# Definition mobile Applikation

# Konzeption und Implementierung einer Test-Applikation

Um die Möglichkeiten der gewählten Entwicklungstools tiefgehender analysieren zu können, sollten die theoretischen Informationen um praktische Erkenntnisse ergänzt werden. Demnach wird als Teil dieser Arbeit ein Beispiel-Spiel konzipiert, mit den jeweiligen Tools umgesetzt und daraus Applikationen für verschiedene Plattformen erzeugt. Die Anwendung soll dann Anhand von vereinbarten Metriken bemessen und analysiert werden. Die gewonnenen Resultate werden daraufhin in Zusammenhang mit den theoretischen Informationen komplementiert und die Entwicklungstools als Ganzes verglichen.

# Definition von Anforderungen

Applikationen aus den Kategorien Business, Unterhaltung und die meisten weiteren ohne den Fokus auf Spiele, greifen in der Regel auf plattformspezifische Benutzeroberflächen zurück. Native Apps besitzen jeweils technische und optische Konventionen. Cross-Plattform Apps versuchen diese möglichst genau zu generieren und einzuhalten, um dem Benutzer ein natives Look-and-feel zu bieten.

Meist folgen Spiele diesen Regeln nicht, da sie in Abhängigkeit des Spielprinzips auf unterschiedliche Eingabeelemente und UIs zurückgreifen.

Das Spiel als solches ist ein komplexes Phänomen, mit dem sich bereits Philosophen, Psychologen, Philanthropen und andere Forscher auseinandergesetzt haben. Ein Teilbereich der Spielwissenschaft, genannt Ludologie (Die Lehre über das Spiel), befasst sich unter anderem mit der Erforschung des digitalen Spielens. Die Forschungsinhalte befassen sich mit Fragestellungen aus den Bereichen:

Geschichte des digitalen Spielens

Elemente des digitalen Spiels

Begrifflichkeit zur Klassifizierung von Spielen und Genres

Regeln und Spielmechanik

Uvm.

%QUELLE: https://de.wikipedia.org/wiki/Ludologie

Wie definiert man aber nun den Inhalt und den Umfang eines Computerspiels?

Um sich in den konkurrierenden Theorien zu verlieren, wird hierfür beispielhaft die Gebrauchsdefinitionen von Salen und Zimmerman herangezogen:

„A game is a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome.”

%TODO: QUELLE: [Salen & Zimmerman 2004, S. 80]

Aus dieser Definition lassen sich wesentliche Konzepte entnehmen.

Spieler

Ein Spiel ist etwas, das ein oder mehrere Spieler aktiv spielen. Die Spieler interagieren mit dem System und Methoden des Spiels.

Konflikt

Der Konflikt bezeichnet den Wettkampfcharakter jedes Spiels. Dieser kann verschiedene Formen annehmen, wie kompetitive Elemente oder der Konflikt gegen das Spielsystem.

Regeln

Die Regeln liefern die Struktur des Spiels, in dem sie festlegen was der Spieler machen kann und was nicht.

Quantifizierbares Ziel

Das Ergebnis eines Spiels ist entweder das der Spieler gewonnen oder verloren hat, oder eine Form eines Erfolgsfaktors erhält. Dieser Erfolgsfaktor kann sich beispielsweise durch ein höheres Level oder eine Art Punktzahl bemerkbar machen.

%QUELLE: Rules of Play: Game Design Fundamentals

Anhand dieser Definitionen werden die Anforderungen für die Beispielapplikation konkretisiert:

Es soll eine Spielfigur geben, auf die der Spieler Einfluss nehmen kann.

Die Spielfigur soll auf Hindernisse treffen, welche die Aktionen des Spielers beeinflussen können.

Der Spieler gewinnt, wenn er ein definiertes Ziel erreicht oder das Spiel nicht beendet wird.

Der Spieler verliert, wenn er eine definierte Anzahl an Fehlversuchen erreicht hat, das Spiel abbricht oder andere Aktionen ausführt, die gegensätzlich zum gewinnen des Spiels stehen.

Der Spieler soll einen Erfolgsfaktor erhalten, um seine Spielweise zu messen.

Weitere allgemeine Anforderungen:

Das Spiel soll Grafiken oder 3D Objekte verwenden.

In dem Spiel sollen Animationen vorkommen, wie zum Beispiel Bewegungen.

Das Spiel soll über Audioelemente verfügen, wie Musik oder Soundeffekte.

Die Applikation soll zudem möglichst simpel gehalten werden, um sie Anhand der einfachsten und elementaren Eigenschaften zu bemessen.

Ein mobiles Spiel das mitunter durch seine Einfachheit, in kürzester Zeit großen Erfolg erlangte, ist das Spiel Flappy Bird.

%QUELLE: https://de.wikipedia.org/wiki/Flappy\_Bird

In diesem Spiel steuert der Spieler durch Antippen des Touchscreens die Flughöhe eines Vogels, um ihn vor dem Aufprall auf den Boden zu bewahren und durch entgegenkommende Hindernisse zu manövrieren. Hierbei handelt es sich um eine Art Endlos-Spiel, das sich dadurch auszeichnet, dass der Spieler solange spielt, bis er verliert. Verlieren kann man, indem die Spielfigur mit einem anderen Objekt kollidiert. Es verfügt über eine Punktzahl, die sich erhöht, wenn der Vogel erfolgreich die Hindernisse passiert hat.

In der Anlehnung zu Flappy Bird, entstanden zahlreiche Spiele die große Ähnlichkeit aufweisen. Auch wird dieses Spielprinzip häufig als Vorlage verwendet, um beispielhaft den Einstieg in die Spielentwicklung mit einer bestimmten Engine oder einem Framework zu erlernen. Da alle vorher genannten Anforderungen in dieses Spielprinzip hineinpassen, wurde ein Spiel konzipiert, das sich an dieser Vorlage orientiert. Das Spiel trägt den Titel Happy Bird.

%TODO: Spielfluss beschreiben!

\section{Spielfluss}

Das Spielprinzip und der Spielfluss bei Happy Bird sind denkbar einfach. Das Spiel wurde zudem auf den Portrait-Modus (Hochformat) festgelegt.

\begin{itemize}

\item Das Spiel wird über das App Icon gestartet.

\item Direkt nach dem Start wird ein Logo des jeweiligen Frameworks oder Engine angezeigt.

\item Daraufhin gelangt man in ein Hauptmenü, das den Titel des Spiels, Hintergrundgrafiken und einen Startbutton anzeigt. Die Hintergrundmusik wird abgespielt.

\item Bei betätigen des Startbuttons wird in die Spielszene gewechselt.

\item Die Spielszene beginnt mit den Hintergrundgrafiken und einem Vogel als Spielfigur.

\item Der Vogel verliert an Höhe, wenn der Spieler nichts tut und steigt bei antippen des Touchscreens an.

\item In einem festgelegten Intervall kommen der Figur nun Hindernisse in Form von grünen Röhren entgegen. Diese besitzen eine Lücke in der Mitte, durch die der Vogel hindurchmanövriert werden muss. Die Position der Lücke befindet sich in einer zufälligen, vertikalen Stelle innerhalb der Sichtbarkeit.

\item Wenn der Vogel erfolgreich das Hindernis passiert hat, erhält man einen Punkt, der oben links im Spiel zu der Gesamtpunktzahl hinzugefügt und dargestellt wird.

\item Das Spiel endet und gilt als verloren, wenn der Vogel mit einem der entgegenkommenden Hindernisse oder dem Boden kollidiert.

\item Der obere Rand der Spielszene wird durch eine unsichtbare Wand blockiert, die nicht passiert werden kann.

\item Bei verlorenem Spiel, wird in das Game Over Menü gewechselt, das wie das Hauptmenü aufgebaut ist. Dies zeigt den Game Over Status an und gibt die Möglichkeit, über den Startbutton ein neues Spiel zu starten.

\end{itemize}

%TODO: Screenshots!

Die in Kapitel \ref{sec:definiton\_anforderungen} definierten Anforderungen, wurden somit alle umgesetzt. Die Spielfigur wird durch einen Spritesheet animiert und ist durch einen Touchscreen-Input steuerbar. Es werden Hindernisse erzeugt, die bei Kollision das Spielende hervorrufen. Der Spieler hat die Möglichkeit Punkte zu erhalten.

Die Hintergrundmusik wird in allen drei Spielszenen gespielt. Sie startet bei Wechsel der Szene nicht von vorne, behält einen globalen Status bei und wird in einer Endlosschleife gespielt. Auf weitere diverse, für eine Testapplikation unnötige Benutzerannehmlichkeiten wurde verzichtet.

# Benutzte Werkzeuge

In diesem Kapitel werden die genutzten Softwarewerkzeuge aufgelistet, die bei der Entwicklung der Testapplikation eingesetzt wurden.

Tool Version

libGDX

Cocos2D-X

Unity3D

RoboVM

Mono

Xcode

Android Studio

Visual Studio

OS X

Windows

Java JDK

Android SDK

Android NDK

Ant

iOS SDK

Sonstiges

Python

C++

Testgeräte:

Android

iOS

# Eingesetzte Komponenten

Bei den eingesetzten und benötigten Komponenten existieren geringe Unterschiede. Um eine Sprite-Animation in Cocos2D und libGDX zu ermöglichen, empfehlen die Frameworks die Nutzung von Spritesheets in Kombination mit einer Property List-Datei (.plist). Bei einem Spritesheet handelt es sich um eine Sammlung von Grafiken in einer Datei. Diese Grafiken können allein oder in einer zusammenhängenden Abfolge stehen. Diese werden dann der Größe entsprechend ausgeschnitten. Durch Zuhilfenahme der Property List (Liste von Eigenschaften), die in Zusammenhang mit dem Spritesheet erstellt werden kann, können Informationen über die einzelnen Sprites ausgelesen werden. Diese geben Auskunft über Namen, Größe, Offset, Skalierung und Rotation der einzelnen Grafiken auf dem Spritesheet und sind auf Basis von XML-Dateien gespeichert oder binär kodiert. Innerhalb der Anwendung können die einzelnen Grafiken dann in einer einstellbaren Geschwindigkeit und Dauer nacheinander angezeigt werden. Diese Art von Animation ähnelt einem Daumenkino.

In Unity3D können ebenfalls Spritesheets genutzt werden, jedoch werden die Property Lists für die Teilung und den Informationshintergrund nicht benötigt. Komfortable Möglichkeiten bieten die sogenannten Prefabs. Ein Prefab kann spezifisch definierte Informationen und Eigenschaften über ein oder mehrere Gameobjekte tragen und diese in einem Objekt abrufbereit verfügbar machen. In Happy Bird werden die Pipe-Hindernisse beispielsweise als Prefab definiert, mit dem Inhalt der Grafiken, der Positionen und der Kollisionsboxen.

Grafiken

Name Verwendung Typ Datentyp Größe

Red\_Bird Spielfigur Spritesheet PNG

Sprite\_1 Spielfigur Sprite PNG

Sprite\_2 Spielfigur Sprite PNG

Sprite\_3 Spielfigur Sprite PNG

Sprite\_4 Spielfigur Sprite PNG

Bird Spielfigur Spritesheet PLIST

Sky Hintergrund Sprite PNG

Ground Hindernis Sprite PNG

Pipe Hindernis Sprite PNG

StartButton Button Sprite PNG

Marker Felt Schrift Font TTF

BGMusic Hintergrundmusik Audio MP3

Audio Interpret:

Kevin McLeod