGL ES*. [online], [navštíveno. 26.9.2016].
URL http://www.learnopengles.com/

[2] Brothaler, K.: *OpenGL ES 2 for Android*. The Pragmatic Bookshelf, 2013, ISBN 978-1937785345.

[3] Google, I.: *Canvas and Drawable API*. [online], [navštíveno. 26.9.2016].
URL https://developer.android.com/guide/topics/graphics/2d-graphics.html#draw-with-canvas

[4] Google, I.: *Displaying Graphics with OpenGL ES*. [online], [navštíveno. 26.9.2016].
URL https://developer.android.com/training/graphics/opengl/index.html

[5] Google, I.: *Google APIs for Android*. [online], [navštíveno. 12.4.2017].
URL https://developers.google.com/android/reference/packages

[6] Google, I.: *Google Play Games Services*. [online], [navštíveno. 24.2.2017].
URL https://developers.google.com/games/services/

[7] Google, I.: *OpenGl ES API*. [online], [navštíveno. 26.9.2016].
URL https://developer.android.com/guide/topics/graphics/opengl.html

[8] ReIndustries: *Android Programming Blog*. [online], [navštíveno. 26.9.2016].
URL http://androidblog.reindustries.com/

[9] The Khronos Group, I.: *OpenGl ES*. [online], [navštíveno. 26.9.2016].
URL https://www.khronos.org/opengles/

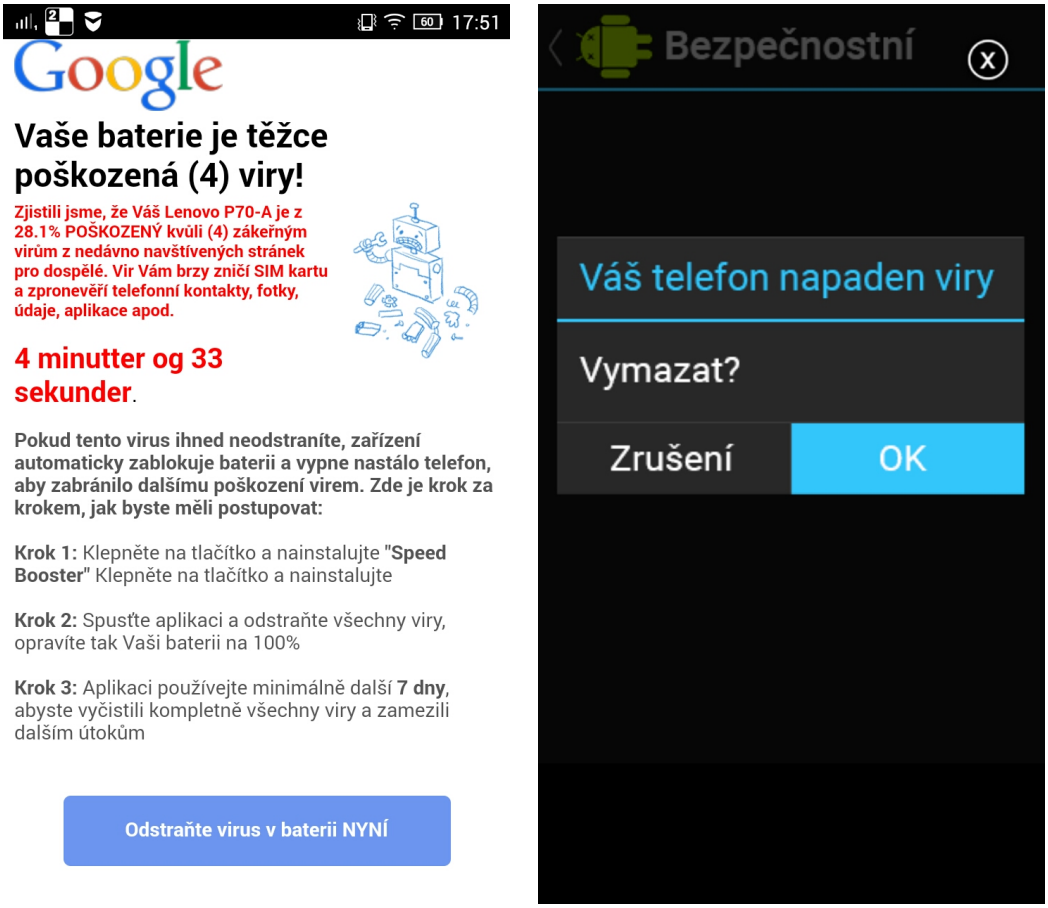
[10] The Khronos Group, I.: *OpenGL ES 2.0 Reference Pages*. [online], [navštíveno. 26.9.2016].
URL https://www.khronos.org/registry/OpenGL-Refpages/es2.0/

Přílohy

Příloha A

Nevhodné reklamy ve hře Ant Smasher

Z mého podhledu nevhodná reklama, která kazí hře jméno.



Obrázek A.1: reklamy zobrazované ve hře Ant Smasher.

Příloha B

Obsah CD

- **src** – adresář se zdrojovými kódy hry
- **dontGoOver16v105.apk** – instalační soubor hry
- **thesis** – adresář se zdrojovým tvarem písemné zprávy
- **thesis.pdf** – písemná zpráva ve formátu PDF
- **poster.pdf** – prezentační plakát
- **video.mp4** – prezentační video
- **readme.txt** – základní informace