|  |
| --- |
| Odisee |
| Project 2 |
| Android Applicatie met LibGDX |

|  |
| --- |
| Wouter Vande Velde  8-2-1995  Mentor: Mr. Demeester |

Contents

[Doelstelling 1 2](#_Toc471171350)

[Overzichtsrapport 2](#_Toc471171351)

[Algemeen 2](#_Toc471171352)

[Ontwikkeling 2](#_Toc471171353)

[Programmeertaal 2](#_Toc471171354)

[Hardware 3](#_Toc471171355)

[Doelstelling 2 3](#_Toc471171356)

[LibGDX 3](#_Toc471171357)

[Het spel 3](#_Toc471171358)

[Collision detection 4](#_Toc471171359)

[File i/o naar intern geheugen 4](#_Toc471171360)

[Graphics 4](#_Toc471171361)

[Camera 4](#_Toc471171362)

[Geluid 5](#_Toc471171363)

[Doelstelling 3 5](#_Toc471171364)

[Google API 5](#_Toc471171365)

[Leaderboard 6](#_Toc471171366)

[Opvolgingsdocument 7](#_Toc471171367)

[Bibliography 13](#_Toc471171368)

[Code 14](#_Toc471171369)

[Core module 14](#_Toc471171370)

[Android module 39](#_Toc471171371)

[Other 47](#_Toc471171372)

# Doelstelling 1

**De student maakt een overzichtsrapport en vergelijking over applicaties op verschillende besturingssystemen.**

## Overzichtsrapport

Het onderwerp van dit overzichtsrapport is applicaties op verschillende besturingssystemen. Er zal besproken worden wat de voordelen zijn van een bepaalde operating system en wat de nadelen zijn. Hoofdzakelijk zullen Android en IOS besproken worden in dit overzichtsrapport en hoe applicaties op deze besturingssystemen draaien.

### Marktaandeel

Een applicatie op IOS kan enkel verkregen worden via de App Store in ITunes terwijl een applicatie op Android verkregen kan worden via de Play Store, Amazon, … of je kan het zelf toevoegen. Dit heeft als voordeel dat er veel meer applicaties beschikbaar voor Android zijn terwijl de applicaties op IOS beter beheerd en gecontroleerd worden en dat deze dus zelden malware zullen bevatten.

In 2014 hadden 84.7% van de smartphones op de markt Android als besturingssysteem terwijl Apple maar 11.7% had. Dit betekent dat applicaties voor Android een veel groter doelpubliek zullen bereiken. Maar ongeacht dit grote verschil is het Apple die in 2014 nog de meeste inkomsten had via applicaties. [1]

**Vastellingen**

|  |  |  |
| --- | --- | --- |
|  | Android | IOS |
| Apps verkrijgen | Play store, Amazon,… | ITunes |
| Marktaandeel | 84.7% | 11.7% |
| Inkomsten | IOS-inkomsten 70% hoger dan Android | |

### Programmeertaal

IOS-applicaties worden geschreven in Swift, dit is een nieuwe taal geschreven door Apple speciaal voor IOS en OS X. Swift is nog een relatief nieuwe taal wat betekend dat het nog steeds volop in ontwikkeling is. Maar Swift is ook backwards-compatibel met zijn voorganger Objective-C wat betekend dat het toch veel libraries ter beschikking heeft. Een voordeel van deze taal is dat het een veilige taal is[2].

Android-applicaties worden geschreven in Java, het voordeel van deze taal is dat het op alle verschillende modellen van Android kan gedraaid worden. [3] Dit komt omdat Java gerund wordt in een virtual machine, ongeacht de onderliggende architectuur. Een sterk punt van Java is dat het gebruik maakt van stack allocatie en het heeft een automatische garbage collection. Een groot nadeel van Java is dat het redelijk traag is, zowel in het compilen als het runnen.

**Vastellingen**

|  |  |  |
| --- | --- | --- |
|  | Android | IOS |
| Programmeertaal | Java | Swift |

### App releasen

Qua ‘gebruiksvriendelijkheid’ voor een programmeur om de app te releasen op de Play Store tegenover ITunes is er een enorm verschil. Om een applicatie op de Play Store te zetten moet je eenmalig 25 euro betalen om toegang te krijgen. De prijs om een applicatie op ITunes te zitten is 100 euro per jaar. Eens het gereleaset is, vragen zowel Google als Apple 30% van de verkoop inkomsten. [4] Een ander gegeven is dat de review-periode voor dat de applicatie online staat, veel korter is bij Google dan bij Apple. Bij Android is dit een paar uur terwijl bij Apple dit gemiddeld 3 dagen is. [5]

**Vastellingen**

|  |  |  |
| --- | --- | --- |
|  | Android | IOS |
| Prijs | Eenmalig €25.00 | Jaarlijks €100.00 |
| Taxen | 30% van voorkoop inkomsten | 30% van voorkoop inkomsten |
| Tijd tot release | Een paar uur | 3 dagen |

### Smartwatch

Voor sommige applicaties is het interessant om een smartwatch te gebruiken. Alhoewel Apple geen nummers vrijgeeft over de verkoop van zijn smartwatches, kan ervan uit gegaan worden dat het een groot publiek bereikt door de vaste userbase. Analysten schatten dan ook dat Apple een aandeel 46% van de markt van smartwatches heeft. De watchOS is makkelijk om te voor programmeren en het wordt voortdurend verder ontwikkeld wat toont dat het een mooie toekomst heeft. [6]

Voor Android Wear gaat het daarentegen iets minder vlot, de release van versie 2.0 werd uitgesteld midden of eind 2017 en sommige partners zoals Motorola hebben al officiële statements gemaakt waarin ze Android Wear afzweren. Versie 1.0 van Android Wear heeft dan ook verschillende problemen, de OS voelt alsof het nog in ontwikkeling is, het heeft een lage battery life, … [6].

Programmeren op zowel Android Wear als watchOS is niet moeilijk, het is gelijkaardig als respectievelijk een Activity en een View te maken zoals bij een gewone applicatie gebeurd.

Apple lijkt dus de strijd te winnen op vlak van Smartwatches.

**Vastellingen**

|  |  |  |
| --- | --- | --- |
|  | Android | IOS |
| Ontwikkeling | Nog veel werk aan | Goede OS |
| Toekomst | Versie 2.0 lijkt veelbelovend maar verschillende partners haken af | Voortdurende ontwikkeling |
| Programmeren | Makkelijk | Makkelijk |

### Hardware

Bij sommige applicaties is het belangrijk dat de hardware die aanwezig is in het toestel de applicatie kan runnen. Het voordeel van IOS is dat het maar op weinig verschillende type devices gerund moet worden, de IPhone, IPad, … Dit betekent dat de specificaties van het model enkel zullen verschillen onder de verschillende modellen. Dit is volledig tegengesteld bij Android, er zijn talloze verschillende toestellen die Android draaien met allemaal verschillende specificaties. De specificaties van nieuwe modellen van iPhone of IPad zijn uitstekend waardoor je zeker kan zijn dat de applicatie gedraaid zal kunnen worden.

**Vastellingen**

|  |  |  |
| --- | --- | --- |
|  | Android | IOS |
| Modellen | Zeer veel verschillende | 1 type |
| Specificaties | Verschillen enorm | Goede specificaties |

# Doelstelling 2

**Hij schrijft een afgewerkte Android-applicatie met behulp van libgdx die volgende concepten uitgebreid illustreert: collision detection, file i/o naar intern geheugen, graphics met OpenGL ES 2.0, toevoegen van geluid, gebruik maken van verschillende motion gestures,...**

## LibGDX

LibGDX is een gratis en open-source framework om games te ontwikkelen, het werd gelanceerd begin 2014. Het sterkste punt van het framework is dat het toestaat om dezelfde code te gebruiken voor verschillende platforms zoals Linux, Windows, IOS, Android… Andere features van LibGDX zijn onder andere dat het gebruik maakt van OpenGL, 2D en 3D ondersteuning aanbiedt, … Alle Features zijn te lezen op de pagina ‘Goals and Features’ op de website van LibGDX [4] . Om meer over de library te leren heb ik volgende site gebruikt, <http://www.gamefromscratch.com/page/LibGDX-Tutorial-series.aspx>. En bovendien heb ik nog verschillende andere tutorials en voorbeelden bekeken.

Elk scherm dat zichtbaar is tijdens het spel noemt een State. Deze states zijn klassen en hebben allemaal een constructor, een methode Update, een methode Render en een methode Dispose. In de constructor wordt alle waardes geïnitialiseerd, in de render methode alles getekend, in update worden bepaalde waardes berekend en in dispose worden de textures gewist als ze niet meer nodig zijn. De methodes render en update worden in een loop steeds opnieuw aangeroepen.

## Het spel

Het spel is een alternatieve versie van Flappy bird, er zijn buizen waarop geklikt moet worden en munten die je kan pakken voor extra punten. Het puntensysteem is ook anders dan bij het originele spel en het spel is in landschapsmodus.

Volgende concepten zijn vermeld in de doelstelling en toegevoegd aan het spel.

### Collision detection

De vogel en de andere textures hebben allemaal grenzen gekregen in de vorm van een rechthoek. LibGDX heeft een methode genaamd Overlap die kijkt of 2 textures over elkaar komen. Zodra de vogel een buis raakt, is het spel gedaan.

### File i/o naar intern geheugen

In de applicatie wordt er gebruik gemaakt van de Shared Preferences voor een lokale highscore bij te houden. LibGDX voorziet een klasse genaamd Preferences die de Shared Preferences klasse van Android zal aanspreken. In deze preferences worden de highscore opgeslagen, de eigenaar van de highscore, de score na elk spelletje en de huidige speler.

*preferences* = Gdx.*app*.getPreferences(**"com.project2.prefs"**);

Naast de speciale klasse in de library van LibGDX wordt in de activity GoogleApiActivity ook gebruik gemaakt van de standaard Shared Preferences.

SharedPreferences *prefs* = **this**.getSharedPreferences(**"com.project2.prefs"**, Context.***MODE\_PRIVATE***);  
SharedPreferences.Editor editor = *prefs*.edit();  
editor.putString(**"currPlayer"**, **currPlayer**);

editor.commit();

### Graphics

#### OpenGL ES

OpenGL graphics zijn niet geïmplementeerd. OpenGL biedt graphics met een hoge kwaliteit en hoge performance. De graphics die nu gebruikt zijn in het spel zijn van een lage kwaliteit, het is dus wel iets waar ik in de toekomst meer aandacht aan moet besteden.

#### G2D

De huidige graphics zijn geïmplementeerd aan de hand van G2D van LibGDX. Dit houdt in dat er een spritebatch werd aangemaakt bij het opstarten. Hiervan wordt de camerapositie gezet, daarna worden textures toegevoegd en uiteindelijk wordt deze spritebatch terug gesloten en getekend.

sb.setProjectionMatrix(**cam**.**combined**);sb.begin();  
sb.draw(**background**,0,0);  
sb.draw(**logo**, **cam**.**position**.**x**-**logo**.getWidth() / 2,**cam**.**position**.**y**-**logo**.getHeight() / 2);  
sb.end();

Op het einde van elke state worden alle gebruikte graphics gewist om geheugen te besparen.

### Camera

Als camera wordt er een orthographic camera gebruikt, de breedte en hoogte van de camera zijn 2 vaste constanten. Tijdens de PlayState wordt deze camera verschoven naar rechts, mee met het verloop van het spel en de x-waarde zal dus steeds stijgen.

### Geluid

LibGDX voorziet een klasse Music en een klasse Sound, met volgende lijn code wordt een muziekbestand uitgelezen en op de achtergrond afgespeeld.

Music music = Gdx.*audio*.newMusic(Gdx.*files*.internal(**"music.mp3"**));

Bij elke tik op het scherm en dus elke keer dat de vogel springt wordt er ook een geluid afgespeeld met volgende lijn code.

Sound flap = Gdx.*audio*.newSound(Gdx.*files*.internal(**"sfx\_wing.ogg"**));

# Doelstelling 3

**Hij zal de gebruiker laten inloggen met een googleaccount, gebruik makend van de Google API en een leaderboard voorzien, alsook gebruik makend van de Google API.**

## Google API

Een van de einddoelen van het project was om de gebruiker te laten inloggen met Google en een Google Leaderboard te tonen. Hiervoor was het plan om gebruik te maken van 2 verschillende API’s, namelijk de Auth API en de Games API.

Om connectie te maken met een API moeten er een paar dingen geiplementeerd worden. De klasse moet eerst en vooral GoogleApiClient.ConnectionCallbacks en GoogleApiClient.OnConnectionFailedListener implementeren. Deze klasses zijn nodig om een googleApiClient te kunnen ‘bouwen’. Dit gebeurd met volgende code:

**mGoogleApiClient** = **new** GoogleApiClient.Builder(**this**)  
 .addConnectionCallbacks(**this**)  
 .enableAutoManage(**this**, 0 */\* clientId \*/*, **this**)  
 .addApi(Games.***API***).addScope(Games.***SCOPE\_GAMES***)  
 *// add other APIs and scopes here as needed* .build();

De 2 klasses die geimplementeerd worden, bevatten volgende methodes:

* onStart(): deze methode wordt automatisch na de start uitgevoerd, hierin wordt ook de connectie gelegd met de online service van Google met volgende code:

mGoogleApiClient.connect();

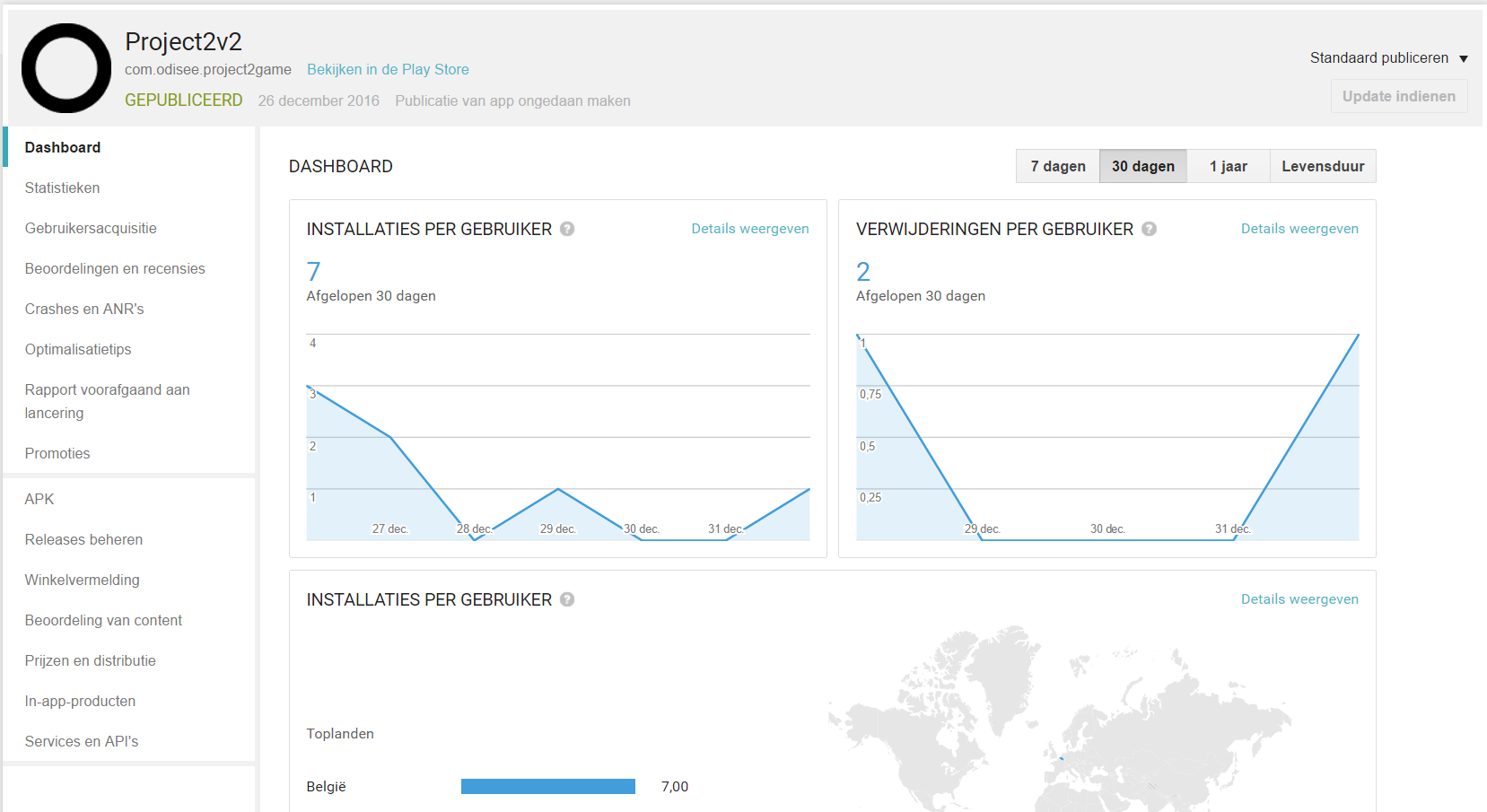
* OnConnected: deze methode wordt aangeroepen als de connectie geslaagd is, hierin zal dan ook het leaderboard opgeroepen worden en zullen scores naar het leaderboard verzonden worden.
* onConnectionFailed(): deze methode wordt aangeroepen als de connectie in onStart() mislukt is, er zal geprobeerd worden om een scherm aan te roepen die de gebruiker opnieuw zou laten inloggen indien dit nog niet gebeurd zou zijn.

### Leaderboard

Het leaderboard moest aangemaakt worden op de Google Developers Console [5]. De Google Developers Console is een web applicatie die helpt om applicaties te managen, hier kan je apps uploaden, product pagina’s opstellen, cijfers zien over je app, …

Voordat het leaderboard dat aangemaakt is, kan werken in de applicatie moet de app eerst gepubliceerd zijn in de Developers Console ( dit is niet hetzelfde als publiceren op de Play Store ) en een Client-API aangemaakt worden. Deze Client-API wordt aangemaakt op de Google API Console [6].

Hieronder zie staat een foto van de gepubliceerde app in de Google Developers Console.



De applicatie op de Google Developers Console moet gelinkt worden met de code op je computer om te kunnen debuggen en connectie te kunnen maken met de Games API. Dit gebeurt aan de hand van de sha-1 vingerprint van de debug-apk in Android Studio en de package name. Elk project in Android Studio heeft een unieke sha-1 en als de app erop gerund wordt, wordt er gekeken of de sha-1 van het project overeenkomt met de sha-1 die toegevoegd is op de Developers Console.

Eens de applicatie afgewerkt was, heb ik hem op de Google Play Store gezet. Om het leaderboard in de release versie te laten werken moest er een tweede Sha-1 toegevoegd worden, namelijk deze van de release-APK. Deze kon verkregen worden met volgend commando:

keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android

Voor de release van de applicatie moest er onder andere een omschrijving toegevoegd worden, een release APK geüpload worden, prestaties toegevoegd worden, screenshots, … Ook het pictogram van de app heb ik veranderd in het logo van Odisee.

De applicatie staat nog steeds op de playstore, te vinden onder link [7] in de bibliografie.

# Opvolgingsdocument

**=================================================================================**

**Project 2 - opvolgingsdocument Academiejaar 2016-2017**

**=================================================================================**

Student: Vande Velde Wouter

Mentor: Peter Demeester

**=================================================================================**

**Beschrijving - datum 20 september 2016**

**=================================================================================**

Onderwerp:

Een 2D-spelletje voor een Android device gemaakt in Android studio met behulp van de Libgdx library.

(Iets in de aard van Flappy Bird)

Einddoelen:

1. <Wouter Vande Velde> - <30/09> De student maakt een overzichtsrapport en vergelijking over applicaties op verschillende besturingssystemen.

<mentor> OK

2. <Wouter Vande Velde> - <30/09> Hij schrijft een afgewerkte Android-applicatie met behulp van libgdx die volgende

concepten uitgebreid illustreert: collision detection, file i/o naar intern geheugen, graphics met OpenGL ES 2.0, toevoegen van geluid,

gebruik maken van verschillende motion gestures,...

<mentor> ok

3. <Wouter Vande Velde> - <30/09> Hij zal de gebruiker laten inloggen met een google account, gebruik makend van de Google API

en een leaderboard voorzien, alsook gebruik makend van de google API.

Motivatie - betrokkenheid:

Ikzelf speel regelmatig spelletjes waaronder spelletjes op de smartphone, via dit project wil ik de beter begrijpen hoe een Android-applicatie en een spel op een

Android-device ineenzitten.

Motivatie - grenzen verleggen:

Een gewone Android-Applicatie leek wat simpel en niet uitgebreid genoeg als project dus lijkt een bewegend en dynamisch spelletje toepasselijker en uitdagender.

Motivatie - professioneel profiel:

Aangezien bijna iedereen in deze tijd een smartphone heeft, zijn apps belangrijk in het dagelijkse leven,

het is interresant om hier kennis over op te doen en deze wereld te ontdekken.

Daar bijkomend is 'games' de meest populaire categorie van applicaties in de google play store.

Bovendien is het begrijpen en toepassen van een onbekende library zoals libgdx iets wat waarschijnlijk vaak zal voorkomen in een carrière.

kerndoelen: 1.a, 4,a, 5.c, 8.i 10.n

Aan de hand van onlineinformatie zal de student een overzicht maken van de kernfunctionaliteiten en features van libgdx.

Dit overzicht gebruikt hij om de basis van de app te kunnen leggen.

Hiervoor wordt er gebruik gemaakt van de uitgebreide informatie op de site van android en libgdx en verschillende tutorials op het internet.

Gaandeweg zal de applicatie vorm krijgen en dynamisch worden. eens de applicatie werkt, kan de design verzorgd worden.

Uiteindelijk zal inloggen via google mogelijk gemaakt worden en zal ook een leaderboard toegevoegd worden.

**=================================================================================**

**Planning - 7/10/2016**

**=================================================================================**

30 oktober

- onderdelen van de library die nodig zijn voor de applicatie begrijpen en kunnen toepassen

- idee hebben voor hoe het spel er zal uitzien, wat zal gemaakt worden

30 november

- applicatie zelf grotendeels afwerken

- audio toevoegen

- overzichtsrapport deels afwerken (einddoel 1)

15 december

- toevoegen van google scorebord en inloggen

- overzichtsrapport afwerken

**=================================================================================**

**Logboek**

**=================================================================================**

Evaluatie 1

26 oktober

- project aangemaakt en gesynct naar git

- beginnen aan tutorial

- verschillende states aangemaakt zoals menu en playstate

- Een bewegend geheel waarin het vogeltje vliegt en de vleugels animeren

27 oktober

- collision detection

- begrijpen hoe memory leaks te voorkomen

- audio toevoegen

28 oktober

- voorbeeld afgewerkt

----------------------

Evaluatie 2

14 november

- start implementatie google login

- nieuwe activity oproepen vanuit libgdx

16 november

- login afgewerkt en succesvol

- login bij startup oproepen

21 november

- score implementeren en tonen tijdens het spelen

- endstate toegevoegd

27 november

- tweede activity aanroepen voor leaderboard

- start games api google toevoegen voor leaderboard

- Google Play Developers Console aangemaakt de app gepublished

30 november

- update gameplay, buizen bewegen

-------------------

12 december

- Toevoegen lokale highscore met behulp van shared preferences

- Auth API verwijderd uit het project en enkel Games overgehouden

13 december

- poging tot maken van apk

- start overzichtsrapport maken

16 december

- google leaderboards werken niet meer

19 december

- portfolio toegevoegd

22-23 december

- nieuw project gemaakt op aanraden van Mr Demeester, alles werkt terug

25 december

- update gameplay, coins en klikbare buizen

26 december

- app toegevoegd aan de playstore

27 december

- bug met leaderboards in release APK opgelost

**=================================================================================**

**Eerste tussenrapportering - 29/10/2016**

**=================================================================================**

Stand van zaken:

Het project wordt correct en compleet gesyncd naar git (wat toch wel een lastige zaak bleek). Dit betekent dat het gedownload kan worden en

uitgevoerd worden op een andere computer.

Android studio is correct geconfigureerd en het spel kan op de desktop gedraaid worden en mijn smartphone is ook correct geconfigureerd zodat ook daarop getest kan worden.

De tutorial is afgewerkt en de basics zijn onder de knie.

De applicatie is op dit moment een werkende, maar nog niet afgewerkte, versie van Flappy bird waarin het spel deftig gespeeld kan worden.

Er zijn ook al verschillende states in aanwezig zoals de menustate en playstate maar deze moeten nog verder afgewerkt worden.

Kort gezegd is de alle voorbereiding afgerond en kan er nu gefocust worden op het creëren van de applicatie.

Reflectie:

De aanpak werkt goed, ik snap tot nu toe alles wat aangehaald wordt in de tutorial en het meeste zal ook van toepassing

zijn in de applicatie die ik zal maken. Hetgene dat nu al gerealiseerd is, is al een mooie applicatie.

De motivatie kwam redelijk laat om aan het project te beginnen maar nu ik eenmaal begonnen ben en het wat snap, is de motivatie toch gekomen.

Zelfevaluatie:

Nieuw verworven kennis A

Toepassing aangetoond B

Planning B

Analyse noden NVT

initiatief - inzet C

rapportering B

Feedback vanwege mentor - <datum van ingave>:

<Hier komt de feedback en opmerkingen vanwege de mentor>

<mentor>volgens je logboek en wat je hierboven schrijft, zit je op schema. Ik vind het wel raar dat je laatste commit ondertussen 15 dagen geleden is. Ik hoop dat je nog altijd

voldoende motivatie hebt om verder te doen aan je app? Je geeft jezelf een C voor initiatief en inzet? Kan je dit een beetje uitleggen?

Hoever sta je met einddoel 1?

Werk regelmatig je logboek bij.

Veel succes!

**=================================================================================**

**Tweede tussenrapportering - 2/12/2016**

**=================================================================================**

Stand van zaken:

Beide API's (leaderboard=gamesAPI en inloggen) zijn geïmplementeerd in de applicatie. Ik had nadien pas gezien dat de authenticatie al automatisch gebeurd via de games API

van google maar heb de authentication API toch laten staan. De Games API zou bovendien volledig moeten werken maar blijkbaar kan deze niet getest worden vanuit een

development environment dus weet ik niet zeker of het correct werkt. Dit ga ik zo snel mogelijk testen, vermoedelijk door een beta te lanceren van de app op de playstore.

Op vlak van de gameplay is alles ook al goed gevorderd, er zijn al bepaalde dingen toegevoegd maar ik plan om er nog extra elementen in te steken (zoals bv bonus coins

en wegklikbare obstakels). Er zijn bovendien al google achievements aangemaakt voor tijdens het spelen maar deze zal ik implementeren als ik tijd over heb.

Ik ben nog niet begonnen aan het overzichtsrapport, ik wou eerst nog wat werk steken in de applicatie.

Reflectie:

Ik denk dat het werk redelijk goed vooruit gaat, alles is tot nu toe gelukt. Ik had op vlak van API misschien eerst wat meer opgezocht want blijkbaar heb je de

authenticatie API niet meer nodig als je de games API gebruikt. Ik heb ook verschillende keren lang vastgezeten maar telkens heb ik dit kunnen oplossen, zo is het bijvoorbeeld

zeer lastig om een nieuwe activity op te roepen vanuit LibGDX. De motitie is ook helemaal terug en ik vind het leuk en interresant om aan het project te werken.

Zelfevaluatie:

Nieuw verworven kennis A

Toepassing aangetoond A-B

Planning B

Analyse noden NVT

initiatief - inzet A

rapportering B

Feedback vanwege mentor - 7/12/2016:

<Hier komt de feedback en opmerkingen vanwege de mentor>

Ik denk dat je je misschien voor rapportering een beetje teveel punten gegeven hebt, aangezien je aangeeft nog niet begonnen te zijn met het overzichtsrapport. Begin hier op tijd aan, want anders wordt het "iets dat er op het einde bij komt" en dat maar half afgewerkt is.

Blijkbaar was je even de motivatie kwijt, maar goed dat je die dus teruggevonden hebt ;-)

Kan je al een demo geven van je app? Tijdens de examens krijg je een kwartier tijd om alles te tonen: portfolio, demo en vragen beantwoorden. Misschien is dat wat kort om ook nog je demo helemaal te tonen. Je kan - als je dat wilt - je demo ook vroeger tonen. In lesweek 12 en 13 heb ik wel wat tijd.

Voor de rest lijkt me je zelfevaluatie redelijk correct te zijn.

**=================================================================================**

**Eindrapportering - 08/01/2017**

**=================================================================================**

Eindrapportering:

De app is afgewerkt. Er kan ingelogd worden met google en het leaderboard wordt weergegeven,en er kan een alternatieve versie van flappy bird gespeeld worden.

Het gehele resultaat is ook geupload op de playstore en beschikbaar op volgende link:

https://play.google.com/store/apps/details?id=com.odisee.project2game&hl=nl

Ook het overzichtsrapport is afgewerkt.

Reflectie:

Alle originele einddoelen zijn bereikt, de app is afgewerkt met login en er is een overzichtsrapport geschreven.

Qua design kan de app nog een pak beter, het vogeltje is bijvoorbeeld heel vlug getekend omdat er copyright op het eerste vogeltje stond.

Ook het menu is niet duidelijk dus daar moeten de knoppen nog opnieuw gemaakt worden.

Qua gameplay zijn er ook fouten opgedoken bij vrienden die de app gebruiken op iets oudere modellen,

de locatiewaardes die bepaald worden door libgdx wanneer het scherm aangeraakt wordt blijken soms te verschillen.

Een ander probleem is dat de vorm van de vogel een cirkel/ovaal is en de buizen rectangles zijn,

via libGDX kan geen collision waargenomen worden tussen een rectangle en een cirkel dus is de hitbox van de vogel iets te groot.

Het gebruik van de API ging ook niet optimaal, ik had eerst de verkeerde api gebruikt en eenmaal ik de juiste api gebruikte, traden er onverklaarbare fouten op.

Er zijn ook veel details die nodig zijn voor de API correct te laten werken maar die nergens in de tutorials vermeld staan.

Als ik het project opnieuw maakte in een poging om het te laten werken, lukt het van de eerste keer.

Volgens mij heb ik het project ook wel goed aangepakt. Misschien had ik wel eerst meer moeten opzoeken over de api zodat ik niet eerst een verkeerde zou implementeren.

Inhoud Portfolio:

Doelstelling 1

Overzichtsrapport

Algemeen

Ontwikkeling

Programmeertaal

Hardware

Doelstelling 2

LibGDX

Het spel

Collision detection

File i/o naar intern geheugen

Graphics

Camera

Geluid

Doelstelling 3

Google API

Leaderboard

Opvolgingsdocument

Bibliography

Code

Bronnenlijst:

|  |  |
| --- | --- |
| [1] | G. Sims, „Google Play Store vs the Apple App Store: by the numbers (2015),” AndroidAuthority, 20 april 2015. [Online]. Available: http://www.androidauthority.com/google-play-store-vs-the-apple-app-store-601836/. |
| [2] | P. Rubens, „10 Things You Should Know About Apple's Swift,” CIO, 21 juli 2014. [Online]. Available: http://www.cio.com/article/2456100/mobile-development/10-things-you-should-know-about-apples-swift.html. |
| [3] | M. Klaus, „Advantages and Disadvantages of Java Application Development,” Selfgrowth, [Online]. Available: http://www.selfgrowth.com/articles/advantages-and-disadvantages-of-java-application-development. |
| [4] | M. Zechner, „Goals and Features,” LibGDX, 2013. [Online]. Available: https://libgdx.badlogicgames.com/features.html. |
| [5] | Google, „Google Developers Console,” Google, [Online]. Available: https://play.google.com/apps/publish/?dev\_acc=16525306877658257287#AppListPlace. |
| [6] | Google , „Google API Console,” Google , [Online]. Available: https://console.developers.google.com/apis/credentials?project=project2-f5a95&hl=nl. |
| [7] | W. V. Velde, „Project2v2,” Odisee, [Online]. Available: https://play.google.com/store/apps/details?id=com.odisee.project2game&hl=nl. |
| [8] | Google, „Start Integrating Google Sign-In into Your Android App,” Google, 26 oktober 2016. [Online]. Available: https://developers.google.com/identity/sign-in/android/start-integrating. |

Zelfevaluatie:

Nieuw verworven kennis A

Toepassing aangetoond A-B

Planning B

Analyse noden NVT

initiatief - inzet A

rapportering B

# Bibliography

|  |  |
| --- | --- |
| [1] | G. Sims, „Google Play Store vs the Apple App Store: by the numbers (2015),” AndroidAuthority, 20 april 2015. [Online]. Available: http://www.androidauthority.com/google-play-store-vs-the-apple-app-store-601836/. |
| [2] | P. Rubens, „10 Things You Should Know About Apple's Swift,” CIO, 21 juli 2014. [Online]. Available: http://www.cio.com/article/2456100/mobile-development/10-things-you-should-know-about-apples-swift.html. |
| [3] | M. Klaus, „Advantages and Disadvantages of Java Application Development,” Selfgrowth, [Online]. Available: http://www.selfgrowth.com/articles/advantages-and-disadvantages-of-java-application-development. |
| [4] | M. Zechner, „Goals and Features,” LibGDX, 2013. [Online]. Available: https://libgdx.badlogicgames.com/features.html. |
| [5] | Google, „Google Developers Console,” Google, [Online]. Available: https://play.google.com/apps/publish/?dev\_acc=16525306877658257287#AppListPlace. |
| [6] | Google , „Google API Console,” Google , [Online]. Available: https://console.developers.google.com/apis/credentials?project=project2-f5a95&hl=nl. |
| [7] | W. V. Velde, „Project2v2,” Odisee, [Online]. Available: https://play.google.com/store/apps/details?id=com.odisee.project2game&hl=nl. |
| [8] | Google, „Start Integrating Google Sign-In into Your Android App,” Google, 26 oktober 2016. [Online]. Available: https://developers.google.com/identity/sign-in/android/start-integrating. |

# Code

## Core module

**Game.java**

**public class** Game **extends** ApplicationAdapter {  
 **private static** Preferences *preferences*;  
 SpriteBatch **batch**;  
 **public static final int *HEIGHT*** = 480;  
 **public static final int *WIDTH*** = 800;  
  
 **private** GameStateManager **gsm**;  
 **private** Music **music**;  
  
 *// interface to call the activity to log in* **public interface** MyGameCallback {  
 **void** onStartActivityLogin();  
 }  
  
 *// \*\* Additional \*\*  
 // Setter for the callback* **public void** setMyGameCallback(MyGameCallback callback) {  
 *myGameCallback* = callback;  
 }  
  
 *// Local variable to hold the callback implementation* **public static** MyGameCallback *myGameCallback*;  
  
  
 @Override  
 **public void** create () {  
 **batch** = **new** SpriteBatch();  
 **music** = Gdx.audio.newMusic(Gdx.files.internal(**"music.mp3"**));  
 **gsm** = **new** GameStateManager();  
 **music**.setLooping(**true**);  
 **music**.setVolume(0.1f);  
 **music**.play();  
 Gdx.gl.glClearColor(1, 0, 0, 1); *// wipes the screen and redraws everything* **gsm**.push(**new** LoginState(**gsm**));  
 }  
  
 @Override  
 **public void** render () {  
 Gdx.gl.glClear(GL20.GL\_COLOR\_BUFFER\_BIT);  
 **gsm**.update(Gdx.graphics.getDeltaTime());  
 **gsm**.render(**batch**);  
 }  
   
 @Override  
 **public void** dispose () {  
 **music**.dispose();  
 }  
  
 **public static** Preferences getPrefs() {  
 **if**(*preferences*==**null**){  
 *preferences* = Gdx.app.getPreferences(**"com.project2.prefs"**);  
 }  
 **return** *preferences*;  
 }  
  
}

**State.java**

**public abstract class** State {  
 **protected** OrthographicCamera **cam**;  
 **protected** Vector3 **mouse**;  
 **protected** GameStateManager **gsm**;  
  
 **public** State(GameStateManager gsm) {  
 **this**.**gsm** = gsm;  
 **cam** = **new** OrthographicCamera();  
 **mouse** = **new** Vector3();  
 }  
  
 */\*\*  
 \* handle the input  
 \*/* **protected abstract void** handleInput();  
  
 */\*\*  
 \* update values, looped  
 \** ***@param dt*** *deltatime, the time between one frame rendered and the next frame rendered  
 \*/* **public abstract void** update(**float** dt);  
  
 */\*\*  
 \* render the textures  
 \** ***@param sb*** *spritebatch: a container for everything that needs to be rendered, sprites etc.  
 \*/* **public abstract void** render (SpriteBatch sb);  
  
 */\*\*  
 \* remove unused data  
 \*/* **public abstract void** dispose();  
}

**GameStateManager.java**

**public class** GameStateManager {  
 **private** Stack<State> **states**;  
  
 **public** GameStateManager() {  
 **states** = **new** Stack<State>();  
 }  
  
 **public void** push(State state) {  
 **states**.push(state);  
 }  
  
 */\*\*  
 \* set a new state  
 \*   
 \** ***@param state*** *\*/* **public void** set (State state) {  
 **states**.pop().dispose();  
 **states**.push(state);  
 }  
  
 **public void** update(**float** dt) {  
 **states**.peek().update(dt);  
 }  
  
 **public void** render(SpriteBatch sb) {  
 **states**.peek().render(sb);  
 }  
}

**Playstate.java**

**public class** PlayState **extends** State {  
 **private final double heightScreen**;  
 **private final** Coin **coin**;  
 **private final** BadTube **badTube**;  
 **private** Bird **bird**;  
 **private** Texture **background**;  
  
 **private** Texture **ground**;

**private** Texture **one**;  
 **private** Texture **two**;  
 **private** Texture **three**;  
 **private** Texture **four**;  
 **private** Texture **five**;  
 **private** Texture **six**;  
 **private** Texture **seven**;  
 **private** Texture **eight**;  
 **private** Texture **nine**;  
 **private** Texture **zero**;

**private double score**;  
  
 **private static final int *TUBE\_SPACING*** = 150;  
 **private static final int *TUBE\_COUNT*** = 5;  
  
 **private** Array<Tube> **tubes**;  
 **private double addMoving**;  
 **private boolean coinTaken**;  
 **private boolean tubeClicked**;  
  
 **public** PlayState(GameStateManager gsm) {  
 **super**(gsm);  
 **bird** = **new** Bird(50,0);  
  
 **cam**.setToOrtho(**false**, Game.***WIDTH*** /2, Game.***HEIGHT*** / 2);  
 **background** = **new** Texture(**"bgs.png"**);  
 **ground** = **new** Texture(**"ground.png"**);  
  
 **one** = **new** Texture(**"1s.png"**);  
 **two** = **new** Texture(**"2s.png"**);  
 **three** = **new** Texture(**"3s.png"**);  
 **four** = **new** Texture(**"4s.png"**);  
 **five** = **new** Texture(**"5s.png"**);  
 **six** = **new** Texture(**"6s.png"**);  
 **seven** = **new** Texture(**"7s.png"**);  
 **eight** = **new** Texture(**"8s.png"**);  
 **nine** = **new** Texture(**"9s.png"**);  
 **zero** = **new** Texture(**"0s.png"**);  
  
 **heightScreen** = **cam**.**viewportHeight**;  
  
 **tubes** = **new** Array<Tube>();  
  
 **coin** = **new** Coin((**float**)(***TUBE\_SPACING***\*0.5));  
 **coinTaken** = **false**;  
  
 **badTube** = **new** BadTube((**float**)(***TUBE\_SPACING***\*0.5));  
 **tubeClicked** = **false**;  
  
 **for** (**int** i = 1; i <= ***TUBE\_COUNT***; i++) {  
 **tubes**.add(**new** Tube(i \* (***TUBE\_SPACING*** + Tube.***TUBEWIDTH***)+100));  
 }  
 }  
  
 @Override  
 **protected void** handleInput() {  
 Vector2 tmp = **new** Vector2(Gdx.*input*.getX()/2, Gdx.*input*.getY()/2);  
 **if**(Gdx.*input*.justTouched()) {  
 */\*\*  
 \* width of screen is 400, height 240  
 \* max value of gdx input is 540 and 956  
 \* y-value: (inputvalue \* (240/540))  
 \* x-value: (inputvalue \* (400/956))  
 \* since x-value keeps increasing, substract position of the tube with position of cam  
 \* remove 15 points if clicked on tube  
 \*/* **if**( **badTube**.getBounds().**y** < (240-(tmp.**y**\*0.44444)) && **badTube**.getBounds().**y**+**badTube**.getBounds().**height** > (240-(tmp.**y**\*0.44444))) {  
 **if**( **badTube**.getBounds().**x** - (**cam**.**position**.**x**-200) < (0.4184\*tmp.**x**) && **badTube**.getBounds().**x**+ **badTube**.getBounds().**width**- (**cam**.**position**.**x**-200) > (0.4184\*tmp.**x**)) {  
 **badTube**.repositionX(**badTube**.getPosBadTube().**x** + (BadTube.***WIDTH***));  
 **score** -= 15;  
 } **else** {  
 **bird**.jump();  
 }  
 } **else** {  
 **bird**.jump();  
 }  
  
 }  
 }  
  
 @Override  
 **public void** update(**float** dt) {  
 handleInput();  
 **bird**.update(dt);  
 **cam**.**position**.**x** = **bird**.getPosition().**x** + 80;  
 **addMoving**+=0.1;  
  
 **if** (**cam**.**position**.**x** - (**cam**.**viewportWidth** / 2) > **coin**.getPosCoin().**x** + **coin**.getCoin().getWidth() || **coinTaken**) {  
 **coin**.repositionX(**coin**.getPosCoin().**x** + (Coin.***WIDTH*** + (**int**)(***TUBE\_SPACING***\*0.5)));  
 **coinTaken** = **false**;  
 }  
  
 **if** (**cam**.**position**.**x** - (**cam**.**viewportWidth** / 2) > **badTube**.getPosBadTube().**x** + **badTube**.getBadTube().getWidth() || **tubeClicked**) {  
 **badTube**.repositionX(**badTube**.getPosBadTube().**x** + (BadTube.***WIDTH***));  
 **tubeClicked** = **false**;  
 }  
  
 **for**(**int** i = 0; i<**tubes**.**size**;i++) {  
 Tube tube = **tubes**.get(i);  
 **if** (**cam**.**position**.**x** - (**cam**.**viewportWidth** / 2) > tube.getPosTopTube().**x** + tube.getTopTube().getWidth()) {  
 tube.repositionX(tube.getPosTopTube().**x** + (Tube.***TUBEWIDTH*** + ***TUBE\_SPACING***)\****TUBE\_COUNT*** );  
 }  
 *// let an additional tube move every +-62 points* **if**(**addMoving** >= 62 ) {  
 **if**(!tube.isMoving()) {  
 **addMoving** -= 71;  
 tube.makeTubesPurple(tube);  
 tube.setMoving(**true**);  
 }  
 }  
 **if** (tube.isMoving()) {  
 tube.repositionY(tube.getPosTopTube().**x**, tube.getPosTopTube().**y**);  
 }  
  
 *// if collision, game is over* **if**(tube.collides(**bird**.getBounds())) {  
 **gsm**.set(**new** EndState(**gsm**, (**int**)**score**));  
 }  
  
 **if**(tube.collides(**coin**.getBounds())) {  
 **coin**.moveCoin();  
 }  
 }  
  
 *// if collision, coinposition resets* **if**(**coin**.collides(**bird**.getBounds())) {  
 **coinTaken** = **true**;  
 }  
  
 *// if collision, game is over* **if**(**badTube**.collides(**bird**.getBounds())) {  
 **gsm**.set(**new** EndState(**gsm**, (**int**)**score**));  
 }  
  
 *// bird floats on the ground* **if**(**bird**.getPosition().**y** <= **ground**.getHeight()) {  
 **bird**.getPosition().**y** = **ground**.getHeight();  
 }  
  
 *// bird reaches ceiling* **if**(**bird**.getPosition().**y** >= **cam**.**viewportHeight**-**bird**.getHeight()) {  
 **bird**.getPosition().**y** = **cam**.**viewportHeight**-**bird**.getHeight();  
 **bird**.ceiling();  
 }  
  
 *// add 30 points if coin is taken* **if** (**coinTaken** ) {  
 **score** += 30;  
 }  
  
 *// remove 15 points if a tube is removed* **if** (**tubeClicked** ) {  
 **score** -= 15;  
 }  
  
  
 **score**+=0.1;  
 **cam**.update(); *// tell gdx that cam is repositioned* }  
  
 @Override  
 **public void** render(SpriteBatch sb) {  
 sb.setProjectionMatrix(**cam**.**combined**);  
 sb.begin();  
 *// to keep the background on the right place, at the position of the camera* sb.draw(**background**, **cam**.**position**.**x** - (**cam**.**viewportWidth**/2), 0);  
  
 sb.draw(**bird**.getTexture(), **bird**.getPosition().**x**, **bird**.getPosition().**y**);  
 **for** (Tube tube : **tubes**) {  
 sb.draw(tube.getTopTube(), tube.getPosTopTube().**x**, tube.getPosTopTube().**y**);  
 sb.draw(tube.getBottomTube(), tube.getPosBotTube().**x**, tube.getPosBotTube().**y**);  
 }  
 sb.draw(**coin**.getCoin(), **coin**.getPosCoin().**x**, **coin**.getPosCoin().**y**);  
 sb.draw(**badTube**.getBadTube(), **badTube**.getPosBadTube().**x**, **badTube**.getPosBadTube().**y**);  
  
 *//draw the current score, a bit obsolete but used for good graphics* String strScore = **""** + (**int**)**score**;  
 **int** widthStrScore = strScore.length() \* 27;  
 **for** (**int** i = 0; i < strScore.length(); i++ ) {  
 **if** (strScore.charAt(i) == 48) {  
 sb.draw(**zero**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
 **else if** (strScore.charAt(i) == 49) {  
 sb.draw(**one**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
 **else if** (strScore.charAt(i) == 50) {  
 sb.draw(**two**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
 **else if** (strScore.charAt(i) == 51) {  
 sb.draw(**three**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
 **else if** (strScore.charAt(i) == 52) {  
 sb.draw(**four**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
 **else if** (strScore.charAt(i) == 53) {  
 sb.draw(**five**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
 **else if** (strScore.charAt(i) == 54) {  
 sb.draw(**six**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
 **else if** (strScore.charAt(i) == 55) {  
 sb.draw(**seven**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
 **else if** (strScore.charAt(i) == 56) {  
 sb.draw(**eight**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
 **else if** (strScore.charAt(i) == 57) {  
 sb.draw(**nine**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight**-**zero**.getHeight()-**zero**.getHeight()/2);  
 }  
  
 }  
  
 sb.draw(**ground**, **cam**.**position**.**x** - (**cam**.**viewportWidth**/2), 0);  
 sb.end();  
 }  
  
 @Override  
 **public void** dispose() {  
 **background**.dispose();  
 **bird**.dispose();  
 **ground**.dispose();  
 **for**(Tube tube : **tubes** ) {  
 tube.dispose();  
 }  
  
 **one**.dispose();  
 **two**.dispose();  
 **three**.dispose();  
 **four**.dispose();  
 **five**.dispose();  
 **six**.dispose();  
 **seven**.dispose();  
 **eight**.dispose();  
 **nine**.dispose();  
 **zero**.dispose();  
  
 System.***out***.println(**"playstate disposed"**);  
 System.***out***.println((**int**)**score**);  
 }  
}

**MenuState.java**

**public class** MenuState **extends** State {  
 **private** Texture **background**;  
 **private** Texture **playBtn**;  
 **private** Rectangle **bounds**;  
 **private** Rectangle **bounds2**;  
 **public** MenuState(GameStateManager gsm) {  
 **super**(gsm);  
 **cam**.setToOrtho(**false**, Game.***WIDTH*** /2, Game.***HEIGHT*** / 2);  
 **playBtn** = **new** Texture(**"playbtn.png"**);  
 **background** = **new** Texture(**"bgs.png"**);  
 **bounds** = **new** Rectangle(Game.***WIDTH*** /2, Game.***HEIGHT*** / 2-**playBtn**.getHeight()\*2, **playBtn**.getWidth()\*2, **playBtn**.getHeight()\*2);  
 **bounds2** = **new** Rectangle(Game.***WIDTH*** /2, Game.***HEIGHT*** / 2+**playBtn**.getHeight()-10, **playBtn**.getWidth()\*2, **playBtn**.getHeight()\*2);  
 }  
  
 @Override  
 **public void** handleInput() {  
 Vector2 tmp = **new** Vector2(Gdx.*input*.getX()/2, Gdx.*input*.getY()/2);  
  
 **if**(Gdx.*input*.justTouched()) {  
 **if** (**bounds**.contains(tmp.**x**, tmp.**y**)) {  
 **gsm**.set(**new** PlayState(**gsm**));  
 System.***out***.println(**"aaaaaaa"**);  
 }  
 **if** (**bounds2**.contains(tmp.**x**, tmp.**y**)) {  
 System.***out***.println(**"bbbbbbb"**);  
 **gsm**.set(**new** HighscoreState(**gsm**));  
 }  
 }  
  
 }  
  
 @Override  
 **public void** update(**float** dt) {  
 handleInput();  
 }  
  
 @Override  
 **public void** render(SpriteBatch sb) {  
 sb.setProjectionMatrix(**cam**.**combined**);  
 *//first open the spritebatch and add everything in it* sb.begin();  
 sb.draw(**background**,0,0);  
 sb.draw(**playBtn**, **cam**.**position**.**x**-**playBtn**.getWidth() / 2,**cam**.**position**.**y**);  
 sb.draw(**playBtn**, **cam**.**position**.**x**-**playBtn**.getWidth() / 2,**cam**.**position**.**y** - **playBtn**.getHeight()-10);  
  
  
 sb.end();  
 }  
  
 @Override  
 **public void** dispose() {  
 **background**.dispose();  
 **playBtn**.dispose();  
 System.***out***.println(**"Menustate disposed"**);  
 }  
}

**LoginState.java**

**public class** LoginState **extends** State {  
  
 **private** Preferences **prefs**;  
 GameStateManager **gsm**;  
 **private** Texture **background**;  
 **private** Texture **logo**;  
 **public** LoginState(GameStateManager gsm) {  
 **super**(gsm);  
 **this**.**gsm** = gsm;  
 **logo** = **new** Texture(**"logo2.png"**);  
  
 **cam**.setToOrtho(**false**, Game.***WIDTH*** /2, Game.***HEIGHT*** / 2);  
 **background** = **new** Texture(**"bgs.png"**);  
  
 Game.*myGameCallback*.onStartActivityLogin();  
 }  
   
 @Override  
 **public void** handleInput() {  
 Vector3 tmp = **new** Vector3(Gdx.*input*.getX()/2, Gdx.*input*.getY()/2, 0);  
 **if**(Gdx.*input*.justTouched()) {  
 **gsm**.set(**new** MenuState(**gsm**));  
 **cam**.unproject(tmp);  
 }  
 }  
  
 @Override  
 **public void** update(**float** dt) {  
 handleInput();  
 }  
  
 @Override  
 **public void** render(SpriteBatch sb) {  
 sb.setProjectionMatrix(**cam**.**combined**);  
 *//first open the spritebatch and add everything in it* sb.begin();  
 sb.draw(**background**,0,0);  
 sb.draw(**logo**, **cam**.**position**.**x**-**logo**.getWidth() / 2,**cam**.**position**.**y**-**logo**.getHeight() / 2);  
 sb.end();  
 }  
  
 @Override  
 **public void** dispose() {  
 **background**.dispose();  
 **logo**.dispose();  
 System.***out***.println(**"LoginState disposed"**);  
 }  
}

**HighscoreState.java**

**public class** HighscoreState **extends** State{  
 **private final int highscore**;  
 **private final** String **currPlayer**;  
 **private final** String **highscoreOwner**;  
 **private** Texture **playBtn**;  
 **private** Preferences **prefs**;  
 **private** GlyphLayout **glyphLayout**;  
 **private** Texture **background**;  
 **private** Rectangle **bounds**;  
  
 **private** Texture **one**;  
 **private** Texture **two**;  
 **private** Texture **three**;  
 **private** Texture **four**;  
 **private** Texture **five**;  
 **private** Texture **six**;  
 **private** Texture **seven**;  
 **private** Texture **eight**;  
 **private** Texture **nine**;  
 **private** Texture **zero**;  
 **private** Texture **cont**;  
 **static int** *OFFSET\_Y* = 0;  
 **private int drawn**;  
  
 BitmapFont **font**;  
  
 **public** HighscoreState(GameStateManager gsm) {  
 **super**(gsm);  
 **cam**.setToOrtho(**false**, Game.***WIDTH*** /2, Game.***HEIGHT*** / 2);  
 **one** = **new** Texture(**"1s.png"**);  
 **two** = **new** Texture(**"2s.png"**);  
 **three** = **new** Texture(**"3s.png"**);  
 **four** = **new** Texture(**"4s.png"**);  
 **five** = **new** Texture(**"5s.png"**);  
 **six** = **new** Texture(**"6s.png"**);  
 **seven** = **new** Texture(**"7s.png"**);  
 **eight** = **new** Texture(**"8s.png"**);  
 **nine** = **new** Texture(**"9s.png"**);  
 **zero** = **new** Texture(**"0s.png"**);  
 **background** = **new** Texture(**"backgroundSimple.png"**);  
 **playBtn** = **new** Texture(**"playbtn.png"**);  
  
 **glyphLayout** = **new** GlyphLayout();  
 **font** = **new** BitmapFont();  
  
 **prefs** = Game.*getPrefs*();  
 **highscoreOwner** = **prefs**.getString(**"highscoreOwner"**);  
 **highscore** = **prefs**.getInteger(**"highscore"**);  
 **currPlayer** = **prefs**.getString(**"currPlayer"**);  
 }  
  
 @Override  
 **public void** handleInput() {  
 **if** (Gdx.*input*.isTouched()) {  
 **prefs** = Game.*getPrefs*();  
 **if**(!**prefs**.getString(**"currPlayer"**, **currPlayer**).equals(**"Guest"**)) {  
 **prefs**.putString(**"action"**, **"showLeaderboard"**);  
 **prefs**.flush();  
 Game.*myGameCallback*.onStartActivityLogin();  
 }  
 **gsm**.set(**new** MenuState(**gsm**));  
 }  
 }  
  
 @Override  
 **public void** update(**float** dt) {  
 handleInput();  
 }  
  
 @Override  
 **public void** render(SpriteBatch sb) {  
 sb.setProjectionMatrix(**cam**.**combined**);  
 sb.begin();  
  
 sb.draw(**background**,(Game.***WIDTH*** /2 - **background**.getWidth())/2,(Game.***HEIGHT*** /2 - **background**.getHeight())/2);  
  
 *// glyphLayout used to center the sentence on the screen since you cannot know the width of the string  
 // sentence: the local highscore is* String localHighscore = **"the local highscore is"**;  
 **font**.getData().setScale(1);  
 **font**.setColor(Color.***BLACK***);  
 **glyphLayout**.setText(**font**,localHighscore);  
 **float** w = **glyphLayout**.**width**;  
 **float** textHeight = **glyphLayout**.**height**;  
 **font**.draw(sb, localHighscore, **cam**.**position**.**x**-w / 2,**cam**.**viewportHeight** - textHeight\*2);  
  
 *// highscore* String strScore = **""** + **highscore**;  
 *// 27 is width of each character in pixels* **int** widthStrScore = strScore.length() \* 27;  
  
 *// by* String by = **"by"**;  
 **glyphLayout**.setText(**font**,by);  
 w = **glyphLayout**.**width**;  
 **font**.draw(sb, by, **cam**.**position**.**x**-w / 2,**cam**.**viewportHeight** - textHeight\*8);  
  
 *// highscoreOwner* **font**.getData().setScale(1.5f);  
 **glyphLayout**.setText(**font**,**highscoreOwner**);  
 w = **glyphLayout**.**width**;  
 **font**.draw(sb, **highscoreOwner**, **cam**.**position**.**x**-w / 2,**cam**.**viewportHeight** - textHeight\*10);  
  
 *// width of screen : cam.position.x - widthStrScore / 2 7  
 // move a little left or right according to the amount of characters : widthStrScore / strScore.length() \* i  
 // draw the highscore* **for** (**int** i = 0; i < strScore.length(); i++ ) {  
 **if** (strScore.charAt(i) == 48) {  
 sb.draw(**zero**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 **else if** (strScore.charAt(i) == 49) {  
 sb.draw(**one**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 **else if** (strScore.charAt(i) == 50) {  
 sb.draw(**two**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 **else if** (strScore.charAt(i) == 51) {  
 sb.draw(**three**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 **else if** (strScore.charAt(i) == 52) {  
 sb.draw(**four**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 **else if** (strScore.charAt(i) == 53) {  
 sb.draw(**five**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 **else if** (strScore.charAt(i) == 54) {  
 sb.draw(**six**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 **else if** (strScore.charAt(i) == 55) {  
 sb.draw(**seven**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 **else if** (strScore.charAt(i) == 56) {  
 sb.draw(**eight**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 **else if** (strScore.charAt(i) == 57) {  
 sb.draw(**nine**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, **cam**.**viewportHeight** - textHeight\*7);  
 }  
 }  
 sb.draw(**playBtn**, **cam**.**position**.**x**-**playBtn**.getWidth() / 2,**cam**.**viewportHeight** - textHeight\*18);  
 sb.end();  
 }  
   
 @Override  
 **public void** dispose() {  
 **background**.dispose();  
 **one**.dispose();  
 **two**.dispose();  
 **three**.dispose();  
 **four**.dispose();  
 **five**.dispose();  
 **six**.dispose();  
 **seven**.dispose();  
 **eight**.dispose();  
 **nine**.dispose();  
 **zero**.dispose();  
 **playBtn**.dispose();  
 System.***out***.println(**"HighscoreState disposed"**);  
 }  
}

**public class** EndState **extends** State {  
 **private** String **highscoreOwner**;  
 **private** String **currPlayer**;  
 **private int highscore**;  
 **private** Texture **background**;  
 **private** Texture **GO**;  
 **private** Rectangle **bounds**;  
 **int score**;  
  
 **private** Texture **one**;  
 **private** Texture **two**;  
 **private** Texture **three**;  
 **private** Texture **four**;  
 **private** Texture **five**;  
 **private** Texture **six**;  
 **private** Texture **seven**;  
 **private** Texture **eight**;  
 **private** Texture **nine**;  
 **private** Texture **zero**;  
 **private** Texture **continueTexture**;  
 **static int** *OFFSET\_Y* = 0;  
 **private int drawn**;  
 **private** Preferences **prefs**;  
  
 **public** EndState(GameStateManager gsm, **int** score) {  
 **super**(gsm);  
 **this**.**score** = score;  
 **one** = **new** Texture(**"1s.png"**);  
 **two** = **new** Texture(**"2s.png"**);  
 **three** = **new** Texture(**"3s.png"**);  
 **four** = **new** Texture(**"4s.png"**);  
 **five** = **new** Texture(**"5s.png"**);  
 **six** = **new** Texture(**"6s.png"**);  
 **seven** = **new** Texture(**"7s.png"**);  
 **eight** = **new** Texture(**"8s.png"**);  
 **nine** = **new** Texture(**"9s.png"**);  
 **zero** = **new** Texture(**"0s.png"**);  
 **continueTexture** = **new** Texture(**"continue.png"**);  
 **cam**.setToOrtho(**false**, Game.***WIDTH*** /2, Game.***HEIGHT*** / 2);  
 **GO** = **new** Texture(**"GO2.png"**);  
 **background** = **new** Texture(**"backgroundSimple.png"**);  
 **drawn** = 0;  
 **currPlayer** = **""**;  
  
 **prefs** = Game.*getPrefs*();  
 **highscore** = **prefs**.getInteger(**"highscore"**);  
 **if**(score>**highscore**) {  
 **highscoreOwner** = **prefs**.getString(**"currPlayer"**, **"Guest"**);  
 **prefs**.putString(**"highscoreOwner"**, **highscoreOwner**);  
 **prefs**.putInteger(**"highscore"**, score);  
  
 }  
 **if** (!**currPlayer**.equals(**"Guest"**)) {  
 **prefs**.putString(**"action"**, **"submit"**);  
 }  
 **highscoreOwner** = **prefs**.getString(**"currPlayer"**);  
 **prefs**.putInteger(**"currScore"**, score);  
 **prefs**.flush();  
 }  
  
 @Override  
 **public void** handleInput() {  
 *// to avoid clicking trough endscreen right away* **if**(**drawn** == 0) {  
 **try** {  
 Thread.*sleep*(2000);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **drawn** = 1;  
 } **else if** (**drawn** == 1){  
 **if** (Gdx.*input*.isTouched()) {  
 **if**(!**prefs**.getString(**"currPlayer"**, **currPlayer**).equals(**"Guest"**)) {  
 Game.*myGameCallback*.onStartActivityLogin();  
 }  
 **gsm**.set(**new** MenuState(**gsm**));  
 }  
 }  
 }  
  
 @Override  
 **public void** update(**float** dt) {  
 handleInput();  
 }  
  
 @Override  
 **public void** render(SpriteBatch sb) {  
 sb.setProjectionMatrix(**cam**.**combined**);  
 *//first open the spritebatch and add everything in it* sb.begin();  
  
 sb.draw(**background**,(Game.***WIDTH*** /2 - **background**.getWidth())/2,(Game.***HEIGHT*** /2 - **background**.getHeight())/2);  
 sb.draw(**GO**, **cam**.**position**.**x**-**GO**.getWidth() / 2,**cam**.**position**.**y**-**GO**.getHeight()/4);  
 String strScore = **""** + (**int**)**score**;  
 **int** widthStrScore = strScore.length() \* 27;  
  
 **for** (**int** i = 0; i < strScore.length(); i++ ) {  
 **if** (strScore.charAt(i) == 48) {  
 sb.draw(**zero**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 **else if** (strScore.charAt(i) == 49) {  
 sb.draw(**one**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 **else if** (strScore.charAt(i) == 50) {  
 sb.draw(**two**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 **else if** (strScore.charAt(i) == 51) {  
 sb.draw(**three**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 **else if** (strScore.charAt(i) == 52) {  
 sb.draw(**four**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 **else if** (strScore.charAt(i) == 53) {  
 sb.draw(**five**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 **else if** (strScore.charAt(i) == 54) {  
 sb.draw(**six**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 **else if** (strScore.charAt(i) == 55) {  
 sb.draw(**seven**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 **else if** (strScore.charAt(i) == 56) {  
 sb.draw(**eight**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 **else if** (strScore.charAt(i) == 57) {  
 sb.draw(**nine**, **cam**.**position**.**x** - widthStrScore / 2 + widthStrScore / strScore.length() \* i, *OFFSET\_Y* + **zero**.getHeight()\*3/2);  
 }  
 }  
 *// show continue after a while* **if**(**drawn** == 1) {  
 sb.draw(**continueTexture**, **cam**.**position**.**x** - **continueTexture**.getWidth() / 2, 0);  
 }  
 sb.end();  
 }  
  
 @Override  
 **public void** dispose() {  
 **background**.dispose();  
 **one**.dispose();  
 **two**.dispose();  
 **three**.dispose();  
 **four**.dispose();  
 **five**.dispose();  
 **six**.dispose();  
 **seven**.dispose();  
 **eight**.dispose();  
 **nine**.dispose();  
 **zero**.dispose();  
 **GO**.dispose();  
  
 System.***out***.println(**"Endstate disposed"**);  
 }  
}

**Tube.java**

**public class** Tube {  
 **private boolean moving**;  
 **private** Texture **topTube**, **bottomTube**;  
 **private** Vector2 **posTopTube**, **posBotTube**;  
 **private** Random **rand**;  
 **private static final int *VALUE\_BETWEEN\_MINANDMAX*** = 150;  
 **private static final int *GAP*** = 100;  
 **private static final int *LOWEST*** = 30;  
 **public static final int *TUBEWIDTH*** = 52;  
  
 **private** Rectangle **boundsTop**, **boundsBot**;  
 **private boolean up**;  
  
  
 */\*\*  
 \* constructor of the tube  
 \*  
 \** ***@param x*** *x-axis position of the tube  
 \*/* **public** Tube( **float** x ) {  
 **topTube** = **new** Texture(**"toptubeGreen.png"**);  
 **bottomTube** = **new** Texture(**"bottomtubeGreen.png"**);  
 **rand** = **new** Random();  
 **moving** = **false**;  
 **up** = **true**;  
  
 **posTopTube** = **new** Vector2(x, **rand**.nextInt(***VALUE\_BETWEEN\_MINANDMAX***)+ ***GAP*** + ***LOWEST***);  
 **posBotTube** = **new** Vector2(x, **posTopTube**.**y** - ***GAP*** - **bottomTube**.getHeight());  
  
 *// invisible rectangles for collision detection* **boundsTop** = **new** Rectangle(**posTopTube**.**x**,**posTopTube**.**y**, **topTube**.getWidth(),**topTube**.getHeight());  
 **boundsBot** = **new** Rectangle(**posBotTube**.**x**,**posBotTube**.**y**, **bottomTube**.getWidth(),**bottomTube**.getHeight());  
 }  
  
 */\*\*  
 \* return the texture topTube  
 \*  
 \** ***@return*** *texture topTube  
 \*/* **public** Texture getTopTube() {  
 **return topTube**;  
 }  
 */\*\*  
 \* return the texture bottomtube  
 \*  
 \** ***@return*** *texture bottomtube  
 \*/* **public** Texture getBottomTube() {  
 **return bottomTube**;  
 }  
  
 **public void** setTopTube(Texture topTube) { **this**.**topTube** = topTube; }  
  
 **public void** setBottomTube(Texture bottomTube) { **this**.**bottomTube** = bottomTube; }  
  
 */\*\*  
 \* return the texture TopTube  
 \*  
 \** ***@return*** *x and y of TopTube  
 \*/* **public** Vector2 getPosBotTube() {  
 **return posBotTube**;  
 }  
 */\*\*  
 \* return the position of toptube  
 \*  
 \** ***@return*** *x and y of TopTube  
 \*/* **public** Vector2 getPosTopTube() {  
 **return posTopTube**;  
 }  
 */\*\*  
 \* get if tubes should move  
 \*  
 \** ***@return*** *true if tubes are moving  
 \*/* **public boolean** isMoving() { **return moving**; }  
 */\*\*  
 \* set the moving variable of the tubes to see if they should move  
 \*  
 \** ***@param moving*** *true if the tubes should move  
 \*/* **public void** setMoving(**boolean** moving) { **this**.**moving** = moving; }  
  
  
  
  
  
  
  
 */\*\*  
 \* method to reposition the tube on the screen on the x-axis  
 \*  
 \** ***@param x*** *x-position of the tube on the screen  
 \*/* **public void** repositionX(**float** x) {  
 **posTopTube**.set(x, **rand**.nextInt(***VALUE\_BETWEEN\_MINANDMAX***) + ***GAP*** + ***LOWEST***);  
 **posBotTube**.set(x, **posTopTube**.**y** - ***GAP*** - **bottomTube**.getHeight());  
  
 **boundsTop**.setPosition(**posTopTube**.**x**, **posTopTube**.**y**);  
 **boundsBot**.setPosition(**posBotTube**.**x**, **posBotTube**.**y**);  
 }  
  
 */\*\*  
 \* method to repostion the tubes up and down  
 \*  
 \** ***@param x*** *fixed x-axis value, wont change in order to keep the tube at the right location  
 \** ***@param y*** *y-axis value of tube to make it go up and down  
 \*/* **public void** repositionY(**float** x, **float** y) {  
 **float** height = y;  
 **if** (**up** == **true** ) {  
 **if**(height < Game.***HEIGHT*** / 2+20){  
 height+=1;  
 }  
 **else if**(height >= Game.***HEIGHT*** / 2){  
 **up** = **false**;  
 }  
 }  
 **if** (**up** == **false** ) {  
 **if**(height >= ***LOWEST***+20){  
 height-=1;  
 }  
 **else** {  
 **up** = **true**;  
 }  
 }  
 **posTopTube**.set(x, height);  
 **posBotTube**.set(x, **posTopTube**.**y** - ***GAP*** - **bottomTube**.getHeight());  
  
 **boundsTop**.setPosition(**posTopTube**.**x**, **posTopTube**.**y**);  
 **boundsBot**.setPosition(**posBotTube**.**x**, **posBotTube**.**y**);  
 }  
  
 */\*\*  
 \* checks if there is a collision between bird and tube  
 \*  
 \** ***@param player*** *the bird  
 \** ***@return*** *boolean if a collision has happened  
 \*/* **public boolean** collides(Rectangle player) {  
 **return** player.overlaps(**boundsTop**) || player.overlaps(**boundsBot**);  
 }  
  
 */\*\*  
 \* method to dispose the textures  
 \*/* **public void** dispose () {  
 **topTube**.dispose();  
 **bottomTube**.dispose();  
 }  
  
 */\*\*  
 \* make the tube purple, that means it moves  
 \*  
 \** ***@param tube*** *\*/* **public void** makeTubesPurple(Tube tube) {  
 tube.**topTube** = **new** Texture(**"toptubePurple.png"**);  
 tube.**bottomTube** = **new** Texture(**"bottomtubePurple.png"**);  
 }

}

**Coin.java**

**public class** Coin {  
 **private** Texture **coin**;  
 **private** Vector2 **posCoin**;  
 **private** Random **rand**;  
 **private static final int *VALUE\_BETWEEN\_MINANDMAX*** = 200;  
 **public static final int *WIDTH*** = 30;  
  
 **private** Rectangle **boundsCoin**;  
  
  
 */\*\*  
 \* constructor of the coin  
 \*  
 \** ***@param x*** *x-axis position of the tube  
 \*/* **public** Coin( **float** x ) {  
 **coin** = **new** Texture(**"coin.png"**);  
 **rand** = **new** Random();  
 **int** distBetween = **rand**.nextInt(8);  
 distBetween +=4;  
 **posCoin** = **new** Vector2(x\*distBetween, **rand**.nextInt(***VALUE\_BETWEEN\_MINANDMAX***));  
  
  
 *// invisible rectangles for collision detection* **boundsCoin** = **new** Rectangle(**posCoin**.**x**,**posCoin**.**y**, **coin**.getWidth(),**coin**.getHeight());  
 }  
  
 */\*\*  
 \* return the texture coin  
 \*  
 \** ***@return*** *texture topTube  
 \*/* **public** Texture getCoin() {  
 **return coin**;  
 }  
  
 */\*\*  
 \* return the position of coin  
 \*  
 \** ***@return*** *x and y of TopTube  
 \*/* **public** Vector2 getPosCoin() {  
 **return posCoin**;  
 }  
  
 */\*\*  
 \* method to reposition the coin on the screen on the x-axis  
 \*  
 \** ***@param x*** *x-position of the tube on the screen  
 \*/* **public void** repositionX(**float** x) {  
 **int** distBetween = **rand**.nextInt(8);  
 distBetween +=4;  
 **posCoin** = **new** Vector2(x\*distBetween, **rand**.nextInt(***VALUE\_BETWEEN\_MINANDMAX***));  
  
 **boundsCoin**.setPosition(**posCoin**.**x**, **posCoin**.**y**);  
 }  
  
 */\*\*  
 \* checks if there is a collision between bird and coin  
 \*  
 \** ***@param player*** *the bird  
 \** ***@return*** *boolean if a collision has happened  
 \*/* **public boolean** collides(Rectangle player) {  
 **return** player.overlaps(**boundsCoin**);  
 }  
  
 */\*\*  
 \* method to dispose the textures  
 \*/* **public void** dispose () {  
 **coin**.dispose();  
 }  
  
 */\*\*  
 \* method to move the coin 25 pixels if needed  
 \*/* **public void** moveCoin() {  
 **posCoin**.**x** += 25;  
 }  
  
 */\*\*  
 \* get the bounds of the coin to detect collision  
 \*   
 \** ***@return*** *bounds coin  
 \*/* **public** Rectangle getBounds() {  
 **return boundsCoin**;  
 }  
}

**Bird.java**

**public class** Bird {  
 **private** Vector3 **position**;  
 **private** Vector3 **velocity**;  
 **private** Rectangle **bounds**;  
 **private** Texture **texture**;  
  
 **private** Sound **flap**;  
  
 **private static final int *GRAVITY*** = -15;  
 **private static final int *MOVEMENT*** = 100;  
  
 **public** Vector3 getPosition() {  
 **return position**;  
 }  
  
 **public** Texture getTexture() {  
 **return texture**;  
 }  
  
 **public** Bird(**int** x, **int** y) {  
 **position** = **new** Vector3(x,y,0);  
 **velocity** = **new** Vector3(0,0,0);  
 **texture** = **new** Texture(**"bird2.png"**);  
  
 **flap** = Gdx.*audio*.newSound(Gdx.*files*.internal(**"sfx\_wing.ogg"**));  
 **bounds** = **new** Rectangle(x,y,**texture**.getWidth(),**texture**.getHeight());  
 }  
  
 */\*\*  
 \* get the height of the bird  
 \*   
 \** ***@return*** *height of bird  
 \*/* **public float** getHeight() {  
 **return texture**.getHeight();  
 }  
  
 **public void** update(**float** dt) {  
 *//birdAnimation.update(dt);* **if** (**position**.**y** > 0) {  
 **velocity**.add(0, ***GRAVITY***, 0);  
 }  
 **velocity**.scl(dt);  
 **position**.add(***MOVEMENT*** \* dt,**velocity**.**y**, 0);  
 **if** (**position**.**y** < 0 ) {  
 **position**.**y** = 0;  
 }  
 **velocity**.scl(1/dt);  
 **bounds**.setPosition(**position**.**x**,**position**.**y**);  
 }  
  
 */\*\*  
 \* set the speed to 250 if the bird has to jump  
 \*/* **public void** jump() {  
 **velocity**.**y**=250;  
 **flap**.play(0.5f);  
 }  
  
 */\*\*  
 \* if the bird hits the ceiling, set speed to 0  
 \*/* **public void** ceiling() {  
 **velocity**.**y**=0;  
 }  
  
 */\*\*  
 \* get bounds of bird  
 \*   
 \** ***@return*** *bounds of bird  
 \*/* **public** Rectangle getBounds() {  
 **return bounds**;  
 }  
  
 **public void** dispose () {  
 **texture**.dispose();  
 **flap**.dispose();  
 }  
}

**BadTube.java**

**public class** BadTube {  
 **private** Texture **badTube**;  
 **private** Vector2 **posBadTube**;  
 **private** Random **rand**;  
 **private static final int *VALUE\_BETWEEN\_MINANDMAX*** = 200;  
 **public static final int *WIDTH*** = 30;  
  
 **private** Rectangle **boundsBadTube**;  
 **private boolean up**;  
  
  
 */\*\*  
 \* constructor of the coin  
 \*  
 \** ***@param x*** *x-axis position of the tube  
 \*/* **public** BadTube(**float** x ) {  
 **badTube** = **new** Texture(**"tubeBlue.png"**);  
 **rand** = **new** Random();  
 **int** distBetween = **rand**.nextInt(8);  
 distBetween +=4;  
 **posBadTube** = **new** Vector2(x\*distBetween, **rand**.nextInt(***VALUE\_BETWEEN\_MINANDMAX***));  
  
  
 *// invisible rectangles for collision detection* **boundsBadTube** = **new** Rectangle(**posBadTube**.**x**,**posBadTube**.**y**, **badTube**.getWidth(),**badTube**.getHeight());  
 }  
  
 */\*\*  
 \* return the texture coin  
 \*  
 \** ***@return*** *texture topTube  
 \*/* **public** Texture getBadTube() {  
 **return badTube**;  
 }  
  
 **public void** setbadTube(Texture coin) { **this**.**badTube** = **badTube**; }  
  
  
 */\*\*  
 \* return the position of coin  
 \*  
 \** ***@return*** *x and y of TopTube  
 \*/* **public** Vector2 getPosBadTube() {  
 **return posBadTube**;  
 }  
  
 */\*\*  
 \* method to reposition the coin on the screen on the x-axis  
 \*  
 \** ***@param x*** *x-position of the tube on the screen  
 \*/* **public void** repositionX(**float** x) {  
 **int** distBetween = **rand**.nextInt(8);  
 distBetween +=4;  
 **posBadTube** = **new** Vector2(x\*distBetween, **rand**.nextInt(***VALUE\_BETWEEN\_MINANDMAX***));  
  
 **boundsBadTube**.setPosition(**posBadTube**.**x**, **posBadTube**.**y**);  
 }  
  
  
 */\*\*  
 \* checks if there is a collision between bird and coin  
 \*  
 \** ***@param player*** *the bird  
 \** ***@return*** *boolean if a collision has happened  
 \*/* **public boolean** collides(Rectangle player) {  
 **return** player.overlaps(**boundsBadTube**);  
 }  
  
 */\*\*  
 \* method to dispose the textures  
 \*/* **public void** dispose () {  
 **badTube**.dispose();  
 }  
  
 **public** Rectangle getBounds() {  
 **return boundsBadTube**;  
 }  
}

## Android module

**AndroidLauncher.java**

**public class** AndroidLauncher **extends** AndroidApplication **implements** Game.MyGameCallback {  
 @Override  
 **protected void** onCreate (Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 AndroidApplicationConfiguration config = **new** AndroidApplicationConfiguration();  
  
 *// create an instance of MyGame, and set the callback* Game myGame = **new** Game();  
 *// Since AndroidLauncher implements MyGame.MyGameCallback, we can just pass 'this' to the callback setter.* myGame.setMyGameCallback(**this**);  
  
 initialize(**new** Game(), config);  
 }  
  
 */\*\*  
 \* start the GoogleApiActivity  
 \*/* @Override  
 **public void** onStartActivityLogin() {  
 Intent intent = **new** Intent(**this**, GoogleApiActivity.**class**);  
 startActivity(intent);  
 }  
}

**GoogleApiActivity.java**

**public class** GoogleApiActivity **extends** FragmentActivity **implements** View.OnClickListener,  
 GoogleApiClient.ConnectionCallbacks,  
 GoogleApiClient.OnConnectionFailedListener{  
  
 **private static final int *REQUEST\_RESOLVE\_ERR*** = 9001;  
 **private** GoogleApiClient **mGoogleApiClient**;  
 **private** TextView **hello**, **who**;  
  
 **private** Button **continueToGame**, **guest\_button**, **sign\_out\_button**;  
 **private** SignInButton **sign\_in\_button**;  
  
 **private** String **currPlayer**;  
 **private static** SharedPreferences *prefs*;  
 **private static int** *start*;  
  
 **private** Button **retry**;  
  
 */\*\*  
 \* create googleapiclient to connect to leaderboard  
 \*  
 \** ***@param savedInstanceState*** *\*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_google\_api***);  
 **mGoogleApiClient** = **new** GoogleApiClient.Builder(**this**)  
 .addConnectionCallbacks(**this**)  
 .enableAutoManage(**this**, 0 */\* clientId \*/*, **this**)  
 .addApi(Games.***API***).addScope(Games.***SCOPE\_GAMES***)  
 *// add other APIs and scopes here as needed* .build();  
  
 **sign\_in\_button** = (SignInButton)findViewById(R.id.***sign\_in\_button***);  
 **guest\_button** = (Button)findViewById(R.id.***guest\_button***);  
 **continueToGame** = (Button)findViewById(R.id.***continueToGame***);  
 **sign\_out\_button** = (Button)findViewById(R.id.***sign\_out\_button***);  
 **retry** = (Button)findViewById(R.id.***retry***);  
 findViewById(R.id.***sign\_in\_button***).setOnClickListener(**this**);  
 findViewById(R.id.***guest\_button***).setOnClickListener(**this**);  
 findViewById(R.id.***continueToGame***).setOnClickListener(**this**);  
 findViewById(R.id.***sign\_out\_button***).setOnClickListener(**this**);  
 findViewById(R.id.***retry***).setOnClickListener(**this**);  
  
 **hello** = (TextView)findViewById(R.id.***hello***);  
 **who** = (TextView)findViewById(R.id.***who***);  
 }  
  
 */\*\*  
 \* automatically called after onCreate  
 \*/* @Override  
 **protected void** onStart() {  
 **super**.onStart();  
 **mGoogleApiClient**.connect();  
 }  
  
 */\*\*  
 \* check if button is clicked  
 \*  
 \** ***@param view*** *current view  
 \*/* @Override  
 **public void** onClick(View view) {  
 **switch** (view.getId()) {  
 **case** R.id.***sign\_in\_button***:  
 signIn();  
 **break**;  
 **case** R.id.***guest\_button***:  
 playAsGuest();  
 **break**;  
 **case** R.id.***continueToGame***:  
 finishAct();  
 **break**;  
 **case** R.id.***sign\_out\_button***:  
 signOut();  
 **case** R.id.***retry***:  
 retry();  
 }  
 }  
  
 **private void** signOut() {  
 *// Games.signOut(mGoogleApiClient);* **sign\_in\_button**.setVisibility(View.***VISIBLE***);  
 **continueToGame**.setVisibility(View.***GONE***);  
 **guest\_button**.setVisibility(View.***VISIBLE***);  
 **sign\_out\_button**.setVisibility(View.***GONE***);  
 }  
  
 */\*\*  
 \* finish the activity and save the shared preferences  
 \* initialize the shared prefs if first time  
 \*/* **private void** finishAct() {  
 *start* = 1;  
 *prefs* = **this**.getSharedPreferences(  
 **"com.project2.prefs"**, Context.***MODE\_PRIVATE***);  
 SharedPreferences.Editor editor = *prefs*.edit();  
 editor.putString(**"currPlayer"**, **currPlayer**);  
 editor.putString(**"action"**, **"nothing"**);  
 editor.putInt(**"currScore"**, 0);  
 **if**(*prefs*.getInt(**"highscore"**,0)==0) {  
 editor.putInt(**"highscore"**, 1);  
 }  
 **if**(*prefs*.getString(**"highscoreOwner"**,**"notCreated"**).equals(**"notCreated"**)) {  
 editor.putString(**"highscoreOwner"**, **"no owner"**);  
 }  
 editor.commit();

*/\*\*  
 \* releasing an official apk on the play store requires the achievements to be implemented, dummy implementation  
 \*/*

**if**(**mGoogleApiClient**.isConnected()) {  
 Games.***Achievements***.unlock(**mGoogleApiClient**, **"CgkI2Y6BmvQZEAIQAg"**);  
 Games.***Achievements***.unlock(**mGoogleApiClient**, **"CgkI2Y6BmvQZEAIQAw"**);  
 Games.***Achievements***.unlock(**mGoogleApiClient**, **"CgkI2Y6BmvQZEAIQBA"**);  
 Games.***Achievements***.unlock(**mGoogleApiClient**, **"CgkI2Y6BmvQZEAIQBQ"**);  
 Games.***Achievements***.unlock(**mGoogleApiClient**, **"CgkI2Y6BmvQZEAIQBg"**);  
 }  
  
 **this**.finish();  
 }  
  
 */\*\*  
 \* play as a guest, set current player to guest  
 \*/* **private void** playAsGuest() {  
 **currPlayer** = **"Guest"**;  
 **hello**.setText(**"Hello Guest!"**);  
 endLogin();  
 }  
  
 */\*\*  
 \* sign in to the google api and get display name  
 \*/* **private void** signIn() {  
 **if** (**mGoogleApiClient**.isConnected()) {  
 **currPlayer** = Games.***Players***.getCurrentPlayer(**mGoogleApiClient**).getDisplayName();  
 **hello**.setText(**"hello "** + **currPlayer** + **"!"**);  
 endLogin();  
 } **else** {  
 **hello**.setText(**"Error logging in, try again or play as guest."**);  
 }  
 }  
  
 */\*\*  
 \* end the login by changing the buttons  
 \*/* **private void** endLogin() {  
 **who**.setText(**"Welcome!"**);  
 **sign\_in\_button**.setVisibility(View.***GONE***);  
 **continueToGame**.setVisibility(View.***VISIBLE***);  
 **guest\_button**.setVisibility(View.***GONE***);  
 **sign\_out\_button**.setVisibility(View.***VISIBLE***);  
 }  
  
 */\*\*  
 \* method is called when the api is connected  
 \*  
 \** ***@param bundle*** *\*/* @Override  
 **public void** onConnected(@Nullable Bundle bundle) {  
 **if**(*prefs* != **null** ) {  
 **if** (*start* == 1 && *prefs*.getString(**"action"**, **"test"**).equals(**"submit"**)) {  
 switchToLeaderboard();  
 submit();  
 }  
  
 **if** (*start* == 1 && *prefs*.getString(**"action"**, **"test"**).equals(**"showLeaderboard"**)) {  
 switchToLeaderboard();  
 showLeaderboard ();  
 }  
 }  
 }  
  
 @Override  
 **public void** onConnectionSuspended(**int** i) {  
  
 }  
  
 **private static int** *RC\_SIGN\_IN* = 9001;  
  
 **private boolean mResolvingConnectionFailure** = **false**;  
 **private boolean mAutoStartSignInflow** = **true**;  
 **private boolean mSignInClicked** = **false**;  
  
*// ...  
  
 /\*\*  
 \* method called when the connection to the api failed  
 \*  
 \** ***@param connectionResult*** *\*/* @Override  
 **public void** onConnectionFailed(@NonNull ConnectionResult connectionResult) {  
 *// retry connecting by showing the intent again* **if** (connectionResult.hasResolution()) {  
 **try** {  
 connectionResult.startResolutionForResult(**this**, ***REQUEST\_RESOLVE\_ERR***);  
 } **catch** (IntentSender.SendIntentException e) {  
 **mGoogleApiClient**.connect();  
 e.printStackTrace();  
 }  
 }  
 *// call to remove the other buttons, even if failed* **if**(*prefs* != **null** ) {  
 **if** (*start* == 1 && *prefs*.getString(**"action"**, **"test"**).equals(**"submit"**)) {  
 switchToLeaderboard();  
 }  
  
 **if** (*start* == 1 && *prefs*.getString(**"action"**, **"test"**).equals(**"showLeaderboard"**)) {  
 switchToLeaderboard();  
 }  
 }  
 }  
  
 */\*\*  
 \* get the result of the activity that is called if the connection failed  
 \** ***@param requestCode*** *\** ***@param resultCode*** *\** ***@param data*** *\*/* @Override  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 **super**.onActivityResult(requestCode, resultCode, data);  
 **switch** (requestCode) {  
 **case *REQUEST\_RESOLVE\_ERR***:  
 **if** (resultCode == ***RESULT\_OK***) {  
 **if** (!**mGoogleApiClient**.isConnecting()  
 && !**mGoogleApiClient**.isConnected())  
 **mGoogleApiClient**.connect();  
 }  
 **break**;  
 }  
 }  
  
 */\*\*  
 \* remove login buttons and show retry button  
 \*/* **public void** switchToLeaderboard () {  
 **sign\_in\_button**.setVisibility(View.***GONE***);  
 **continueToGame**.setVisibility(View.***GONE***);  
 **guest\_button**.setVisibility(View.***GONE***);  
 **sign\_out\_button**.setVisibility(View.***GONE***);  
 **who**.setVisibility(View.***GONE***);  
 **retry**.setVisibility(View.***VISIBLE***);  
 **if**(!**mGoogleApiClient**.isConnected()) {  
 **hello**.setText(**"An error occured connecting to the leaderboard, retry connecting."**);  
 }  
 }  
  
 */\*\*  
 \* submit the highscore  
 \*/* **public void** submit() {  
 **if** (**mGoogleApiClient**.isConnected()) {  
 Games.***Leaderboards***.submitScore(**mGoogleApiClient**, getString(R.string.***leaderboard\_id***), *prefs*.getInt(**"currScore"**, *prefs*.getInt(**"currScore"**,0)));  
 **this**.finish();  
 }  
  
 }  
  
 */\*\*  
 \* show the leaderboard  
 \*/* **public void** showLeaderboard () {  
 **if** (**mGoogleApiClient**.isConnected()) {  
 **final int** BOARD\_REQUEST\_CODE = 1;  
 startActivityForResult(Games.***Leaderboards***.getLeaderboardIntent(**mGoogleApiClient**, getString(R.string.***leaderboard\_id***)), BOARD\_REQUEST\_CODE);  
 **this**.finish();  
 }  
  
 }  
  
 */\*\*  
 \* retry connection to the leaderboard manually  
 \*/* **public void** retry() {  
 **if**(!**mGoogleApiClient**.isConnected()) {  
 **mGoogleApiClient**.connect();  
 **if** (**mGoogleApiClient**.isConnected()) {  
 **retry**.setVisibility(View.***GONE***);  
 **if** (*prefs*.getString(**"action"**, **"test"**).equals(**"submit"**)) {  
 submit();  
 }  
  
 **if** (*prefs*.getString(**"action"**, **"test"**).equals(**"showLeaderboard"**)) {  
 showLeaderboard();  
 }  
 } **else** {  
 **this**.finish();  
 }  
 }  
  
  
 }  
}

**Activity\_google\_api.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/activity\_google\_api"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 android:background="#FFFFFF"  
 tools:context="com.odisee.project2game.GoogleApiActivity"**>  
  
 <**com.google.android.gms.common.SignInButton  
 android:id="@+id/sign\_in\_button"  
 android:layout\_width="200dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="173dp"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"** />  
  
 <**Button  
 android:background="@drawable/button\_selector"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/sign\_out\_button"  
 android:layout\_width="200dp"  
 android:text="Continue as another user"  
 android:textSize="15dp"  
 android:textColor="#000000"  
 android:textAlignment="center"  
 android:onClick="goButtonClicked"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="173dp"  
 android:visibility="gone"**></**Button**>  
  
  
 <**Button  
 android:background="@drawable/button\_selector"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/guest\_button"  
 android:layout\_width="200dp"  
 android:text="Continue as a guest"  
 android:textSize="15dp"  
 android:textColor="#000000"  
 android:textAlignment="center"  
 android:onClick="goButtonClicked"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="113dp"**></**Button**>  
  
 <**TextView  
 android:layout\_height="wrap\_content"  
 android:id="@+id/hello"  
 android:layout\_width="wrap\_content"  
 android:textColor="#000000"  
 android:text="Hello there!"  
 android:textAlignment="center"  
 android:layout\_marginBottom="34dp"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="65dp"**></**TextView**>  
  
 <**TextView  
 android:layout\_height="wrap\_content"  
 android:textColor="#000000"  
 android:id="@+id/who"  
 android:layout\_width="wrap\_content"  
 android:text="Who are you?"  
 android:textAlignment="center"  
 android:layout\_marginBottom="13dp"  
 android:layout\_marginTop="84dp"  
 android:layout\_centerHorizontal="true"**></**TextView**>  
  
 <**Button  
 android:background="@drawable/button\_selector"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/continueToGame"  
 android:layout\_width="200dp"  
 android:text="Continue to game"  
 android:textAlignment="center"  
 android:visibility="gone"  
 android:onClick="goButtonClicked"  
 android:layout\_alignBottom="@+id/sign\_in\_button"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="113dp"**></**Button**>  
  
 <**Button  
 android:background="@drawable/button\_selector"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/retry"  
 android:layout\_width="200dp"  
 android:layout\_centerHorizontal="true"  
 android:text="connect to online leaderboard"  
 android:textSize="15dp"  
 android:textColor="#000000"  
 android:textAlignment="center"  
 android:onClick="goButtonClicked"  
 android:visibility="gone"  
 android:layout\_marginTop="100dp"**></**Button**>  
  
</**RelativeLayout**>

## Other

**Android/build.gradle**

// part of code

dependencies {  
 compile **'com.google.android.gms:play-services-games:9.6.0'** compile **'com.google.android.gms:play-services-auth:9.6.0'**}  
apply **plugin**: **'com.google.gms.google-services'**

**Android/androidmanifest.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.odisee.project2game"  
 android:versionCode="3"  
 android:versionName="1.1"**>  
  
 <**uses-sdk  
 android:minSdkVersion="8"  
 android:targetSdkVersion="25"** />  
  
 <**android:uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"** />  
 <**android:uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"** />  
  
 <**application  
 android:allowBackup="true"  
 android:icon="@drawable/ic\_launcher2"  
 android:label="@string/app\_name"  
 android:theme="@style/GdxTheme"**>  
 <**activity  
 android:name=".AndroidLauncher"  
 android:configChanges="keyboard|keyboardHidden|orientation|screenSize"  
 android:label="@string/app\_name"  
 android:screenOrientation="landscape"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
 <**activity  
 android:name=".GoogleApiActivity"  
 android:label="@string/title\_activity\_google\_api"**></**activity**>  
 <**meta-data android:name="com.google.android.gms.games.APP\_ID"  
 android:value="@string/app\_id"** />  
 <**meta-data android:name="com.google.android.gms.appstate.APP\_ID"  
 android:value="@string/app\_id"** />  
 </**application**>  
  
</**manifest**>

**Strings.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**resources**>  
  
 <**string name="app\_name"**>Project2</**string**>  
 <**string name="title\_activity\_google\_api"**>GoogleApiActivity</**string**>  
 <**string name="app\_id"**>60658299545</**string**>  
 <**string name="leaderboard\_id"**>CgkImeWQ\_OEBEAIQBg</**string**>  
  
</**resources**>

**Core/build.gradle**

// part of code

buildscript {  
 dependencies {  
 classpath **'com.android.tools.build:gradle:1.5.0'** classpath **'com.google.gms:google-services:3.0.0'** }  
}

**GoogleServices.json**

//Generated code by Google Developers Console

{  
 **"project\_info"**: {  
 **"project\_number"**: **"60658299545"**,  
 **"project\_id"**: **"firebase-project2game"** },  
 **"client"**: [  
 {  
 **"client\_info"**: {  
 **"mobilesdk\_app\_id"**: **"1:60658299545:android:c2baa6ae1cf029c6"**,  
 **"android\_client\_info"**: {  
 **"package\_name"**: **"com.odisee.project2game"** }  
 },  
 **"oauth\_client"**: [  
 {  
 **"client\_id"**: **"60658299545-6r6dulvbbhs0nblbvfdhhmetilacr5q7.apps.googleusercontent.com"**,  
 **"client\_type"**: 1,  
 **"android\_info"**: {  
 **"package\_name"**: **"com.odisee.project2game"**,  
 **"certificate\_hash"**: **"6A2600621FC7F0F62A5E11B6D6141E98F3DBCD25"** }  
 },  
 {  
 **"client\_id"**: **"60658299545-0putb7c31nlmqg873ki8sj8crfqg7f2i.apps.googleusercontent.com"**,  
 **"client\_type"**: 3  
 }  
 ],  
 **"api\_key"**: [  
 {  
 **"current\_key"**: **"AIzaSyBAkiuqxEy4bILq\_vB2ufc8rFr2Ajx\_gHU"** }  
 ],  
 **"services"**: {  
 **"analytics\_service"**: {  
 **"status"**: 1  
 },  
 **"appinvite\_service"**: {  
 **"status"**: 1,  
 **"other\_platform\_oauth\_client"**: []  
 },  
 **"ads\_service"**: {  
 **"status"**: 1  
 }  
 }  
 }  
 ],  
 **"configuration\_version"**: **"1"**}