\chapter{Plattformübergreifende Frameworks zur Spieleentwicklung}

\section{Gamespezifische Frameworks und Engines}

\subsection{Cocos2D-X}

\subsection{Libgdx}

\subsection{Unity3D}

\subsection{Weitere Frameworks}

\section{Entwicklungsumgebungen}

\subsection{Unterstützte IDEs}

\subsection{Systembedingte Einschränkungen}

%TODO: Kapitel 6

\chapter{Gegenüberstellung der Frameworks}

%Features und Einschränkungen

%\section{Skalierbarkeit der Menge der Plattformen} % ???

\section{Programmiersprachen}

libGDX nutzt einzig und allein Java für die Entwicklung, was für reine Android Projekte ein großer Vorteil ist. Denn dadurch müsste man bei der Kompilierung kaum Kompromisse eingehen.

\medskip

Mit Coco2D-X hat man die Wahl zwischen C++, Lua und JavaScript, welche auf allen unterstützten Plattformen funktionieren. Einzige Einschränkung liegt bei JavaScript, da diese sich bei Cocos2D-X nicht mit Windows Phone verknüpfen lässt.

%QUELLE: http://www.cocos2d-x.org/wiki/Cocos2d-x

Lua ist eine schnelle, von der Syntax simpel gehaltene Skriptsprache, mit objektorientierten Eigenschaften. Die geschriebenen Skripte werden durch einen Interpreter in Bytecode übersetzt und ist kompatibel mit der Sprache C. Lua befindet sich derzeit in Version 5.3 und wird häufig in der Spieleentwicklung eingesetzt.

%QUELLE: <http://www.lua.org/about.html>

Mit der Unterstützung von C++, welches direkt von C abgeleitet ist, hat man die Möglichkeit äußerst perfomante und portable Software zu schreiben. Voraussetzung ist dabei ein sicherer Umgang, da C++ im Vergleich mit anderen High-Level Sprachen, sich weniger um Logikfehler kümmert und entsprechend auch keine Warnungen ausgibt.

Für Entwickler die zum Beispiel aus dem Webbereich kommen, bietet JavaScript einen weiteren Einstieg in Cocos2D-X.

\medskip

In Unity3D finden die Sprachen C\#, UnityScript und Boo Verwendung. Alle für Unity3D nutzbaren Sprachen entspringen Microsofts .NET Framework. Das bedeutet auch, dass es möglich ist weitere .NET Sprachen zu benutzen, wenn diese ihre Skripte in das DLL (Dynamically Linked Library) Format kompilieren können. Diese DLL Dateien können dann dem Unity Projekt hinzugefügt und verwendet werden.

%QUELLE: http://docs.unity3d.com/Manual/UsingDLL.html

\medskip

Boo ist eine von Python beeinflusste Sprache für .NET und Mono, welche ohne Klammern und Semikolon auskommt. Die Typisierung ist generell statisch, kann aber trotzdem dynamische „Duck-Typing“ Eigenschaften nutzen. Der Name Duck-Typing entsprang dem sogenannten „Ententest“ zur Typisierung, welcher wiederum auf einem Gedicht von James Whitcomb Riley zurückzuführen ist.

“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.”

Beim Duck-Typing werden Objekte nicht durch ihre Klasse typisiert, sondern durch die vorhandenen Attribute und Methoden. Daher findet die Typisierung erst zur Laufzeit durch den Interpreter statt.

%QUELLE: <http://boo-language.github.io/>

Oft wird UnityScript in der Unity Community, aber auch von dem Unternehmen selbst, fälschlicherweise mit JavaScript gleichgesetzt, als wären die beiden Sprachen äquivalent. Auch wenn syntaktische Ähnlichkeiten bestehen, gibt es große semantische Unterschiede.

UnityScript kann, wie die meisten objektorientierten Programmiersprachen, Klassen definieren und daraus Objekte erstellen. JavaScript hingegen besitzt zwar ebenfalls objektorientierte Eigenschaften, wobei diese aber sogenannte „Prototypes“ verwendet. Dies sind Objekte die zuerst von einer Funktion definiert werden und danach mit dem „new“ Schlüsselwort instanziiert werden. Nach der Instanziierung ist es möglich, das Objekt um zusätzliche Eigenschaften zu erweitern. Des Weiteren kann UnityScript im Gegensatz zu JavaScript, Klassen, Objekte, Funktionen und Variablen mit Zugriffsmodifikatoren / Sichtbarkeiten versehen. UnityScript wurde speziell für die Unity Engine konzipiert und ist proprietär, was es schwierig macht, genaue Spezifikationen zu finden.

%QUELLE: <http://wiki.unity3d.com/index.php/UnityScript_versus_JavaScript>

C\#

\section{Benutzerinteraktion und Sensoren}%Accelerometer etc

Smartphones und Tablets besitzen in der Regel kaum oder gar keine Hardwarebuttons, die für Anwendungen neu belegbar sind. Spiele die für solche mobilen Systeme entworfen worden, greifen daher auf eine oder zwei der beiden populärsten Steuerungsmechanismen zurück.

Überwiegende Praxis ist es, den vorhandenen Touchscreen für Eingaben und Interaktionen zu nutzen. Dies kann über bekannte Gesten geschehen oder über die Virtualisierung von Aktionsflächen, wie zum Beispiel Buttons und Regler.

Eine andere Möglichkeit in das Spielgeschehen einzugreifen, bieten die geräteseitigen Hardwaresensoren. Der Gyroskop Sensor kann somit beispielsweise für ein Lenkrad genutzt werden, womit bei einem Autorennen durch kippen des Geräts gesteuert wird.

\section{Free- und Pro- Versionen}

\section{Einfluss auf Einstellungen}

\section{Zusätzlich benötigte Software}

\section{Aktualität - Versionen - Community}

\section{Zukunftsaussichten}