ENTWICKLUNG EINES SPIELES FÜR DEN APP- UND DEN PLAY-STORE

Maturarbeit von Gabriel Sonderegger

Betreuung: Dr. R. Steiger
Kantonsschule Schaffhausen

2017

INHALT

DANKSAGUNG

Ich danke meiner Betreuungsperson Dr. Rainer Steiger für die Unterstützung während der Arbeit und allen Personen die mein Spiel heruntergeladen haben. Ausserdem Danke ich meinem Bruder und Vater für die Korrektur meiner Arbeit.

VORWORT

Für meine Maturarbeit wollte ich mich in ein Gebiet begeben, welches mir Spass macht.
Ich habe schon zuvor Spiele für den App Store und den Computer entwickelt und veröffentlicht, jedoch hatte ich bisher noch nie monetäre Strukturen in mein Programm integriert. Mein Ziel ist es deshalb das erste Mal ein Spiel zu entwickeln, dass nicht nur funktioniert und Spass macht, sondern auch einen Umsatz erzielt.

Ich werde den Prozess der Entwicklung einer mobilen Applikation beschreiben, welche dann auf zwei Betriebssysteme veröffentlicht wird. Für die Entwicklung der App werden zwei Betriebssysteme sowie diverse Werkzeuge gebraucht. Diese werde ich zuerst vorstellen. Die App wird dabei für iOS[1] und Android[2] entwickelt. Ich werde vor allem auch auf externe Strukturen wie die Integration von Werbung oder Konsumgütern eingehen. Dementsprechend wird in dieser Arbeit auch die Monetisierung der App behandelt. Am Schluss wird die App veröffentlicht und die Resultate der Downloadanzahl sowie der Umsatz werden analysiert.

DAS SPIEL

Das Spiel welches ich entwickelt habe ist ein Fertigkeitsspiel namens „**0123**"
Das Spiel besteht im Wesentlichen aus drei Szenen. **(**Abbildung 1**)**
Im Spiel soll dabei innert kurzer Zeit eine Ziffer in einer Ziffernmatrix gefunden werden. Diese Ziffer soll dann bevor die Zeit vorbei ist, gedrückt werden. Das Spiel wird graduell schwerer, sodass sich nicht nur die Ziffern, sondern auch die Farben vermischen.
Das Ziel des Spieles ist es, möglichst viele Punkte zu erreichen und seinen Highscore zu übertreffen.
Dabei ist das Spiel kostenlos und soll ein Online-Bestenliste, In-App-Produkte (digitale Produkte) und Werbung anbieten.

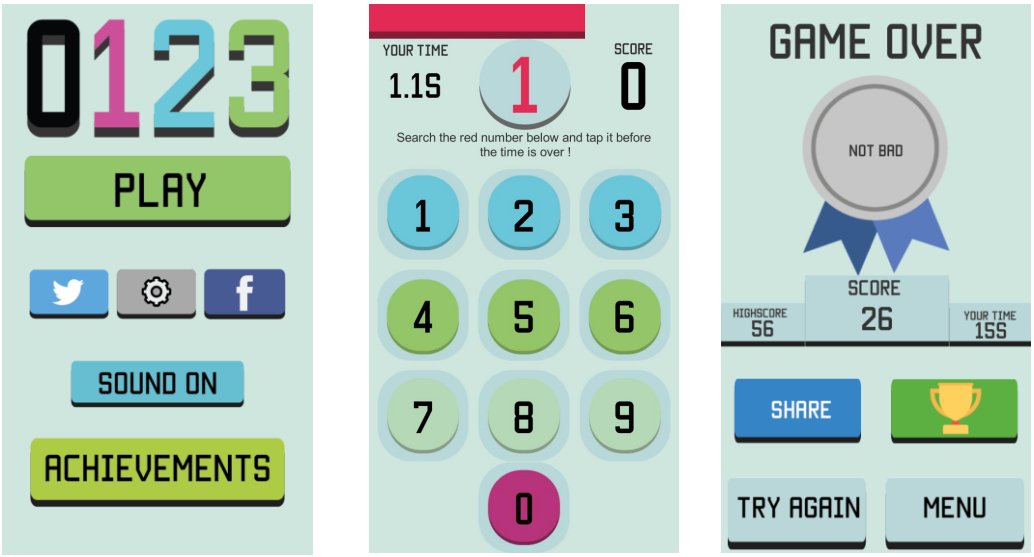


Abbildung 1 : Screenshots des Spieles 0123

[1] iOS: steht für i-Operating-System und ist das Betriebssystem von Apple für Mobilgeräte
[2] Android: das meist verbreitete Betriebssystem für Mobilgeräte

THEORIE

Vor dem Beginn der eigentlichen Entwicklung müssen Rahmenbedingungen für das Projekt aufgestellt werden. Zuerst soll man sich entscheiden, für welche Betriebssysteme die App entwickelt werden soll. Für dieses Projekt wurden die zwei meist verbreiteten Systeme „iOS" und „Android" ausgewählt.

Der einzige Store welcher Apple für Apps anbietet, ist der **App Store.** Für Android wird der grösste Android Store ausgewählt, der **Playstore** von Google**.** Folgend werden die zwei Stores kurz beschrieben.

DER APP STORE

Der App Store ist mit mehr als 2 Mio. Apps die zweit grösste Plattform um Apps herunterzuladen und bietet Apps für Applegeräte an. Jeden Tag werden durchschnittlich mehr als 1500 neue Apps hochgeladen und über 85% der Spiele sind kostenlos.
(Anon., 2017)

DER PLAYSTORE

Mit über 3.5 Mio Apps ist der Playstore der grösste Anbieter für Applikationen. Jeden Tag werden mehr als 3000 neue Apps hochgeladen und Mehr als 90 % der Spiele sind kostenlos
(Anon., 2017)

Nach der Wahl der Betriebssysteme und einem kurzen Einblick in die Stores, wird der Blick auf die verschiedenen Entwicklungen gerichtet. Apple und Google haben je Ihre eigenen Softwares und Stores um ein App zu entwickeln. Folgend werden die beiden Prozeduren beschrieben.

PROGRAMMIEREN IN DER IOS UMBGEBUNG

Die von Apple angebotene Software zur Entwicklung von Apps ist **XCode**. Sie unterstützt mehrere geläufige Programmiersprachen wie Phyton oder Javascript. Am häufigsten wird jedoch die eigens für iOS entwickelten Sprachen „Objective C" und seit 2014 „Swift" verwendet. XCode bietet auch eine stabile und einfache Weise an das User-Interface zu gestalten, sei es durch Programmierung oder mittels der graphischen Benutzeroberfläche. Ein markanter Nachteil ist, dass XCode nur auf einem Apple Gerät installiert werden kann.
Nach der Fertigstellung der App muss eine Lizenz von Apple erworben werden. Apple verlangt dafür eine jährliche Entwicklergebühr. Dafür wird die App von Menschen geprüft bevor sie auf den App Store hochgeladen wird.
Von einer Idee kann, mittels Software, das Spiel entwickelt werden (Abbildung 2). Auf dem Appstore kann der Benutzer dies dann herunterladen.

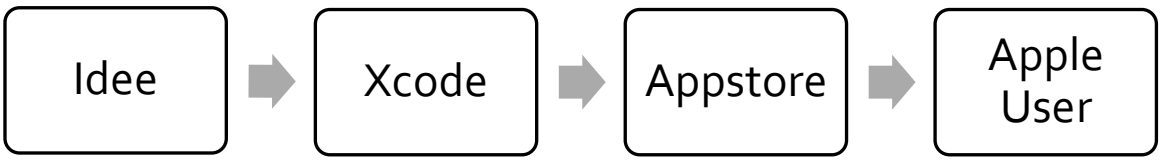


Abbildung 2 : Der Prozess für die Entwicklung einer iOS App.

ITUNES CONNECT / DEVELOPER PORTAL

Um das App hochzuladen und im Store zu registrieren werden von Apple zwei Werkezeuge in Form zweier Websiten angeboten: iTunes-Connect und das Developer Portal.

iTunes-Connect (Abbildung 3) ist die offizielle Online-Ablage von Apple um die Apps zu verwalten. Hier können zum Beispiel die Preise der Apps verändert werden oder Statistiken der Downloads eingesehen werden.

Das **Developer Portal** ist eine zusätzliche Website. Dort kann auf Ressourcen, die während der Entwicklung benötigt werden, zugegriffen werden.

Im Detail werden diese Werkzeuge nicht behandelt werden, sie werden in dieser Arbeit jedoch für die Behandlung der Konsumgüter und Bestenlisten wieder vorkommen.

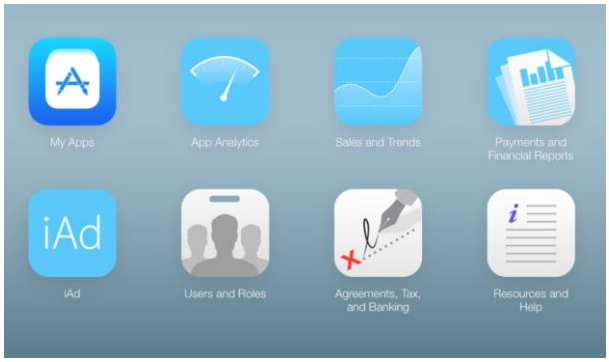


Abbildung 3 : Der Startbildschirm von iTunes-Connect

PROGRAMMIERUNG IN DER ANDROID UMBGEBUNG

Die Entwicklungsumgebung von Google ist: **„Android Studio".**
Diese Software unterstützt die Programmiersprache **Java**. Dabei ist zu beachten, dass Android das meist verbreitete Betriebssystem für Smartphones ist. Dementsprechend wird es von vielen Geräten genutzt. Daraus folgt eine grosse Bandbreite an Bildschirmgrössen und Versionen der Betriebssysteme. Bei der Veröffentlichung wird für den Google Play Store eine einmalige Entwicklergebühr verlangt. Nach dem Upload der App ist das Spiel sofort zum Download bereit.

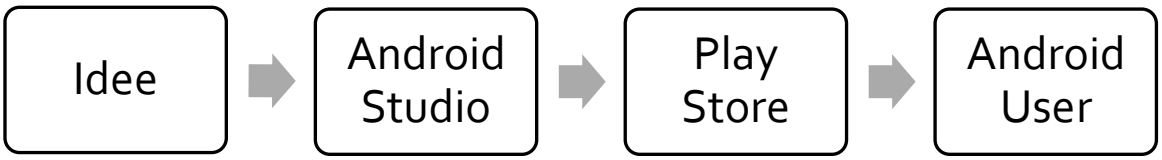


Abbildung 4: Der Prozess zur Entwicklung einer App für den Play Store.

Zwar ist der Prozess fast gleich wie bei iOS, jedoch ist markant, dass eine andere Software, ein anderer Store und ein anderer Endbenutzer vorhanden sind (Abbildung 4). Um das App hochzuladen und im Store zu registrieren bietet Google eine Plattform in Form einer Website an. Die **Google Play Console.**

GOOGLE PLAY CONSOLE

Analog zu Apples ITunes Connect, hat Google die Google Play Console, um das App im Play Store zu registrieren. Auch hier wird erst später in der Arbeit auf dieses Werkzeug zurückgegriffen.

PLATTFORMÜBERGREIFENDE ENTWICKLUNG

Eine Entwicklung für beide Betriebssysteme erscheint logisch, jedoch ist eine zweimalige Entwicklung aufwendig. Der Aufwand wäre mindestens doppelt so gross, würde man für beide Systeme unabhängig entwickeln wollen.
Eine Lösung, die es erlaubt das Projekt für den App- und den Android Store gleichzeitig zu entwickeln, wäre dementsprechend hilfreich.

Diese Art von Entwicklung wird plattformübergreifende Entwicklung oder auch **Cross-Plattform Development** genannt. Die Idee ist dabei sehr einfach. Das Spiel wird mit **einer Software** und **einer Sprache** entwickelt und mit der gleichen Software, für zwei Betriebssysteme exportiert (Abbildung 5).

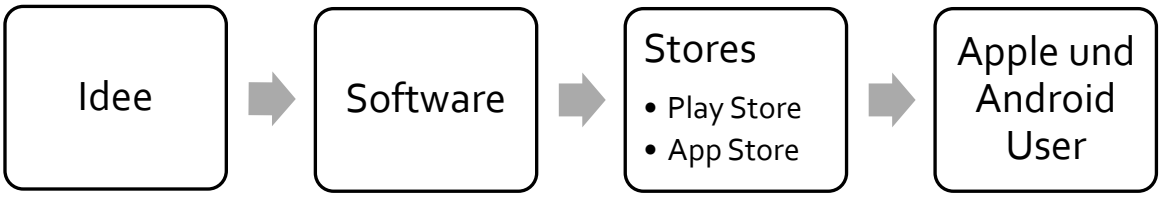


Abbildung 5, zeigt die Entwicklung einer App mittels Cross-Platform-Development

Zum Glück gibt es Softwares, die ein Cross-Platform-Development erlauben. In dieser Arbeit wird zwischen den Softwares Xamarin und Unity unterschieden, welche durch Recherche in die engere Auswahl traten. In einer kleinen Nutzwertanalyse werden die beiden Softwares gegenübergestellt. (Abbildung 6)

Software	**Muss Kriterien**		**Soll Kriterien**		
	Einfache Gestaltung des User Interface	Exportierung für iOS und Android	Für Spiele Entwickelt	Einfache Handhabung der Software (1-10)	Geeignete Libraries (1-10)
Unity	✔	✔	✔	**8**	**9**
Xamarin	✔	✔	✘	**6**	**7**

Abbildung 6 : Der Vergleich zwischen Unity und Xamarin

Beide Softwares erfüllen die gewünschten Bedingungen. Unity ist jedoch für die Entwicklung von Spielen spezialisiert und bietet mehr Bibliotheken[3] für die Entwicklung an. Als angenehmes Feature hat Unity auch eine freundlichere Benutzeroberfläche. Aufgrund des höheren Erfüllungsgrad der Kriterien bei Unity, wird dies auch als Entwicklungsumgebung für das Spiel genutzt. Folgend wird die Software kurz beschrieben.

UNITY 3D

Unity 3D ist eine Software, spezialisiert für die Entwicklung von Spielen. Ursprünglich für 3D Spiele entwickelt, unterstützt dieser Editor auch die Entwicklung von Spielen im zweidimensionalen Raum. Unity bietet auch eine Vielzahl von Bibliotheken an. So hat Unity für die Integration der Werbung, sowie der digitalen Produkte, einen kostenlosen Service bereitgestellt. Mit einem Unity-Konto kann von einer Website aus, dieser Service gesteuert werden.

Abbildung 7 zeigt das Unity Logo

Mit Unity wird auch die Software MonoDevelop installiert. MonoDevelop ist ein C# Editor[4], den man braucht um das Spiel zu programmieren.

[3] Bibliotheken oder Libraries sind Ansammlungen an Programmstrukturen, die als Hilfsmodule die Entwicklung vereinfachen.
[4] Im Editor kann das Programm geschrieben und kompiliert, also für den Computer übersetzt, werden.
Bild: https://roadtovrlive-5ea0.kxcdn.com/wp-content/uploads/2015/03/Unity-Logo-featured.png

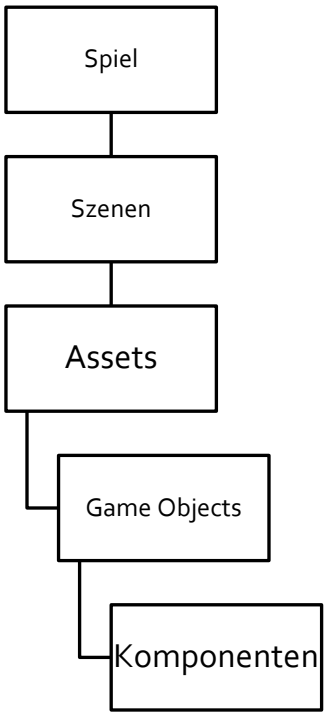


Abbildung 8 zeigt den Aufbau des Spieles

Ein **Spiel** besteht im Wesentlichen aus **Assets** und **Szenen.**

Die Bausteine für unser Spiel sind die **Assets.** Das sind alle Ressourcen wie Bilder oder Skripte.

Die **Szenen** werden durch diese Assets aufgebaut. Man kann eine Szene mit einem Level oder einem Menu des Spieles gleichstellen. Sie enthält alle Objekte und die zugehörigen Skripte, welche im Spiel verwendet werden. Unity kann nur eine Szene auf einmal abspielen, wird also eine neue Szene aufgerufen, so gehen nicht abgespeicherte Werte verloren.

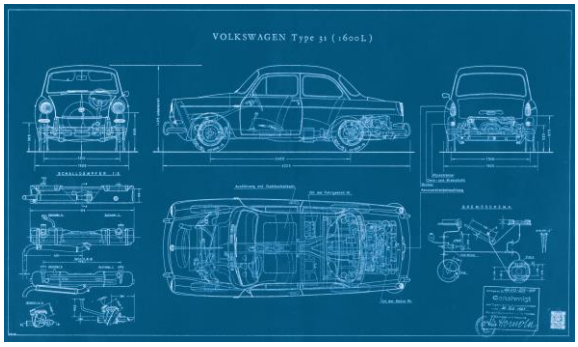
Ein Teil der Assets bilden die die **Game Objects**.
Die Game Objects sind dabei aktive Elemente die in die Szene implementiert sind. Ein Game Object besitzt einen Koordinatenwert und beliebig viele **Komponenten**. Die Komponente sind Skripte welche dem Game Object eine zusätzliche Funktion erteilen. So ist zum Beispiel ein Bild in Unity, ein Game Object mit der Komponente „Image", welches auf ein Bild im Asset Folder zugreift.

OBJEKTORIENTIERTES PROGRAMMIEREN UND KLASSEN

Die dabei benutze Programmierstruktur benutzt ein ähnliches Muster der Objekte. Diese Form zu programmieren nennt man **Objektorientiertes Programmieren (kurz OOP).**

OOP ist ein Programmierparadigma[5], indem sich sogenannte Objekte miteinander austauschen. Was dabei ein Objekt ist, wird nun anhand eines Beispiels erläutert.
Es wird nun ein kleiner Exkurs in ein Fantasiespiel gemacht, welches der Realität etwas näherkommt. Stellen Sie sich vor man würde ein Rennfahrerspiel entwickeln.

[5] Ein Programmierparadigma ist Programmierungsstil wie man ein Programm strukturiert



Konstruktor

Attribute:
Farbe: rot
Max v: 150 km/h

Weitere Objekte mit verschiedenen Attributen

Abbildung 9, Zeigt das Konzept des OOP, anhand eines Fahrzeuges in einem Rennfahrerspiel.

(hs-augsburg, 2014)

Das Objekt sei nun ein Fahrzeug. Für die Herstellung eines Objektes braucht es wie in der Realität einen Bauplan. Dieser Bauplan wird **Klasse** genannt.
Im Beispiel des Rennfahrspieles bestimmt die Klasse, wie die Fahrzeuge aussehen müssen und welche Geschwindigkeit sie erreichen können. Dabei legt die Klasse z.B. fest, dass zur Herstellung eines Autos ein Parameter „Maximalgeschwindigkeit" existieren muss. Jedoch hat die Klasse noch keine konkreten Werte.

Wird nun ein Objekt initialisiert wird durch den **Konstruktor** das Fahrzeug erstellt.
Der Konstruktor stellt dabei den Speicher im Programm frei und füllt dann im Initialisierungsprozess die Werte der Attribute des Objektes aus.
Ein Objekt erhält nun konkrete Werte. Dabei werden die Attribute des Fahrzeuges festgelegt, welche das Objekt eindeutig definieren.
Man kann nun weitere Fahrzeuge mit verschiedenen Werten initialisieren. Die Objekte unterscheiden sich dabei von ihren Werten und Attributen sind jedoch von der gleichen Klasse „Fahrzeugplan". Ein Fahrzeug kann dabei auch aus anderen Objekten bestehen. Weitere Objekte des Fahrzeuges wären etwa die Räder oder der Motor, welche wiederum selbst eigene Attribute besitzen.

Wurde nun ein Objekt erstellt, hat es verschiedene Aufgaben. So soll in diesem Beispiel das Fahrzeug beschleunigen, lenken und bremsen können. Solche Aufgaben nennt man **Methoden.** Die Methoden verlangen teilweise Parameter welche von anderen Objekten bereitgestellt werden. So ist zum Beispiel die Geschwindigkeit der Fahrzeuge von der Qualität des Motors abhängig. Die Objekte müssen also in der Lage sein sich gegenseitig auszutauschen.
Folgend werden die Methoden „`bremsen()`" und „`beschleunigen()`" dargestellt (Abbildung 10). Beide Methoden verlangen einen Parameter im Form eines Integers[6] für die Geschwindigkeit.

```
1. public void bremsen(int Geschwindikeit){
2.     Geschwindikeit --;
3. //eine lineare Bremsung erfolgt durch die Subtraktion der Geschwindi
   gkeit um -1
4. }
5. public void beschleunigen(int Geschwindikeit){
6.     Geschwindikeit ++;
7. //eine lineare Beschleunigung erfolgt durch die Addition
   der Geschwindigkeit um +1
```

Abbildung 10 : Die zwei Methoden, bremsen( ) und beschleunigen( )

[6] Ein Integer ist einen Variablentyp in der Programmiersprache mit dem Wert einer natürlichen Zahl.

Wird nun eine solche Methode im Objekt aufgerufen kann das Objekt initialisiert werden.

Genau die gleiche Art zu programmieren wird auch in der Entwicklung von „**0123**" genutzt. Im Rennfahrer Beispiel besteht ein Auto aus mehreren Objekten. In „0123" wird ein Knopf-Objekt aus Elementen wie einem Textfeld oder Bildern bestehen.
Bisher wurde das Konzept der Spielmechanik erklärt. Das Programm wird aber neben der Funktionalität des Spieles auch in der Lage sein, Werbung aufzuschalten. Folgend wird dieses Thema behandelt.

DIE WERBUNG

Die Werbung kann in „**Interstitial Ads**" und „**Rewarded Ads**" kategorisiert werden.

INTERSTITIAL ADS

„**Interstitial Ads**" ist der Begriff für Werbung, welche im Vollbildschirm Modus angezeigt wird. Das Spiel wird dabei pausiert. Die Werbung kann nach kurzer Zeit oder direkt übersprungen werden. Diese Art von Werbung unterbricht den Spielfluss direkt und kann für den Spieler mühsam sein.

REWARDED ADS

„**Rewarded Ads**" ist der Begriff für Werbung, welche den Spieler für das Aufschalten der Werbung belohnt. Beispielsweise kann in einem Spiel, ein neuer Charakter freigeschaltet werden. Die Rewarded Ads haben meist keinen Knopf um geschlossen zu werden. Vorteilhaft ist, dass diese Werbung vom Spieler gestartet werden muss. Folgend wird auch der Spielfluss deshalb nicht gestört.
(kiip, 2014)

Die Werbung wird in diesem Projekt über Unity integriert. Statistiken über die Werbung können dabei im **Unity Dashboard** unter https://dashboard.unityads.unity3d.com abgerufen werden. Eine weitere Möglichkeit das Spiel zu monetisieren, sind digitale Produkte oder **In-App-Produkte**.

IN-APP-PRODUKTE

Die In-App-Produkte können in verschiedene Kategorien aufgeteilt werden. Wichtig ist der Unterschied zwischen **Consumable** und **Non-Consumable** Produkten.
Ein Consumable Produkt kann wie der Name schon sagt, konsumiert, also aufgebraucht werden. Deshalb kann dieses Produkt auch mehrmals gekauft werden.
Ein Non-Consumable Produkt jedoch, wird einmal gekauft und gehört dann dem Benutzer.

Abbildung 11 : In-Game-Store des Spieles Clash of Clans

Ein Beispiel für ein Consumable Produkt ist der Kauf von einer digitalen Währung in einem Spiel. Zum Beispiel können im Spiel **Clash of Clans** [7] Diamanten für einen gewissen Preis erworben werden. Hat man einmal einen Sack Diamanten gekauft, kann dies beliebig oft wiederholt werden.

[7] Clash of Clans ist ein Mehrspieler-Online-Strategiespiel für Mobilgeräte und finanziert sich hauptsächlich mit In-App-Produkten.
Bild : https://theyounglionscoc.files.wordpress.com/2014/06/clash-of-clans-gems.png

Ein typisches Beispiel für ein Non-Consumable Produkt wäre das „Wegkaufen" der Werbung. Für einen gewissen Betrag wird auf der Software nie wieder Werbung aufgeschaltet.
(Apple, 2017)

DIE ENTWICKLUNG DES SPIELES

Wir starten nun mit der Entwicklung des Spieles. Das Projekt wird dabei in drei parallele Prozesse unterteilt:

I. Die Programmierung der Logik
II. Das Design
III. Das Erstellen der Meta Daten

Die Logik ist das Programmieren des Spiels. Sie ist der Prozess, der die Funktionalität der App gewährleistet. Für dieses Projekt werden ungefähr zehn Skripte mit ungefähr je zweihundert Zeilen geschrieben.

Das Design ist der Prozess zur Erstellung der erforderlichen Ressourcen sowie der Planung. Neben der Erstellung der einzelnen Bilder, wird hier darauf geachtet, dass das Spiel nicht nur spielbar ist, sondern auch Spass macht.

Das Erstellen der **Meta Daten** beinhaltet alle Hintergrund-Prozesse, wie die Erstellung von Bestenlisten oder Produkte. Die Metadaten werden mithilfe der Logik im Spiel integriert. Dieser Prozess ist der Zeitaufwändigste. Aufgrund der zwei Betriebssysteme muss alles doppelt ausgeführt werden

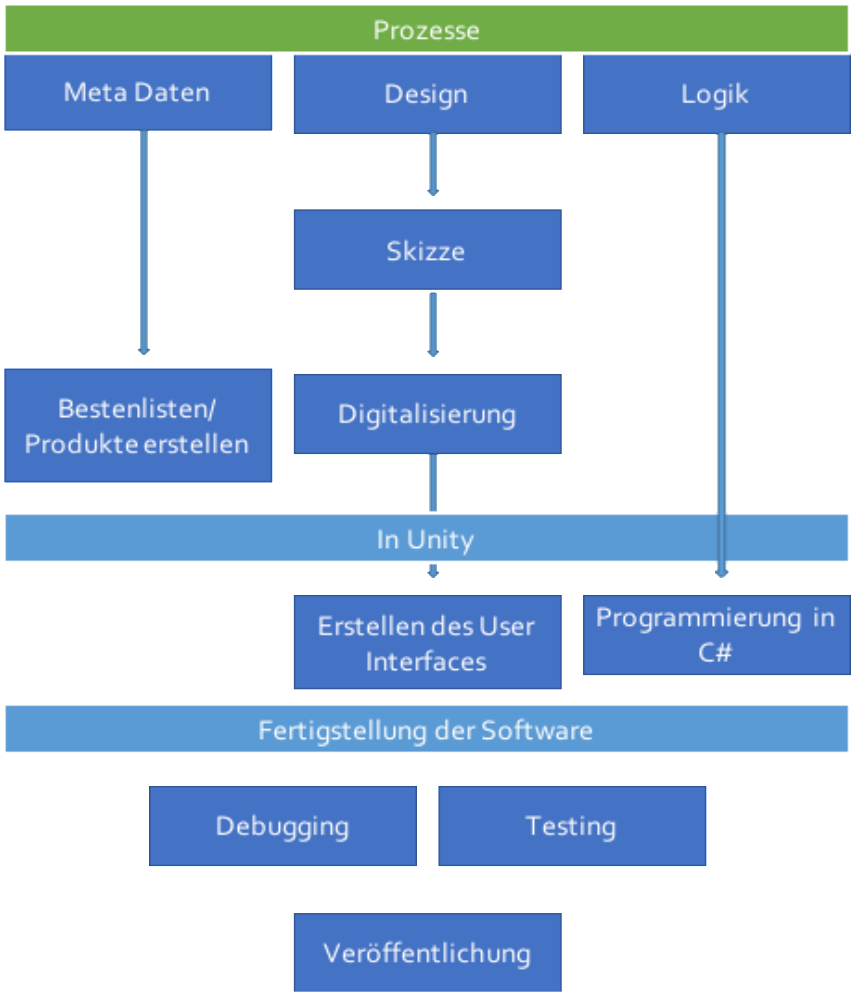


Abbildung 12, Die Prozesse der Spielentwicklung

Nach der Entwicklung, muss das Spiel durch das **Debugging** getestet werden. Das Debugging ist der Prozess, um allfällige Fehler zu entfernen. Dieser Schritt benötigt meistens am meisten Zeit da teilweise mit jedem gelösten Fehler zwei neue dazukomme. Da viele Fehler während der Entwicklung noch unbekannt sind, muss das Spiel getestet werden. Dafür wird die Software vor der Veröffentlichung an Aussenstehende verteilt. In dieser Arbeit werden dies Familienmitglieder oder Freunde sein.

DAS DESIGN

Damit das Spiel professionell aussieht, ist es wichtig auf ein gutes und einheitliches Design zu achten. Um die Einheit des Designs zu gewährleisten, werden folgende Randbedingungen aufgestellt.

- Das Spiel soll eine reduzierte Farbpalette und wenig Bedienknöpfe enthalten.
- Das Spiel soll während der ganzen Benutzung, mit nur einem Finger bedienbar sein.

Zu Beginn wird das Design auf Papier aufgezeichnet. Es werden die einzelnen Bildschirme des Spieles grob skizziert und die wichtigsten Knöpfe und deren Funktionen aufgeschrieben.

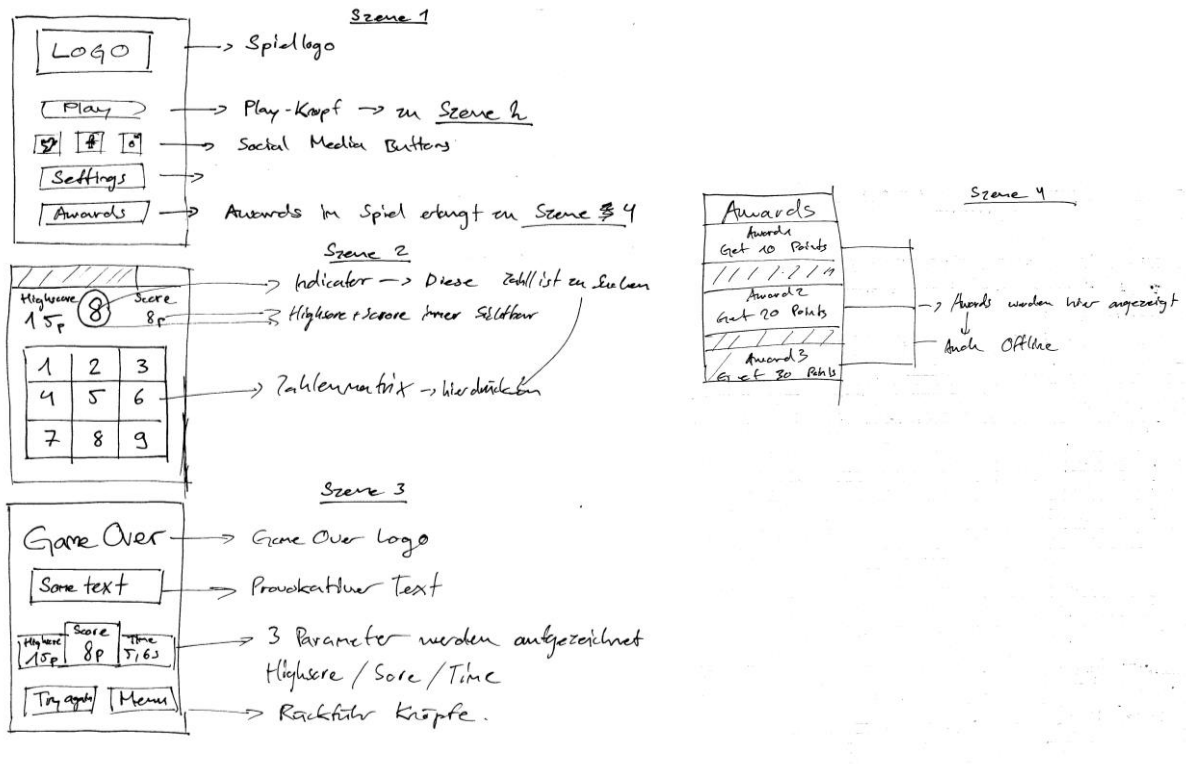


Abbildung 13 : Erste Skizzen für das Spiel

Als nächstes werden diese Skizzen digitalisiert. Dafür wird die Software **Illustrator von Adobe** benutzt. **Adobe Illustrator** (kurz AI) ist ein vektorbasiertes Grafik- und Zeichenprogramm. Es dient dem Herstellen von Computergrafiken, die man ohne Qualitätsverlust beliebig in ihrer Grösse verändern kann – im Unterschied zu pixelbasierten Bildbearbeitungsprogrammen wie z. B. Photoshop (Wikipedia, 2017)
Nachdem das Design digitalisiert ist, werden aus den einzelnen Elemente der Graphik ein **Sprite-sheet** erstellt werden. (Abbildung 14)
Das Sprite-Sheet ist eine Ansammlung von graphischen Elementen in einer einzelnen Datei. Unity wird daraus später wieder einzelne Bilder erstellen. Dies dient der Übersichtlichkeit und der Grösse des Projektes. Wichtig ist, dass die graphischen Elemente voneinander getrennt werden. Am einfachsten geht dies, indem ein durchsichtiger Abstand zwischen den Elementen übriggelassen wird.

Unity wird so, aus dem ein Element machen, was von einem leeren Alphakanal[8] umgeben ist. Da nicht jedes Bildformat diesen Parameter unterstützt werden meine Bilder deshalb alle als PNG[9] Dateien exportiert.

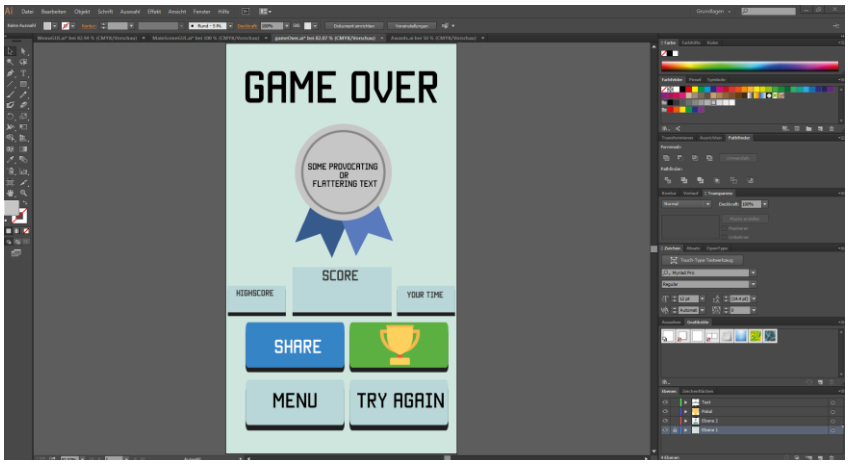


Abbildung 14 : Design des „Game Over“ in Illustrator

Abbildung 15 : Ein Sprite Sheet für die Szene der Errungenschaften

DER AUFBAU IN UNITY 3D

In Unity wird ein neues Projekt namens **0123** erstellt. Die erstellten Bilder werden dabei in den Ordner **Assets** geladen. Wie schon im Design Prozess skizziert, werden die einzelnen Szenen nun mithilfe der Bilder aufgebaut.

ERSTELLEN DER SZENEN

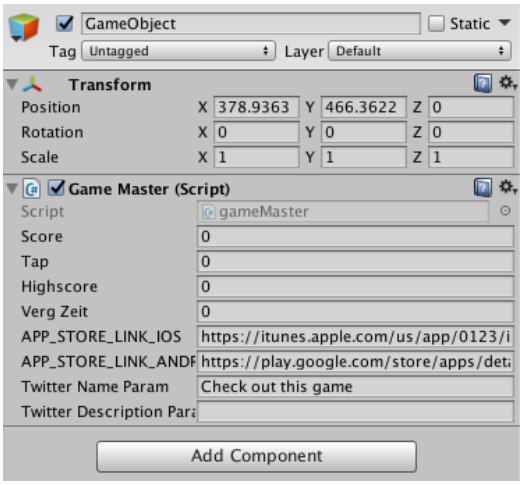


Abbildung 16 , Ein GameObject mit einer Koordinatenkomponente und einem Script

Mit der Vorlage der Skizzen erstellt man die verschiedenen Szenen aus GameObjects.
Die GameObjects sind dabei die Bausteine für das gesamte Projekt. Anfangs sind sie leere Objekte, welche mit der Zeit mit Komponenten gefüllt werden.
Einige Komponenten und fertige GameObject-Vorlagen bietet Unity dabei schon an. Diese Vorlagen können beliebig verändert und erweitert werden und beschleunigen den Prozess einer Szenenerstellung erheblich.

Mithilfe der Vorlagen und Skizzen werden die Elemente der Szenen nun einzeln nacheinander aufgebaut. Nachdem man die GameObjects erstellt und editiert hat, müssen noch die Abstandsverhältnisse von einem Objekt zum anderen definiert werden. Je nach Aufösung sowie **Orientierung** des Gerätes muss sich die Szene dynamisch anpassen können.

[8] Der Alphakanal ist ein Parameter welcher die Transparenz der Bilder bestimmt
[9] PNG ist ein Bidlformat mit einem Alpharaumparameter und einer verlustfreien Datenkompression.

In Abbildung 17 sieht man ein Beispiel zweier Knöpfe die ein bestimmtes Verhältnis zueinander haben. Durch das Setzten von Attributen kann eine Szene für verschiedene Smartphonesbildschirme designt werden.In Unity werden dafür **Layout-Komponenten** genutzt.Diese definieren die Abstandsverhältnise der untergeordneten Objekte. Deshalb muss die Szene hierarchisch Strukturiert werden. Manche Elemente bestimmen die Grösse und das Verhältnis anderer Objekte. So würde man im Beispiel der Abbildung 17 die Knöpfe 1 und 2, in ein übergeordnetes Game Object packen und dieses in der Szene zentrieren, so dass es immer mittig ist. Dies mag etwas kompliziert klingen deshalb muss für das Verständniss das Parent-Child Prinzip erläutert werden.

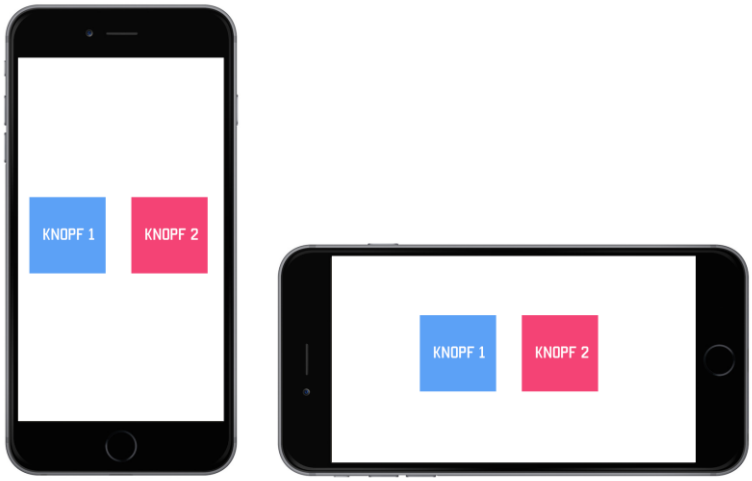


Abbildung 17 Verschiedene Orientierungen eines Smartphones und 2 zentrierte Knöpfe

PARENT / CHILD

Bei einer hierarchischen Struktur wie in der beschriebenen Szene, wird zwischen Parent und Child unterschieden. Ein Parent hat ein oder mehrere Children über die er verfügt. Werden Werte, wie z.B. die Koordinaten im Parent verändert, so werden die Werte aller Children auch verändert. Wird jedoch ein Wert im Child verändert hat das keinen Einfluss auf die Werte des Parents.

Durch diese Struktur kann man der Szene eine **Baumstruktur** geben um die Grössen und Verhältnisse der einzelnen Elemente zu steuern. Abbildung 18 zeigt dabei eine Szene im Spiel 0123. In der mittleren Navigationsleiste sieht man die Elemente hierarchisch dargestellt. Das GameObject „Main Canvas" enthält drei untergeordnete Children: „TimeBar" „Upper Text" und "Buttons". Diese Elemente enthalten wiederum andere Elemente. **Die Vertical Layout Group-Komponente** reiht die drei Elemente von der oberen Mitte nach unten. Diese Objekte selber haben auch Layout Komponenten um die Szene vollständig zu gliedern. Dieser Prozess wird wiederholt bis die Szene alle Elemente automatisch anordnen kann.

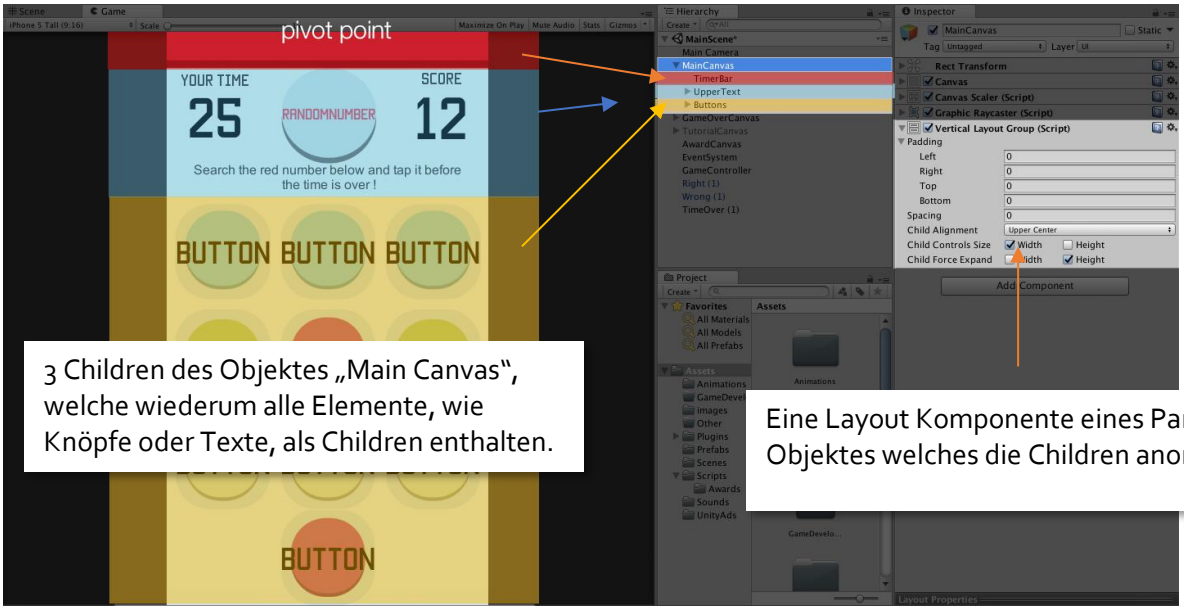


Abbildung 18 Eine Szene in Unity gegliedert in Parent und Child Objekte sowie einer Layout Komponente des Parent „Main Canvas"

PROGRAMMIERUNG DER SPIEL MECHANIK

Nach dem vollständigen Erstellen der Szenen in Unity kann das Spiel programmiert werden. Die Programmierung ist in zwei Abschnitten aufgeteilt. Zuerst wird die Spielmechanik entwickelt, da ohne die das Spiel gar nicht spielbar wäre. Hat man eine funktionierende Mechanik, können die externen Faktoren wie Werbung und Bestenlisten integriert werden.

Ein Script kann in Unity direkt erstellt werden und durch einen externen Editor[10] bearbeitet und kompiliert werden. Erstellt man eine C# Script direkt in Unity wird eine Standard Vorlage geladen. Ein leere Unity-C# Klasse enthält automatisch die zwei Funktionen `Start ()` und `Update ()` und wird von der Klasse MonoBehaviour erweitert (Abbildung 19).

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class Test : MonoBehaviour {
6.
7.     // Use this for initialization
8.     void Start () {
9.
10.         }
11.
12.         // Update is called once per frame
13.         void Update () {
14.
15.         }
16.     }
```

Abbildung 19

`Start()` ist die Funktion welche aufgerufen wird, wenn die Szene mit dem enthaltenen Programm aufgerufen wird. `Update()` wird jede Frame aufgerufen, alles was kontinuierlich verändert werden muss wird in dieser Funktion integriert.
Für das Verständnis des nächsten Teiles muss folgende Terminologie definiert werden.

- Das Script **GameMaster** ist in allen Szenen vorhanden und enthält alle wichtigen globalen Variabeln und Funktionen des Spieles.
- Das Script **ChangeNumbers** steuert die Hauptszene des Spieles.
- Das Script **GUI** steuert die Veränderung aller graphischen Elemente in der Hauptszene
- Das **Purchaser** Script wird für den Kauf eines digitalen Konsumguts gebraucht.

[10] In diesen Fall benutzen wir den Editor Mono-Behaviour

DAS CHANGENUMBERS SCRIPT:

Im Spiel wird im oberen Bereich des Bildschirms eine Zahl angezeigt, welche gesucht und möglich schnell gefunden werden soll. (Abbildung 20).

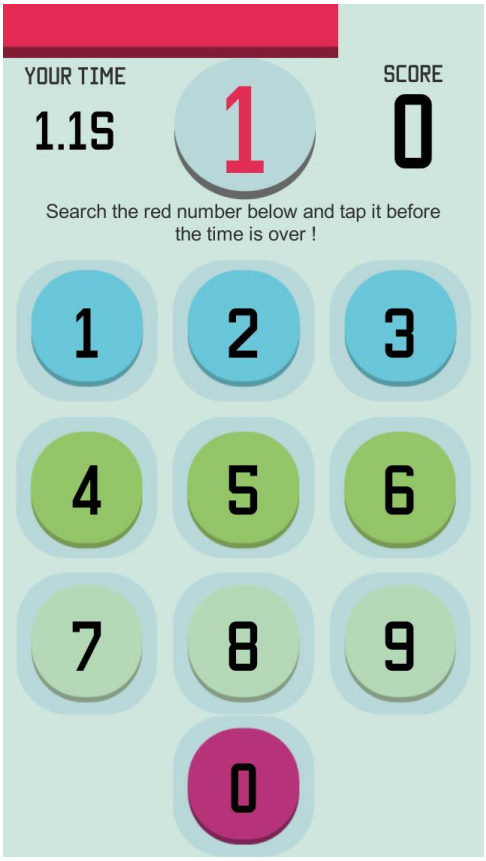


Abbildung 20, Die Hauptszene des Spieles

Ein wichtiger Aspekt des Spieles ist, dass es dabei kontinuierlich schwerer wird. Zu Beginn ist nur der Zeitdruck ein Hindernis. Mit der Zeit sollen sich die Zahlen und dann auch die Farben vertauschen. Die Zahlen sowie die Farben sind dabei in je einem Array[11] gespeichert.

Wir benutzten für das Vertauschen den **Knuth Shuffle Algorithmus** um allfällige Wiederholungen zu vermeiden.

KNUTH SHUFFLE

Nehmen wir an wir haben eine endliche Zahlenfolge und starten in der Position 0.

0123456789

Wir nehmen nun zufällig eine Position zwischen 0 und 9, sei dies zum Beispiel 5.

0123456789

Die Zahl der fünften Position wird nun mit der Zahl der Position 0 ausgetauscht. Der neue Array sieht nun folgendermassen aus:

5123406789

Dieser Schritt wird nun wiederholt, nur dass man anstatt einer zufälligen Zahl zwischen 0 und 9 eine Zahl zwischen 1 und 9 auswählt. Wird die gleiche Position ausgewählt verändert sich die Zahl nicht wird eine andere Zahl ausgewählt ändert sich die Zahlenfolge. Mit dieser Methode iterieren wir durch den Array, bis man den ganzen Array durchmischen konnte (Wikipedia, 2017).

Programmiert sieht der C# Code so aus:

```
1. public void reshuffle(int[] array)
2.     {
3.         // Knuth shuffle algorithm
4.         for (int i = 0; i < array.Length; i++ )
5.         {
6.             int tmp = array[i];
7.             int r = Random.Range(i,array.Length);
8.             array[i] = array[r];
```

[11] Ein Array ist eine Variable die mehrere Werte enthaltet.

```
9.              array[r] = tmp;
10.               }
11.      populuateNumbers();
12.         }
```

Abbildung 21 zeigt die Reshufflefunktion

Wir übergeben der Funktion einen Array vom Typ Integer. In einer Schleife wird jedes Element des Arrays von 0 bis zum letzten Element durchgenommen. Die Variabel „i" gibt dabei die Position des Wertes im Array an, „array.Length" die Länge des Arrays . Der Wert wird zuerst in eine temporäre Variable zwischengespeichert. Einer weiteren Variable „**r**" wird ein zufälliger Wert zwischen i und „array.Length" zugewiesen. In der Zeile 8 und 9 werden die Werte von der Position i und der Variabel r ausgetauscht. Die Funktion populateNumbers() füllt die Werte in die Text-Objekte in der Szene ein.

```
1. void populuateNumbers(){
2.         int index = 0;
3.         foreach(Text text in ArrayOfTexts){
4.             ArrayOfTexts[index].text = digits[index].ToString();
5.             index += 1;
6.         }
7.     }
```

Abbildung 22 : Die Funktion populateNumbers(), welche die Zahlen in das Spiel einfügt

Analog kann der gleiche Algorithmus für Arrays von anderen Typen benutzt werden, wenn man den Typus der Variablen und des Parameters verändert.
Die Reshuffle Funktion soll erst nach einer gewissen Zeit und nach der Auswahl einer Zahl aufgerufen werden. Die Funktion `clickOnButton()` ist für die Aktionen zuständig, welche nach der Auswahl einer Zahl geschehen sollen. In der folgenden Abbildung sieht man einen Ausschnitt dieser Funktion.

- In der Zeile 3 werden die Werte des Arrays graphisch angezeigt.
- In Zeile 4-10 kommen die Shuffle-Algorithmen ins Spiel.

Zuerst wird angefragt ob der Spieler die richtige Zahl gedrückt hat. Ist dies der Fall wird verglichen wievielmal der Spieler schon die Zahl drückte. Danach kommt die Funktion „reshuffle()" und „recolour()" die sich nur im Typus des verlangten Parameters unterscheiden.

```
1. public void clickOnButton(GameObject Text){
2.         if(!lost && PlayerPrefs.GetInt("Tutorial") != 1){
3.         int TextNumber = int.Parse((Text.GetComponent<Text>()).text)
   ;
4.         if(TextNumber == randomNumber){
5.             if ((gm.Tap) >= 3){
6.                 if(!lost){reshuffle(digits);}
7.             }
8.             if ((gm.Tap) >= 10){
9.                 if(!lost){recolour(colors);}
10.                }
11.                //Set Score
```

Abbildung 23: Die Funktion clickOnButton() welche bei Knopfdruck einer Ziffer aufgerufen wird.

Weitere Details der Spielmechanik werden in dieser Arbeit nicht behandelt. Der zweite Teil des Programmes behandelt, die Integration der Externen Service wie Werbung, Bestenlisten sowie digitale Produkte. Deshalb wird die App im nächsten Schritt in den Stores registriert.

REGISTRIEREN DER APP IM APP-STORE UND IM PLAY-STORE

Im zweiten Teil des Programmierungsprozesses, werden zuerst die zugehörigen Meta Daten erstellt, damit im Code darauf zugegriffen werden kann. Als erstes muss die App im Store registriert werden.

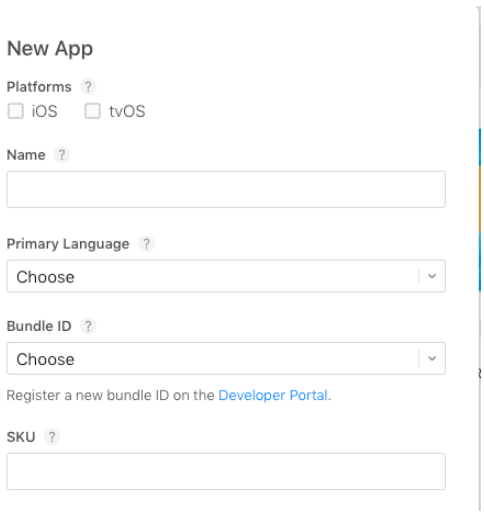


Abbildung 24 Die verlangten Parameter bei der Erstellung einer App auf ITunes-Connect

Auf ITunes-Connect erstellt man unter „Meine Apps" eine neue App mit dem „+" welches oben links angezeigt wird.
Es werden nun mehrere Parameter verlangt:

- Plattform
- Name
- Sprache
- Bundle ID
- SKU

Die Plattform wird in diesem Fall iOS sein. Der Name des Spieles ist **0123** und die Sprache ist Deutsch. Die zwei letzten Parameter müssen jedoch folgend genauer erläutert werden.

BUNDLE IDS & SKU

Das Spiel wird während der Benutzung, auf den registrierten Eintrag im Store zugreifen. Damit das Spiel den Eintrag auch eindeutig identifizieren kann, hat jede App eine Adresse in Form einer ID.

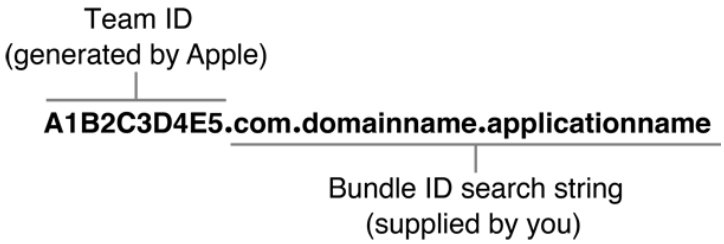


Abbildung 25, der Aufbau einer App-ID

Im Developer Portal wird für das App eine **App ID**[12] erstellt (Abbildung 25).
Der erste Teil, also die Team ID, wird von Apple selbst erstellt und ist unveränderbar.
Der zweite Teil der App ID ist die **Bundle ID**.
Die Bundle ID wird vom Entwickler erstellt und hat normalerweise das Format einer umgekehrten Domain. Für dieses Projekt wird die Bundle ID **"com.GSStudios.Game0123"** verwendet.
Diese ID wird zum Beispiel benutzt, um im Programm Konsumgüter zu kaufen.
Eine **Stock Keeping Unit** oder kurz **SKU** ist eine einmalige Verfolgungsnummer einer App im Appstore. Der wesentliche Unterschied bei der SKU und der Bundle ID ist, dass sie nicht alphanumerisch ist. Vergleichbar wäre diese mit einem Barcode eines Produktes im Supermarkt. Für das Programm ist dies jedoch nicht weiter relevant.
(Apple, 2015) (Langley, 2015)

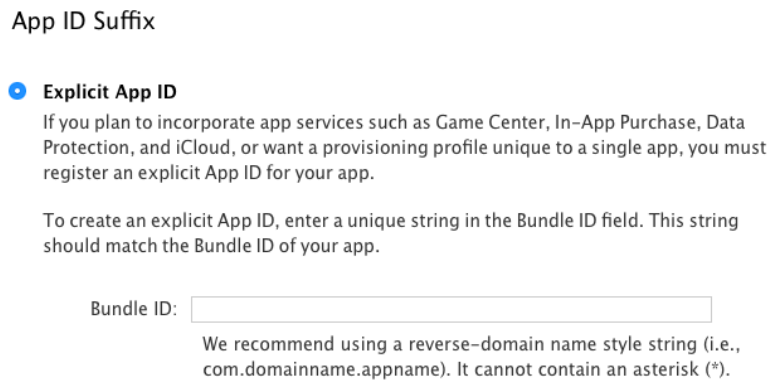


Abbildung 26 Beim einer neuen App ID wird eine Bundle ID vom Entwickler erstellt.

[12] Eine App ID ist ein 2-stelliger String, um ein oder mehrere Apps von einem Entwicklungsteam zu unterscheiden.

GOOGLE PLAY CONSOLE

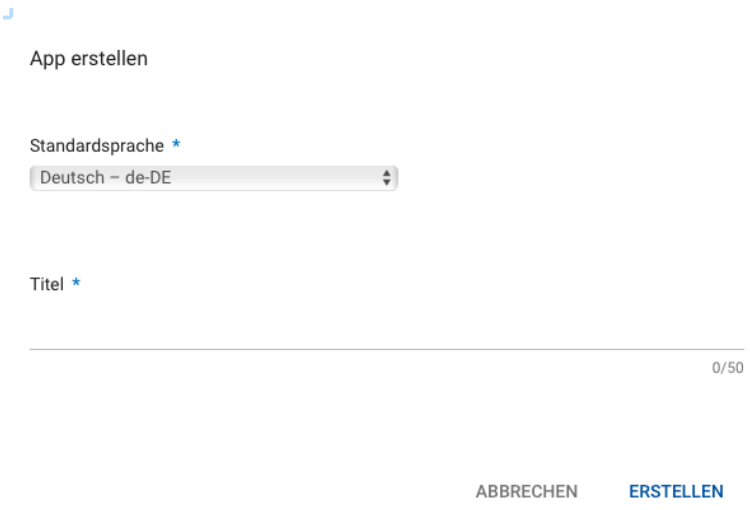


Abbildung 27 Das Eingabefeld bei der Erstellung einer neuen App in der Google Play Console

Analog muss die App auch im Play Store registriert werden. In der Developer Console wählt man dafür oben rechts „Neue App erstellen" aus. Vorerst muss nur ein Titel eingefügt werden.
Das App wäre im Playstore soweit schon registriert. Anders wie bei Apple muss jedoch zuerst eine Beta-Version der App auf die Plattform hochgeladen werden. Dabei werden auch die Werte wie eine Bundle ID mitgegeben. Dieser Teil wird jedoch erst im Kapitel der Veröffentlichung behandelt.

IN-GAME-PRODUKTE UND WERBUNG

Unity bietet wie schon erwähnt eine grosse Bandbreite an Bibliotheken und Service an. Hat man ein Unity-Konto erstellt kann man im Fenster **„Service"** durch einen Knopfdruck die Werbung integrieren.

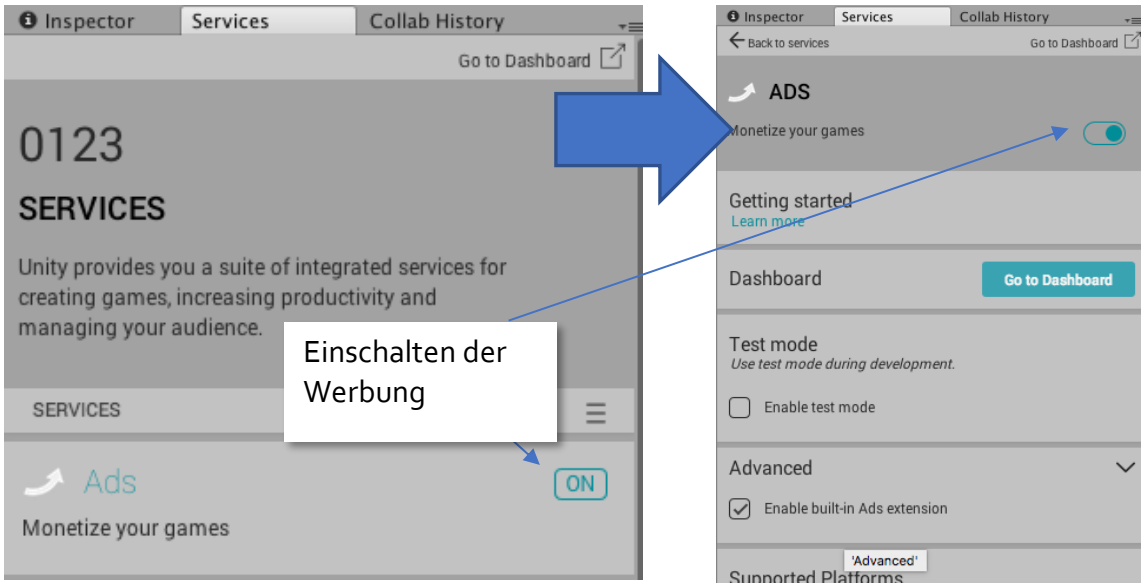


Abbildung 28 das Einschalten der Werbung-Library in Unity

Das folgende Programm wird die Werbung im Spiel initialisieren und zwei öffentliche Funktionen bereitstellen. Diese werden dann an den benötigten Stellen aufgerufen.
Die Initialisierung sowie die Funktionen für den Aufruf der Werbung im Spiel werden folgend gezeigt. (Abbildung 29)

```
1.  using System.Collections;
2.  using UnityEngine;
3.  using UnityEngine.Advertisements;
4.
5.
6.  public class gameMaster : MonoBehaviour {
7.
8.      void Start(){
9.          Advertisement.Initialize("1388654", false);
10. }
11.     public void ShowAd()
12.     {
13.         Debug.Log("Showing an Ad");
14.         Advertisement.Show();
15.     }
16.
17.     public void ShowRewardedAd()
18.     {
19.         if (Advertisement.IsReady("rewardedVideo"))
20.         {
21.             var options = new ShowOptions { resultCallback = HandleShowResult };
22.             Advertisement.Show("rewardedVideo", options);
23.         }
24.     }
25.     private void HandleShowResult(ShowResult result)
26.     {
27.         switch (result)
28.         {
29.         case ShowResult.Finished:
30.             Debug.Log ("The ad was successfully shown.");
31.             cts.AllowedToContinue = false;
32.             cts.ContinueTimer = 300;
33.             chng.Revive ();
34.             break;
35.         case ShowResult.Skipped:
36.             Debug.Log("The ad was skipped before reaching the end.");
37.             break;
38.         case ShowResult.Failed:
39.             Debug.LogError("The ad failed to be shown.");
40.             break;
41.         }
42.     }
43.
44. }
```

Abbildung 29 Programmcode im Game Master für die Werbung

In der `Start()` Funktion wird die Werbung initialisiert, sprich heruntergeladen. Wichtig dabei ist der erste Parameter in `Advertisement.Initialize()`. Der String"Game ID" ist dabei eine ID die von Unity deinem Spiel zugeordnet wird. Sie wird bei der Verlinkung des Apps mit dem Unity Ads Service festgelegt, bei einer falschen Zuordung wird der Umsatz nicht vergütet.
In Zeile 22 und 14 ruft die Funktion `Advertisement.Show ()` die Werbung im Spiel auf. Dabei wird im verlangten String Parameter zwischen einer Interstitial Ad und einer Rewarded Ad unterschieden. Bei einer Rewarded Ad kann dann im Callbackhandler, bei einem gelungen aufschalten der Werbung, eine Belohnung einprogrammiert werden.
Die Funktionen `ShowAd(…)` und `ShowRewardedAd(…)` sind dabei **public**, also öffentlich, und können beliebig oft im Spiel aufgerufen werden, solange das Script in der Szene enthalten ist.

Im Spiel wird die Interstitial Ad jeder zehnte Spielversuch aufgerufen, um den Spielfluss nicht stark zu behindern. Die Rewarded Ads werden im Spiel alle 5 min angeboten. Dafür wird dem Spieler die Möglichkeit angeboten mit dem letzten Spielstand weiter zu spielen.

IN-GAME-PRODUKTE

Die In-App oder In-Game Produkte sind Produkte die vom Entwickler im App oder Play Store erstellt werden und vom Spieler für Geld erworben werden können. Da die Produkte zuerst in den einzelnen Stores erstellt werden müssen, muss die Programmierung auch vorerst aufgeschoben werden. Im folgenden Kapitel wird deshalb zuerst das Erstellen der Produkte im App Store und Play Store aufgezeigt.

DAS ERSTELLEN VON IN-APP-PRODUKTEN

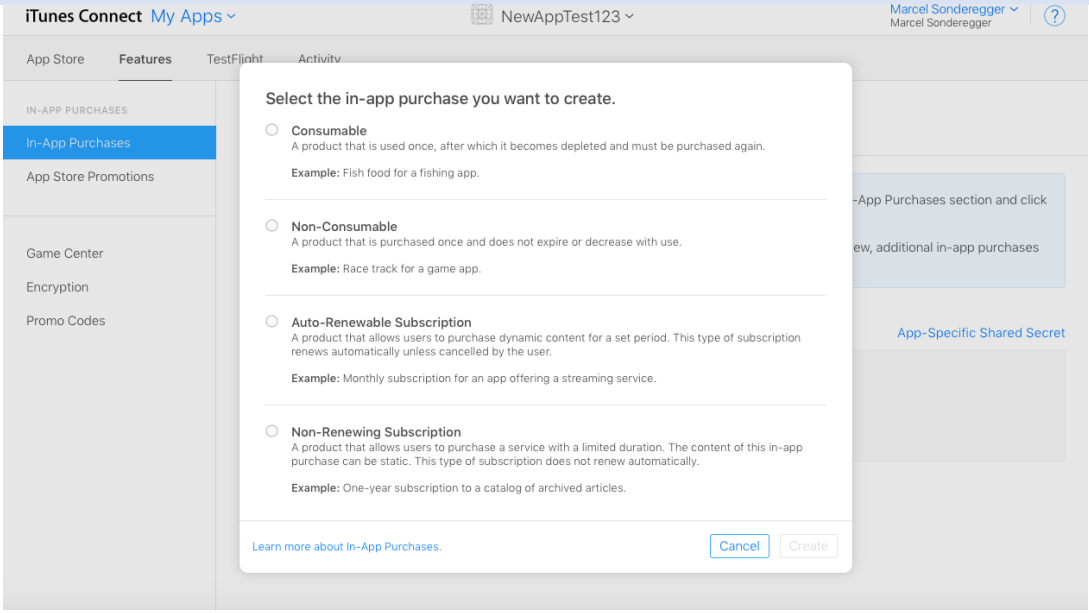


Abbildung 30 Der Bildschirm in ITunes-Connect vor dem Erstellen eines digitalen Produktes.

Wird in ITunes Connect unter Features >> In-App Purchases ein neues Produkt erstellt, muss zuerst der Produkttyp ausgewählt werden. Da im Spiel **0123** die Werbung für einen gewissen Betrag komplett ausgeschaltet wird, ist dieses Produkt ein Beispiel für ein **Non-Consumable Produkt**. Hat manden Typ ausgewählt müssen weitere periphere Faktoren wie Beschreibungen oder den Namen des Produktes augewählt werden (Abbildung 30). Wichtig für die Programmierung ist die „**Product ID**". Die Product ID ist ein einzigartiger alphanumerischer String welcher das Produkt auszeichnet. Im Fall des Produktes von 0123 wird **com.gsstudios.removeads0123** angeben.

Der Preis kann per Dropdownmenu ausgewählt werden und wird hier auf CHF 1.- gesetzt.

Analog müssen diese Schritte auf der Google Play Console unternommen werden.
Unter In-App-Produkte kann oben rechts „Verwaltetes Produkt erstellen" ausgewählt werden. (Abbildung 31)

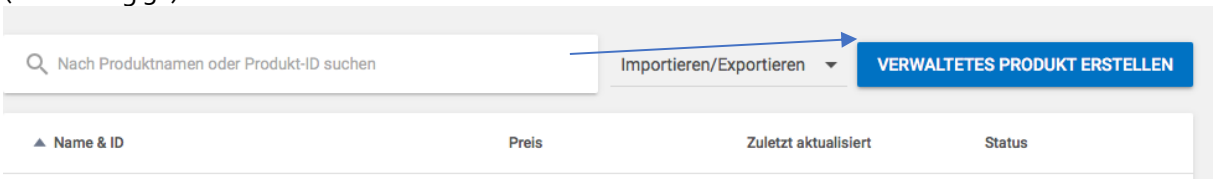


Abbildung 31: Erstellen eines Produktes im Play Store

Auch hier wird eine Produkt ID und ein Name gewählt. Um später das Programm nicht umschreiben zu müssen wird hier die gleiche Product ID eingefügt. Des Weiteren können auch hier periphere Faktoren wie Beschreibungen oder Screenshots eingefüllt werden. Der Preis kann im Play Store variable definiert werden. Es wird auch CHF 1.- als Preis eingegeben.

Im Programm wird das Asset **Unity-IAP** verwendet. Hierfür wird ein Script namens **Purchaser** aus der Bibliothek verwendet welches noch angepasst werden muss.
Im Purchaser Script wird die Product ID als String abgespeichert.

```
1.  public static string RemoveAdsID = "com.gsstudios.removeads0123";
```

in der Funktion `InitializePurchasing()`wird das Produkt zusammen mit der ID als Objekt erstellt. (Zeile 12). Dabei initialisiert die Funktion die Produkte für den Kauf.

```
1.          public void InitializePurchasing()
2.          {
3.              if (IsInitialized())
4.              {
5.                  return;
6.              }
7.
8.              var builder = ConfigurationBuilder.Instance(StandardPurchasing
9.  Module.Instance());
10.
11.
12.             builder.AddProduct(RemoveAdsID, ProductType.NonConsumable);
13.             UnityPurchasing.Initialize(this, builder);
14.         }
```

Abbildung 32 Code-Ausschnitt des Purchaser Script

In mehreren Methoden wird das Objekt für die Anfrage vorbereitet und bearbeitet. Was genau mit dem Objekt geschieht ist für das Spiel nicht weiter relevant.

Mit der Funktion `BuyProductID(String)` wird die Anfrage erstellt um ein Produkt zu kaufen. Verpackt wird diese Funktion in einer öffentlichen Funktion `BuyRemoveAds()` damit man vom Spiel aus darauf Zugriff hat.

```
1.  public void BuyRemoveAds()
2.          {
3.              BuyProductID(RemoveAdsID);
4.          }
5.
6.          void BuyProductID(string productId)
7.          {  //Weiterer Code…
```

Abbildung 33 die öffentliche Funktion BuyRemoveAds() für das Spiel.

(Apple, 2017) (Langley, 2015)

PROGRAMMIERUNG DES GAMECENTERS UND DER BESTENLISTEN

Nach den In-App-Produkten kommt die Integration der Bestenliste. Es wird GameCenter[13] für Apple und PlayConsole[14] für Android benutzt. Dabei wird man sich den integrierten Bestenlisten widmen.

Die Bestenliste wird in ITunes Connect unter **Features >> Leaderboards** erstellt werden.

Auch hier wird neben einem Referenznamen eine **Bestenlisten ID** verlangt (Abbildung 34).
Diese ist wieder ein eindeutiger alphanumerischer String, der als Referenz für die Bestenliste im Appstore verwendet wird.
Angegeben wird für 0123:
com.0123leaderboard.GSStudios.

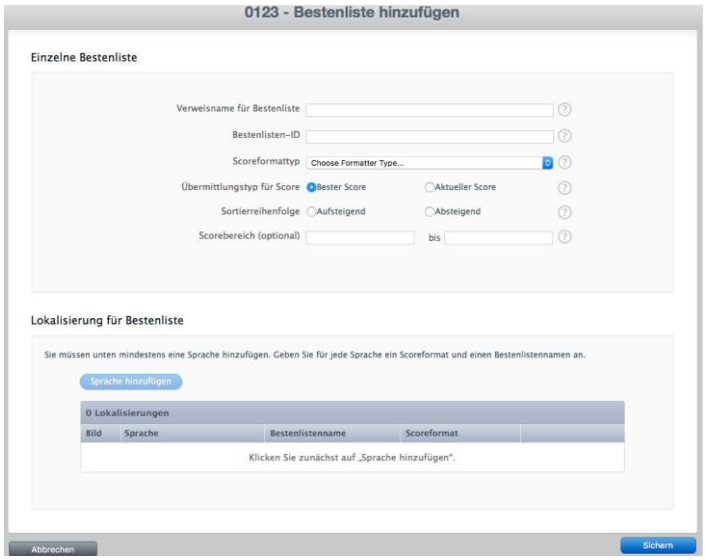


Abbildung 34 : Ausschnitt aus ITunes-Connect für die Erstellung der Bestenlisten

Das Score-Format kann mit einem Dropdownmenu ausgewählt werden. Da die Punkte im Spiel keine Dezimalzahlen, sondern ganze Zahlen sind, wird der Typ „Integer" ausgewählt. Die weiteren Parameter sind für dieses Projekt nicht relevant

In Unity muss dieser Score jetzt dieser Bestenliste übermittelt werden. In der Funktion `ReportScore()` wird der Score in Zeile 4 übermittelt.
In Zeile 2 wird der Score in ein **long** [15]Wert umgewandelt da die Funktion `Social.ReportScore()` solch einen Paramtertyp verlangt.

```
1.  public void ReportScore(){
2.          long score = System.Convert.ToInt64(PlayerPrefs.GetInt("Highscore")
    );
3.
4.          Social.ReportScore(score, "com.bestscore0123.GSStudios", HighscoreC
    heck);
5.  // Report Score to IOS Leaderboard
6.      }
```

Abbildung 35 die Report Score Funktion übermittelt den Score an die Bestenliste

Aufgerufen wird die Funktion `ReportScore()` jedes Mal wenn ein neuer Highscore erreicht wird.

Das Spiel ist nun für iOS fertig programmiert und könnte exportiert und veröffentlicht werden. Die Werbung oder die digitalen Produkte würden auf Android jedoch nicht funktionieren. Der Punkt ist angekommen indem eine zweite Version des Projektes erstellt werden muss, um das Spiel für beide Geräte kompatibel zu machen.

[13] Game Center ist das von Apple Angebotene Social-Gaming-Netzwerk.
[14] Play Console ist das von Google Angebotene Social-Gaming-Netzwerk
[15] long ist ein Integer mit 64 Bit

ANDROID ANPASSUNGEN

Anders als bei Apple können einer Bestenliste im Play Store nicht einfach Daten zugeschickt werden. Der Store muss das Spiel identifizieren können, damit er dessen Daten auch in die Bestenliste integriert. Für den Play Store müssen wir also eine neue Library von Github [16] importieren namens „**play-games-plugin-for-unity**."
Die Library ermöglicht die Integration einer Bestenliste für das Unity Projekt. Vorerst müssen aber einige Schritte in der Google Play Console unternommen werden.
Unter „**Bestenlisten**" wird eine neue Bestenliste erstellt. Die Parameter sind gleich wie bei Apple, bis auf die Leaderboard-ID, welche automatisch erstellt wird. Als Referenznamen für die Bestenliste wird „**Highscore 0123**" angegeben. Weitere Bestenlisten können beliebig hinzugefügt werden.
Unter den Bestenlisten ist ein Link mit der Beschriftung „**Ressourcen Abrufen**". Dieser Link ruft ein Fenster auf (Abbildung 36). Der XML-Code wird in die Zwischenablage kopiert.

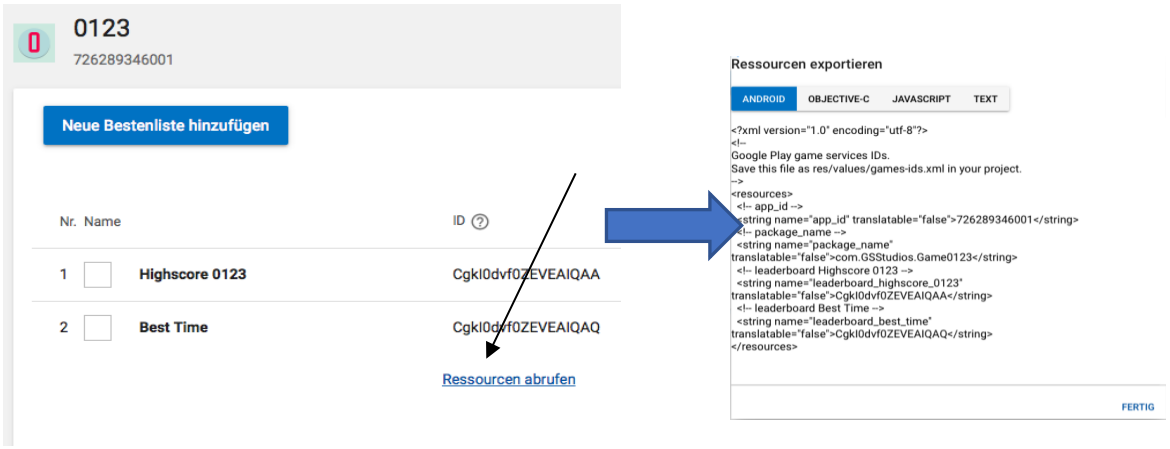


Abbildung 36 zeigt zwei Bestenliste und die Texteinheit die in die Zwischenlage kopiert werden muss

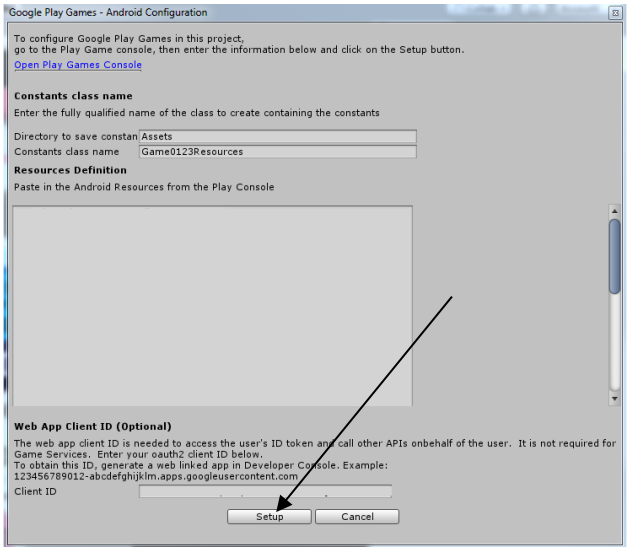


Abbildung 37 das Eingabefeld für das Android Setup in Unity

Zurück in Unity müssen diese Parameter eingesetzt werden. Öffnet man ein Fenster unter **Window>>GooglePlaySetup>>Setup>> Android Setup** sieht man ein grösseres Eingabefeld (Abbildung 37). Hier müssen die Ressourcen wieder hineinkopiert werden. Klickt man den Knopf **Setup** wird automatisch ein C# File erstellt, womit die Library auf die Bestenlisten zugreifen kann. Für die In-Game-Produkte müssen keine weiteren Änderungen vorgenommen werden, da die gewählte Produkt ID für den PlayStore und denn AppStore identisch ist.

[16] GitHub ist eine online Ablage und Versionsverwaltungsystem, welche viele OpenSource Bibliotheken enthält.

DEBUGGING UND TESTING

Im nächsten Schritt wird das Spiel als Datei exportiert und anschliessend veröffentlicht. Dabei wird die fertige Datei jedoch nicht sofort auf den Stores veröffentlicht, sondern zuerst ein paar Freunden und Verwandten zum Testen übergeben.
Dieser Prozess bezeichnet man als Debugging und Testing. Das Spiel funktioniert im Editor auf dem Computer, jedoch weiss man nicht ob allfällige Fehler auf den Smartphones entstehen können. Aus diesem Grund muss die Software zuerst auf verschiedene Geräte exportiert und getestet werden.

Das Spiel wurde gleichmässig an Android Nutzern und iOS Nutzern verteilt. Hier noch ein paar Nennenswerte Fehler:

- Die Transaktionen der In-App Produkte konnten nicht durchgeführt werden. Grund war das Fehlen eines Händler Konto für den App-Store und Play-Store.
- Der Bestenlisten wurden keine Werte übergeben, der Grund war, dass kein Integer mit 64 Bit übergeben wurde. Der Editor hatte mir jedoch keinen Fehler ausgegeben.
-

DIE VERÖFFENTLICHUNG

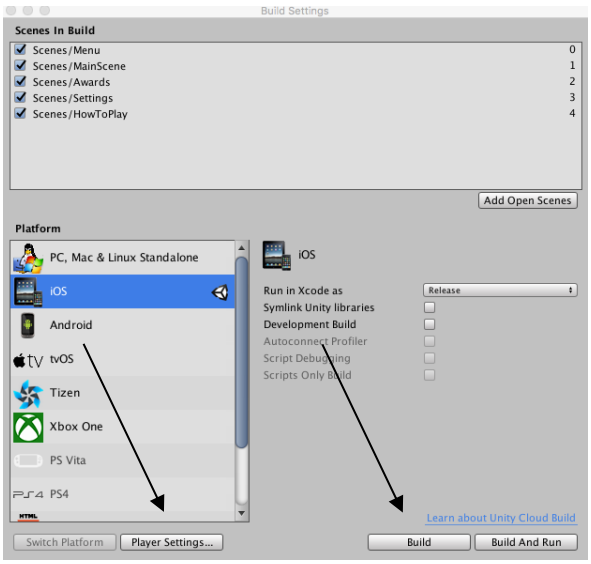


Abbildung 38 Das Build Settings Fenster in Unity

Das Spiel ist im Editor fertig und wurde getestet, so kann es jetzt exportiert werden.

Unter **File >> Build Settings** kann die gewünschte Plattform ausgewählt werden. Man wird auch hier für beide Plattformen individuelle Anpassungen machen müssen.

IOS

Vor dem Export muss unter **Player Settings>>Icon** ein Bild der Grösse 1024x1024 Pixel eingefügt werden. Dieses Bild wird von Unity automatisch für alle Geräte konvertiert. Klickt man nun auf **Build**, wird ein **Xcode File** erstellt. Öffnet man das zugehörige File in XCode kann das Spiel unter **Product>>Archive** fertiggestellt werden und in eine Datei des Formates IPA zusammengestellt werden. XCode öffnet dann ein Fenster mit einem Knopf „Upload to App Store" (Abbildung 39). Das IPA File wird auf der ITunes-Connect Seite hochgeladen werden. In ITunes Connect können nun alle zusätzlichen Daten für die Anzeige im Store, sowie das gerade Hochgeladene File ausgewählt und editiert werden. Hat man eine Beschreibung sowie die Screenshots hochgeladen, kann das Spiel oben rechts mit dem Knopf „Submit for Review" zur Überprüfung eingereicht werden. Nach ein paar Tagen sollte das Spiel überprüft sein und im Appstore zum Download bereitstehen.

Abbildung 39 : Das Fenster vor dem Upload zu ITunes-Connect

ANDROID

Im Fenster von Abbildung 38 muss Android als Plattform ausgewählt werden. Klickt man nun auf Player Settings hat man neben dem Icon einen weiteren Faktor zu beachten. Für Android muss ein Schlüsselfile erstellt werden. Das Schlüsselfile ist ein Sicherheitsfaktor damit das Projekt nicht gestohlen werden kann. Gibt man ein Passwort ein, kann durch den Knopf „Browse Keystore" ein solches Schlüsselfile generiert werden(Abbildung 40). Dieses File muss dann unter **Key** wieder ausgewählt werden. Wichtig zu beachten ist dass spätere Versionen des Spieles das gleiche Schlüsselfile besitzen müssen. Ansonsten wird die Version von Google nicht akzeptiert. Klickt man nun wieder in den Build Settings (Abbildung 38) auf **Build,** wir ein **APK**[17] File erstellt.

Abbildung 40 zeigt die Verschlüsslung des Projektes

Das File kann nun auf der Google Play Console hochgeladen werden. Unter App Versionen kann wie in (Abbildung 41) das APK hochgeladen werden.

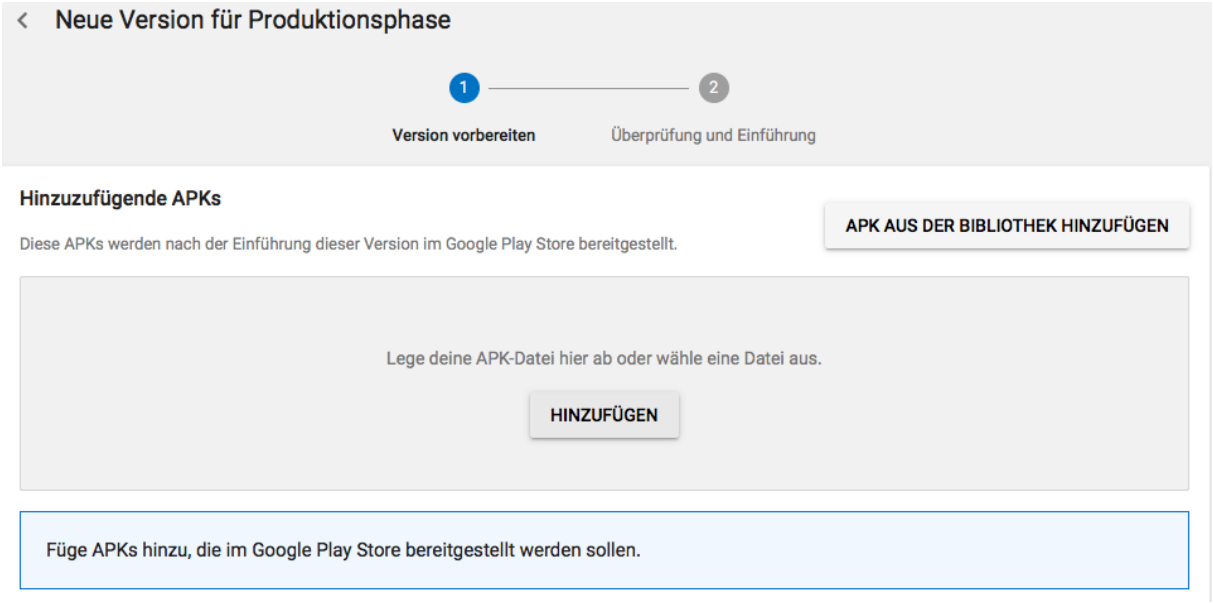


Abbildung 41 : Das letzte Fenster auf der Google Play Console vor der Veröffentlichung

Auch hier müssen dann noch die einzelnen Schritte wie das Einfügen einer Beschreibung und Screenshots durchgeführt werden. Wenn alle Anforderungen erfüllt sind kann das Spiel auch mit einem Knopfdruck veröffentlicht werden. Anders wie bei Apple wird das Spiel in ein paar Stunden online sein und zum Download bereitstehen. Das App wird dafür auch nicht geprüft, es ist also durchaus möglich eine App hochzuladen welche gar nicht funktioniert.

[17] Ein APK File ist das Pendant zum IPA File des App Stores.

MARKETING

Das Spiel „**0123**" wurde am 8. September veröffentlicht. Wie schon in der Arbeit erwähnt, ist es auf den zwei grössten App-Stores erhältlich. Ein wesentlicher Nachteil ist, dass auf diesen beiden Stores **pro Tag mehrere tausende Apps** hochgeladen werden. Zu hoffen, dass das Spiel sich von alleine verbreitet ist deshalb unrealistisch. Hauptsächlich wird über **Facebook** und **Whatsapp** versucht die App zu verbreiten. Weitere Möglichkeiten benötigen oft mehr Kapital

Facebook bietet die Möglichkeit sich in „Gruppen" mit Personen gewisser Interessen zu verbinden. Diese Gruppen beinhalten meist ein paar tausend Personen, können aber auch deutlich grösser oder kleiner sein. Der Nachteil ist, dass diese Art der Verbreitung sehr unpersönlich ist und viele dieser Gruppenmitglieder auch versuchen ihr eigenes Spiel zu verbreiten. Für dieses Projekt bin ich um die 50 Gruppen mit durchschnittlich 1000-2000 Mitgliedern beigetreten. r

Über Whatsapp habe ich die Links der App an möglichst alle meine Freunde und Bekannten gesendet.

RESULTATE

Das Spiel wurde in den letzten Schritten veröffentlicht. Ob ich ein Millionär sein werde wird sich nun zeigen.

DOWNLOADS

Die Statistiken auf Abbildung 42 und Abbildung 43von Apple und Google bereitgestellt. Die Graphiken zeigen einen Graphen der Downloads im Zeitraum vom 8. September - 8. November 2017.

APP STORE:

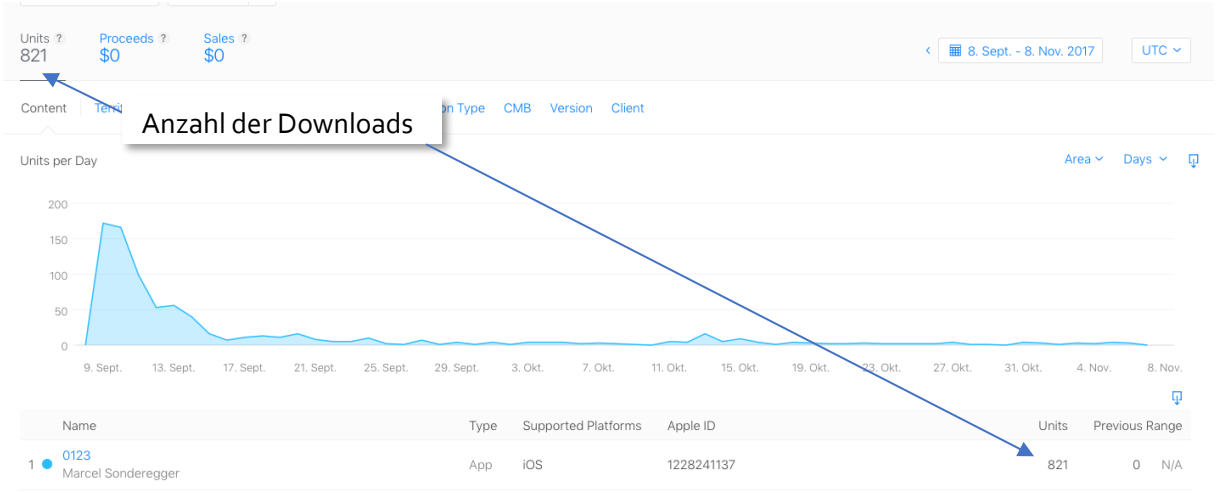


Abbildung 42 Downloads auf iOS Plattformen

Vom 8. September bis zum 8. November wurde die App **821 Mal** auf dem App Store heruntergeladen. Das globale Maximum der Statistik befindet sich gleich nach der Veröffentlichung. Die Downloads nehmen nach September stark ab und haben keine grösseren Peaks.

PLAY STORE:

Abbildung 43

Das App wurde auf dem Play Store **187 Mal** heruntergeladen. Das globale Maximum ist dabei Mitte Oktober und weitere lokale Maxima sind deutlich im September und Anfangs Oktober zu erkennen. Die Anzahl der Downloads im November sind dabei sehr gering.

WERBUNG

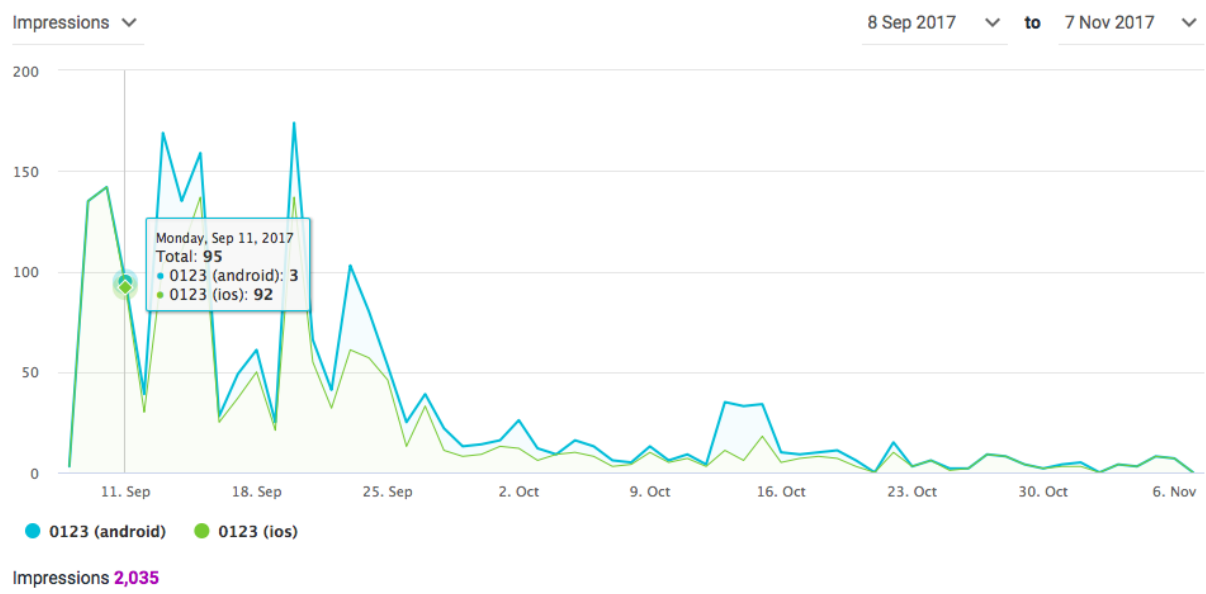


Abbildung 44

Diese Graphik zeigt den Verlauf der **Impressions**[18]. Man erkennt eine Ähnlichkeit im Verlauf der Graphen zum dem des Umsatzes (Abbildung 45). Dabei ist zu beachten, dass die Werte der

[18] Impressions sind die Anzahl der aufgeschalteten Werbungen

Vertikalen Achse nur für iOS korrekt dargestellt sind. Die Graphen überlagern sich, obwohl die Werte nicht gleich sind. Wie in der Abbildung 44 gezeigt, wurden zum Beispiel am 11. September 95 Werbungen im Total aufgeschaltet. 92 davon waren jedoch von iOS und nur 3 von Android.

Mit der Werbung wurde innerhalb von 2 Monaten ein Umsatz von 14.54 USD erzielt.

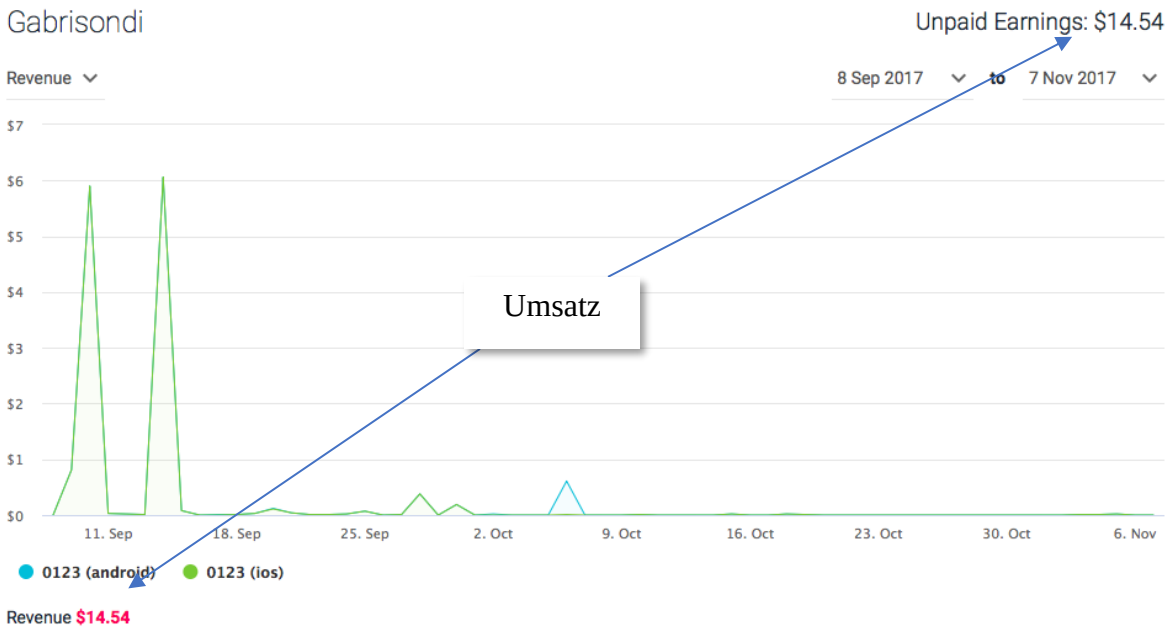


Abbildung 45 zeigt den Graphen des Umsatzes welcher mit der Werbung erzielt wurde

Die höchsten Tagesumsätze wurden dabei am 11. Und 14 September erreicht.
Des Weiteren wurde ein Umsatz von 2 CHF durch In-App Käufe auf dem Play Store erzielt. Die digitalen Produkte wurden also zwei Mal erfolgreich gekauft. Auf den iOS Geräten wurde dabei kein Produkt gekauft.

DISKUSSION DER RESULTATE

Im Total wurde das Spiel auf beiden Plattformen **1008** Mal heruntergeladen und ein Umsatz von **USD 14.54 + CHF 2.00** erzielt.

Das Projekt funktioniert auf beiden Betriebssysteme und kann fliessend gespielt werden. Das Spiel wurde dabei trotz geringem Marketing über tausend Mal heruntergeladen, so dass ein Umsatz von mehr als CHF 15.- erzielt werden konnte. Das Spiel wurde vorwiegend auf iOS heruntergeladen . Das Marketing auf Facebook zeigte nur geringe Wirkung für iOS, erzielte aber ein Downloadmaximum bei Android. Die Android Nutzer sind dabei auch die einzigen, die Geld für ein digitales Produkt ausgegeben hatten.

RÜCKBLICK

Das Ziel des Projektes war es ein funktionstüchtiges Spiel zu erstellen, welches auf dem App Store und auf dem Play Store erhältlich ist. Dabei sollte mit digitalen Produkten und Werbung Umsatz erzielt werden. Durch plattformübergreifende Entwicklung konnte das Projekt auch einfach realisiert werden. Des Weiteren wurde über Social Media das Spiel verbreitet.
Das wesentliche Ziel dieser Arbeit wurde also erreicht.

AUSBLICK

VERSION CONTROL

In weiteren Projekten werde ich ein **Version Control,** also ein Versionskontrol-System einführen, da das Projekt schnell unübersichtlich wurde, als auf zwei Computer gearbeitet wurde. Ein Versionskontrol-System erlaubt es verschiedene Versionen des Projektes in einer Ablage zu Speichern. Vorteile dieses Systems ist es, dass alle Änderungen am Projekt oder an einer Version übersichtlich dokumentiert sind. Ältere Versionen können auch schnell wiederhergestellt werden. So würde für dieses Projekt eine Version für iOS und eine Version für Android erstellt werden. Wird ein Fehler auf einer Plattform behoben, ist dies in der Versionskontrolle dokumentiert und kann einfach übertragen werden. So muss nicht jeder Fehler sofort in der zweiten Version korrigiert werden und der Arbeitsfluss wird nicht behindert.

MARKETING

Das Marketing kann noch stark verbessert werden. Weitere Optionen erfordern jedoch meist ein Kapital. Des Weiteren waren die angesprochene Personengruppe auf Facebook oft selbst Entwickler und wollten auch nur ihr Spiel verkaufen. Trotzdem hat Facebook hat eine breite Bandweite an Personen bereitgestellt, jedoch wurden andere Plattformen wie Twitter oder Instagram kaum genutzt. Man könnte des Weiteren:

- Durch Blogs oder einer Website eine Community für das Spiel gewinnen.
- Diverse App-Review Websiten über Email anfragen.
- Ein simpler Trailer auf YouTube erstellen. Dies würde einen Einblick in das Spiel geben.

Ich werde versuchen möglichst viele der erwähnten Punkte in mein nächstes Projekt zu integrieren.

LITERATURVERZEICHNIS

Anon., 2016. *https://glossar.hs-augsburg.de/.* [Online]
Available at: https://glossar.hs-augsburg.de/Versionsverwaltung
[Zugriff am 19 Oktober 2017].

Anon., 2017. *42matters.* [Online]
Available at: https://42matters.com/stats
[Zugriff am 2 September 2017].

Apple, 2015. *developer.apple.com.* [Online]
Available at:
https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/AppID.html
[Zugriff am 3 September 2017].

Apple, 2016. *developer.apple.com.* [Online]
Available at:
https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide/UsingiTunesConnect/UsingiTunesConnect.html
[Zugriff am 2 September 2017].

Apple, 2017. *Apple Developer.* [Online]
Available at:
https://developer.apple.com/library/content/documentation/LanguagesUtilities/Conceptual/iTunesConnectInAppPurchase_Guide/Chapters/CreatingInAppPurchaseProducts.html
[Zugriff am 4 September 2017].

Apple, 2017. *Apple Developer.* [Online]
Available at:
https://developer.apple.com/library/content/documentation/LanguagesUtilities/Conceptual/iTunesConnectInAppPurchase_Guide/Chapters/CreatingInAppPurchaseProducts.html

hs-augsburg, 2014. *glossar.hs-augsburg.de.* [Online]
Available at: https://glossar.hs-augsburg.de/Objektorientierte_Programmierung
[Zugriff am 17 10 2017].

kiip, 2014. *blog.kiip.me.* [Online]
Available at: http://blog.kiip.me/developers/mobile-ad-options/
[Zugriff am 10 oktober 2017].

Langley, K., 2015. *Learning Unity IOS Game Development.* UK(Birmingham): Packt Publishing.
Statista, 2017. *www.statista.com.* [Online]
Available at: https://www.statista.com/topics/1729/app-stores/
[Zugriff am 2 September 2017].

Unity, kein Datum *unity3d.com.* [Online]
Available at: https://unity3d.com/de/learn/tutorials/topics/ads-analytics/integrating-unity-iap-your-game
[Zugriff am 12 10 2017].

Wikipedia, 2017 . *Wikipedia.* [Online]
Available at: https://de.wikipedia.org/wiki/Adobe_Illustrator

Wikipedia, 2017. *Wikipedia.* [Online]
Available at: https://de.wikipedia.org/wiki/Plattformunabh%C3%A4ngigkeit