Spieleprogrammierung in Unity

Vertiefungsarbeit zum Modul 154

Kevin Thalmann und Ursin Zimmermann

M154

Inhaltsverzeichnis

[2 Einleitung 2](#_Toc434831538)

[3 Bezug zur Einführung 2](#_Toc434831539)

[4 Projektvorgaben 2](#_Toc434831540)

[5 Informieren 3](#_Toc434831541)

[5.1 Unity 3](#_Toc434831542)

[5.1.1 Grundlegende Informationen 3](#_Toc434831543)

[5.1.2 Unity Ads 3](#_Toc434831544)

[5.1.3 Unity Analytics 3](#_Toc434831545)

[5.1.4 Asset Store 3](#_Toc434831546)

[5.1.5 Aufbau von Unity 4](#_Toc434831547)

[5.2 Visual Studio 5](#_Toc434831548)

[5.3 Versionierung 5](#_Toc434831549)

[6 Planung 6](#_Toc434831550)

# Einleitung

In dieser Vertiefungsarbeit werden wir mit Unity ein Spiel programmieren. Wir untersuchen dabei die Features und Services von Unity und werden dann eine Spiel-Idee ausarbeiten. Danach werden wir uns Gedanken über den Aufbau des Spieles aus Sicht der Programmierung machen.  
Bei der Implementation werden wir TDD verwenden und gleichzeitig Unit Tests in Unity ausprobieren. Am Schluss werden wir das Spiel Auf dem PC und / oder Handy veröffentlichen.   
Vorkenntnisse haben wir beide in C# und TDD jedoch keine in Unity.

# Bezug zur Einführung

In Unity werden wir vor allem die Implementation anschauen. Denn das einige was sich verändert ist die Entwicklungsumgebung. Das Design wird sich ein bisschen verändern, aber nicht viel da Unity auch Objekt Orientiert ist, jedoch wird alles auf Script aufgebaut, die dann zurechtgezogen werden.  
  
Wir werden auch TDD verwenden, was in Unity dank vorhandenem Framework einfach umzusetzen ist.

# Projektvorgaben

**Entwicklungsumgebung**: Unity, Visual Studio  
**Versionierung**: Github  
**Vorgehensweise**: TDD, IPERKA  
**Sprache**: COmmits und Code in Englisch, Dokumentation n Deutsch.

# Informieren

## Unity

### Grundlegende Informationen

Auf der offiziellen Website von Unity werden 2 Versionen angeboten:

* Eine Kostenlose „Personal Edition“, welche weniger Features anbietet und bei jedem Start vom Spiel das Unity Logo anzeigt. Die Einschränkung hierbei ist, dass man ab 100‘000 Fr. Umsatz pro Jahr die „Professional Edition“ kaufen muss.
* Für 75$/Jahr oder einmalig 1500$ wird die „Professional Edition“ angeboten. Man erhält zusätzliche Features wie Cloud Build, oder besseren Support. Auch während der Programmierung erhält man mehr Features.

Links zum Verglich: <https://unity3d.com/get-unity>

Wir haben uns für Unity „Personal Edition entschieden“, da wir alle Features haben die wir brauchen und Ads und Analytics für beide Versionen zur Verfügung steht.

### Unity Ads

Unity Ads werden als kostenloser Service angeboten, bei dem man Werbung in sein Spiel schalten kann (nur bei Export zu Android und Apple) und damit Geld verdienen:  
<https://unity3d.com/services/ads>

### Unity Analytics

Analytics werden von Unity auch kostenlos angeboten und können mit der „Personal Edition verwendet werden. Analytics erlauben es das Verhalten des Spielers und vieles mehr zu Analysieren und mit Grafiken darzustellen:  
<https://unity3d.com/services/analytics>

### Asset Store

Unity bietet einen Asset Store an auf dem man verschiede Figuren, Scripts und weiter herunterladen kann. Jedes Asset im Store unterliegt standardmässig den Unity Standard Lizenzen und können im Projekt verwendet und verändert werden, auch kommerziell. Die Assets werden von Unity selber oder anderen Community Mitgliedern erstellt:  
<https://unity3d.com/services/analytics>

### Aufbau von Unity

## Visual Studio

Als IDE haben wir Visual Studio verwendet. Wir haben beide unsere Lizenz von DreamSpark genutzt. Unity hat seit der letzten Version den Support für Visual Studio verstärkt, was es vereinfacht Unity Code zu debuggen und in Unity einzubinden.

Unity und Visual Studio Support: <http://docs.unity3d.com/Manual/VisualStudioIntegration.html>

## Versionierung

Zur Versionierung haben wir GitHub verwendet. Das Projekt ist dann zwar Open Source, aber wir können auch dann nach dem Projekt immer noch daran Arbeiten und müssen keinen Server haben um Git zu verwenden.  
 Wir haben uns geeinigt die Commits möglichst klein zu halten und nach Jedem Feature einen Commit zu machen. Wir machen dabei keine Branches für Features, das das Projekt zu klein ist.

Link zum Branch: <https://github.com/Saphatonic/M154>

# Planung

## Game Design