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

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

Contents

[Doelstelling 1 2](#_Toc470875604)

[Overzichtsrapport 2](#_Toc470875605)

[Algemeen 2](#_Toc470875606)

[Ontwikkeling 2](#_Toc470875607)

[Programmeertaal 2](#_Toc470875608)

[Hardware 3](#_Toc470875609)

[Doelstelling 2 3](#_Toc470875610)

[LibGDX 3](#_Toc470875611)

[Het spel 3](#_Toc470875612)

[Collision detection 4](#_Toc470875613)

[File i/o naar intern geheugen 4](#_Toc470875614)

[Graphics 4](#_Toc470875615)

[Camera 4](#_Toc470875616)

[Geluid 5](#_Toc470875617)

[Doelstelling 3 5](#_Toc470875618)

[Google API 5](#_Toc470875619)

[Auth API 5](#_Toc470875620)

[Games API 5](#_Toc470875621)

[Overzichtsrapport 6](#_Toc470875622)

[Algemeen 6](#_Toc470875623)

[Ontwikkeling 6](#_Toc470875624)

[Programmeertaal 6](#_Toc470875625)

[Hardware 7](#_Toc470875626)

[Opvolgingsdocument 8](#_Toc470875627)

[Bibliography 9](#_Toc470875628)

# 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.

### Algemeen

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 zelf een 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 | |

### Ontwikkeling

### 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 zou zijn [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 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 |
|  |  |  |

### 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 1 type device gerund wordt, de IPhone. 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 een iPhone zijn goed waardoor je zeker kan zijn dat de applicatie gedraaid zal kunnen worden.

**Vastellingen**

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

# 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 [1] . 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();

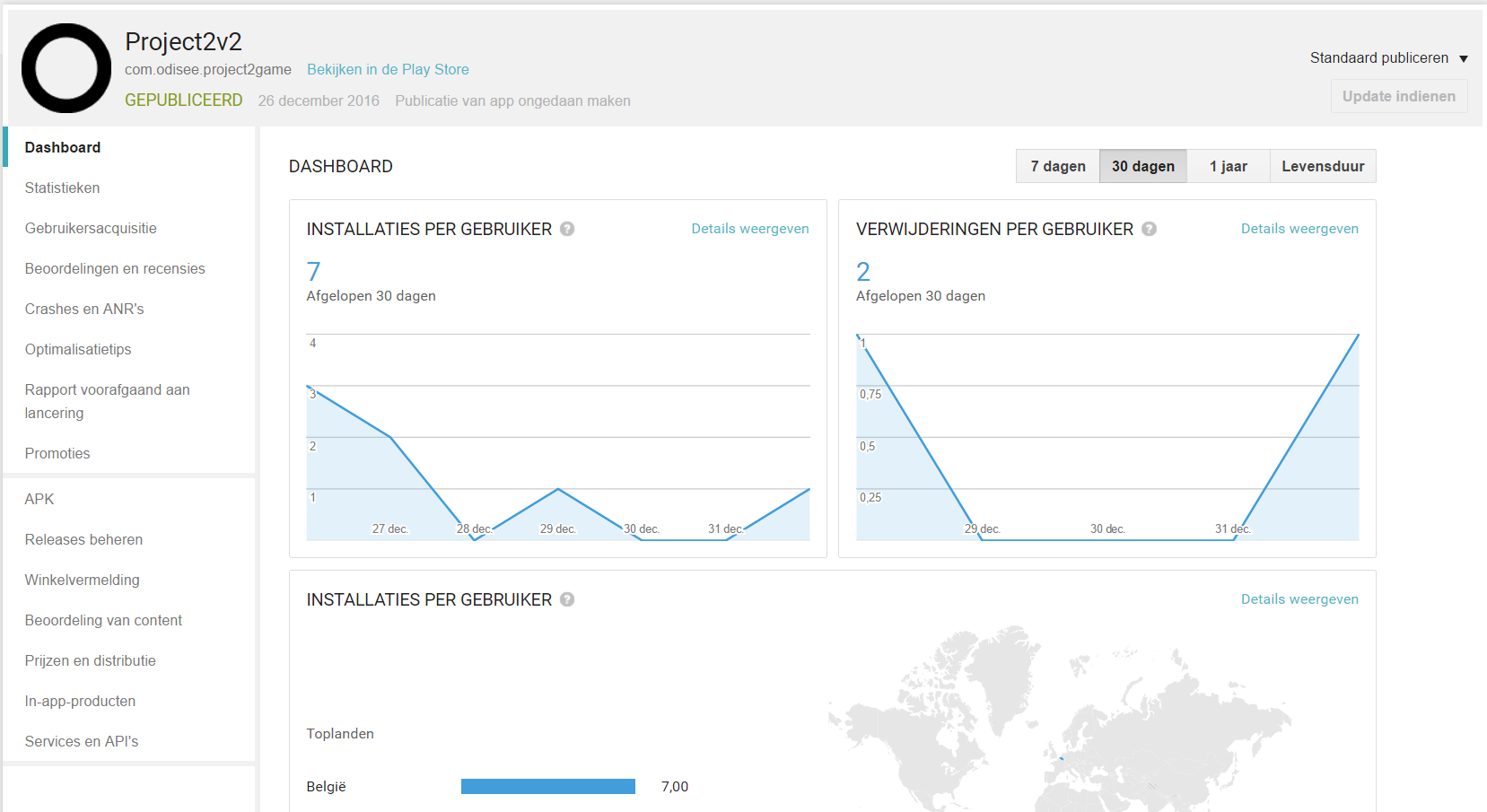
* OnConnected: deze methode wordt aangeroepen als de connectie geslaagd is, hierin zal dan ook het leaderboard opgeroepen worden en zullen scores naar het leaderboard verzonden worden.
* onConnectionFailed(): deze methode wordt aangeroepen als de connectie in onStart() mislukt is, er zal geprobeerd worden 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

# 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] | 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. |