ulm university universität ulm

Ulm University | 89069 Ulm | Germany

Faculty of Engineering and Computer Science

institute of media informatics

Proposal for Master Thesis: Development of a Platform for Serious Games with Procedurally Generated Content

Project-Members and authors:
Michael Legner
800817
michael.legner@uni-ulm.de

Reviewer:
Prof. Dr.-Ing. Michael Weber
Prof. Dr. Helmuth A. Partsch

Supervisor:
Julian Frommel, Julia Greim

Year:
2014

Print: PDF-LaTeX 2_ε

Abstract

In this masters thesis, a platform for developing serious games will be developed. This proposal outlines the subject and how it will be addressed.
Serious games are distinguished from entertainment games as their primary concern is to teach. Form a technical standpoint the requirements are not different, therefore the first step in this thesis is to evaluate game engines to find a suitable one for future usage in a research project that has started at Ulm University. Key points are ease of learning, supported platform, future support by the developer and an active community. There is already a huge number of engines available, how they will be reduced to only three candidates which will be evaluated in detail is described in section 3. The engine will be extended by a module to procedurally generate content and adapt it to the player.
After an engine is chosen, a small roleplaying game will be developed to test the created features. A rough work plan is stated in section 5, followed by the implications on research in section 6.

Contents

Abstract iii

1. Introduction 1

2. Thesis statement 3

3. Approach and Methods 5

3.1. Cost-utility Analysis 6

4. Preliminary Results 15

4.1. Open Source Game Engines 15

4.2. Closed Source Free of Charge Engines 17

4.3. Results 18

5. Work Plan 19

6. Implications of Research 21

A. Appendix 23

A.1. Engine Cost-Utility Analysis 23

1

Introduction

Procedurally generated content has been used in computer and video games since the very early days and one of its first usage was to save precious space on disks and memory. The 1984 game Elite by David Brabam and Ian Bell managed to store 8 galaxies with 256 star systems each in the memory of an BBC Micro home computer which was only 48 kilobytes. In the 2006 XBox Live Arcade game Roboblitz, textures were procedurally generated to save disk space as Microsoft enforced strict size limits on downloadable titles. Furthermore, advanced algorithms are heavily used in the demo scene where file sizes of executables are restricted to sizes that are less than an average document file today.

Another use is to increase the replay value of games by randomly generating levels

(Examples: Rogue[1], Diablo[2], Dwarf Fortress[3]), items (Diablo[2], Borderlands[4]) and even adapt to the players play style (Galactic Arms Race[5]). For some of those task, middleware systems are available, for example CityEngine[6] for urban environments, Terragen[7] for landscapes and Speedtree[8] for vegetation.

The term *serious games* was coined by Clark C. Abt in his 1968 Book of the same name[1], which describes the concepts of games whose primary focus is not entertainment, but teaching. This does not mean they cannot be fun, it is just not their primary intention. procedurally generated content (PCG) has also been used in serious games, mainly to generate different scenarios and adapt to players.

At Ulm University, a joint project[2] involving the departments for computer science and psychology has launched in 2014 with the goal to develop processes and approaches to make better serious games. The project is funded by the Carl Zeiss Stiftung[3]. It focuses both on didactic and technical challenges, which is the reason why both departments with their respective strengths are brought together.

In this thesis, the goal is to develop a platform used by students and researchers for serious games over the course of the research project.

[1] Michael Toy and Glenn Wichman, around 1980
[2] Blizzard Entertainment, 1996, www.diablo.com
[3] Tan Adams, 2006, www.bay12games.com/dwarves/
[4] Gearbox Software, 2009, www.borderlandsthegame.com
[5] Evolutionary Games, 2009, http://galacticarmsrace.blogspot.com/
[6] http://www.esri.com/cityengine
[7] http://www.planetside.co.uk/products/terragen3
[8] http://www.speedtree.com/

2

Thesis statement

The goal is to develop a software platform based on an existing game engine with extensions to help developing serious games in the future. In a modular architecture, a module to procedurally generate content and adapt it to players performance and physical condition will be developed.

3

Approach and Methods

First, the goal is to find a game engine suitable for the needs of serious games. Finding a suitable engine is critical, as it will influence the course of the thesis and is not easily revertible, therefore the decision making process has to be done carefully.
The engine must have all the functionality needed to start developing a game without having to make changes in its core. The learning curve should not be too steep, as it will be used by other at Ulm University without major experience in software development. It would be beneficial if the gameplay code can be written in a language different than C/C++, as it is not part of the major lectures but Java instead.
For a early evaluation, a cost-utility analysis is done to find two to three potential candidates, which then are examined in detail. With those candidates, a small game is developed to test its capabilities, ease of use and time needed to familiarize with the engines.
For procedural content generation, a suitable approach has the be chosen. Criteria are

complexity of implementation, use in serious and commercial games, and performance. Also, the randomness of the results has to be controlled, not all will be suited as they have to be playable.
To test the platform, a small RPG game will im implemented using the game engine and the PCG. Content not yet seen by the player should be generated according to his performance and physical condition, measured by a wristband. Adaption should be as subtle as possible, because if the players notices that the difficulty is raising or dropping, it could have an impact on his behavior. It will be checked if the game does cover all the needs of the generation algorithms, if not additional small games will be developed to cover them.

3.1. Cost-utility Analysis

Game development is not only a big industry, but also a widespread hobby among software developers. As a result, an enormous number of commercial and free, open and closed source game engines have been developed. The Wikipedia List of Game Engines[4] lists 253 engines and libraries in total, while the DevDB[5] on the developer resource website `www.devmaster.net`[6] lists 368 engines. It is not possible to try every one of them therefore they have to be filtered to a bearable number.Beforehand, engines that are not flexible enough are eliminated, mostly 2D and 2.5D engines and engines that specialize in a certain genre.
Cost-utility analysis is a form of financial analysis. In this work the so called "Nutzwert-analyse" is used, which differs from the Cost-utility analysis in english speaking contests, the latter is mostly used in health economics. The basic idea is to define criteria, categorize and weight them. The process itself is by far not as elaborate as other processes in decision making, but here the main use is to filter a vast amount of candidates in a short time period. The weights and degree of fulfillment are set subjectional and were updated multiple times over the course of the thesis.
The weights are set 0 (not applicable) to 9 (very important), for a detailed list see table 3.1a and degrees of fulfillment are set on a 0 (not available) to 5 scale (very good), detailed shown in table 3.1b.

Score	Weight
0	-
1-2	nice to have
3-5	medium
6-8	important
9	very important

(a) Weights used in cost utility analysis

Score	Degree of Fulfillment
0	not available
1	poor
2	fair
3	satisfactory
4	good
5	very good

(b) degrees of fulfillment used in cost-utility analysis

As reference, the DevDB is used as it provides more information about the activity of the engine, with additional candidates added by recommendations of the research staff at Ulm University. First, only active engines were considered and all without or with a very small number user reviews are eliminated. The list of remaining engines can be found in table 3.1. Next, the cost-utility analysis is done, the criteria are listed in table 3.2. The following section describes the criteria in detail and the one afterwards shows and discusses the results.

Selected Engines The selection consists of some well known and lesser know engines. Generally, open source solutions are prefered, as they provide the ability to do change at the core if needed (which is not intented) and independence from companies. Most of the proprietary engines have multiple licencing levels, including access to source code but only for a large sum, which is not suited for an academic budget. Some engines are only available through a licencing fee, but may be available for academic purposes at a reduced fee or even without cost. This is the case for the Unreal Development Kit, which is a free version based on the Unreal Engine 3 by Epic Games. It was initially not a candidate, since the focus of the engine is on big budget titles. But with the increasing importance of small, independent game developer for the industry, having a cheaper version with less functionality can attract small developer teams and even students. Therefore, Epic Games (and other companies) have started a program targeted to get more academic users into Unreal Engine 4[7].

Table 3.1.: considered game engines in cost-utility analysis.

Name	Supported Platforms	Language Engine	Language other	Licence
Blender Game Engine	Windows, Linux, MacOS X	C/C++, Python	C/C++, Python	GPL
Cafu Engine	Windows, Linux	C/C++	C/C++	GPL
Crystal Space	Windows, Linux, MacOS X	C/C++	C/C++	LGPL
Esenthel Engine	Windows, Linux, MacOS X, Browser, Android, iOS	C/C++, Java, Obj-C, JavaScript	C/C++, Obj-C, JavaScript	Proprietary, free available
Irrlicht	Windows, Linux, MacOS X	C/C++, C#, VB.net	C/C++, C#, VB.net	ZLIB
jMonkey	Windows, Linux, MacOS X, Browser, Android	Java	Java, Ruby, Python, Javascript, PHP	BSD
libgdx	Windows, Linux, MacOS X, Browser, Android, iOS	Java, C/C++	Java	Apache 2
Neoaxis 3D Engine	Windows, MacOS X	C/C++, C#	C/C++, C#	Proprietary, free available
OGRE	Windows, Linux, MacOS X	C/C++	C/C++	MIT

Panda3D	Windows, Linux, MacOS X	C/C++, Python	C/C++, Python	BSD
ShiVa Engine	Windows, Linux, MacOS X, Browser, Android, iOS	C/C++	C/C++	Proprietary, free available
Unity	Windows, Linux, MacOS X, Browser, Android, iOS	C/C++	C#, JavaScript	Proprietary, free available
C4 Engine	Windows, Linux, MacOS X	C/C++	C/C++	Proprietary, cost
DX Studio	Windows	C/C++, VB	C/C++, VB	Proprietary, free available
Leadwerks Engine	Windows	C/C++, C#, Java, Perl, Python	C/C++, C#, Java, Perl, Python	Proprietary, cost
Visual3D Game Engine	Windows, Browser	C#	C/C++. C#, Ruby, Python, F#, Lua	Proprietary, cost
Unreal Development Kit	Windows, Linux, MacOS X, Browser, Android, iOS	C/C++	C/C++	Proprietary, cost

3.1.1. Criteria

In this section, the criteria used in the cost-value analysis are detailed. An overview is shown in table 3.2.

Licence

While game engines with open source licences are generally prefered, closed source solutions are not ruled out if other criteria have good scores. Therefore, open source licencees get a four times higher weight than closed source, with the other scored with 0.

Technical Aspects

Technical aspects cover criteria programming language, supported functionality, architecture and supported platforms.
Most game engines are written in C/C++, mostly for performance reasons, but some can use code written in a different language through bindings or interpreters such as Python or .NET languages as C#. This is beneficial as C/C++ is not in the curriculum of the major lectures at Ulm University, which mainly focuses on Java as initial programming languages. Since most students unfamiliar with it tend to have an avoidance of lower layered languages and often only heard of the downsides (pointer arithmetic in particular), at least having an option to write code in a different language would benefit adoption rate. Changes at the core of the engine are not intended.
Functionality is not a key point, as most engines support all the basic functions in terms of graphics, sound and input. Advanced features, especially in graphics, are not a hard requirement.
Supported platforms focuses mainly on computers with Windows, Linux or MacOS. Having the ability to deploy a game in a browser environment would benefit testing as setup time would basically be non-existent. Mobile platforms, in particular Apple iOS and Android would also be beneficial. Availability of either of them or both will be rewarded with additional points. Additional platforms such as gaming consoles are of no additional value and therefore not be rewarded with points.

Activity of Developers

Since the engine decided on will be use in future projects, it should be actively developed and maintain by the developers to avoid running into a dead end or having to do more work on our own which not is not tied in with developing a game, but fixing bugs and developing additional functionality. Updates should be released in regular cycles. Having developers actively interacting with the community would benefit the learning curve or time needed to fix problems, as quick help would be available.

Tool Support

Having the ability to use the engine and develop games in familiar Integrated Development Environment (IDE) would be beneficial in terms of familiarization with the engine. In our case, eclipse is the standard IDE, although mostly students can choose what they want as long as all needed functionality is available, but are then on their own. The ability to use assets from artistic tools to create models, animations and textures would also be of benefit. If the engine itself provides tools for tasks like programming, modelling or level design would reduce development time and the need to find compatible solutions.

Community

An engine can be good on paper, but if it is not widely used it is of no benefit for us. A widespread engine with experienced users can help in easing the learning curve and solving problems quickly. Generally the number of active projects and their general progress is good indicator, although not all problems are rooted in the used technology..

Ease of Learning

The most important aspect is the ease of learning of the engine. As the projects will mostly developed by students, it cannot be expected from them to have much experience in developing projects of this scope, but is also a learning process. Therefore, having a

complex engine is of no benefit. A steep learning curve is probably the most significant aspect, which can be eased by having sufficient literature (ideally both analog and digital) and tutorials. Seen the scope of a game engine, good documentation ist a must and would be beneficial to developers. Having support directly from the developers can also be a great help, as their knowledge of the inner workings of the engine usually surpasses that of an average user which would benefit in solving problems related more to the internals. This criteria is closely tied in with the general activity of the developers.

Table 3.2.: Criteria for cost-utility analysis

Criteria	**Weight**
licence	*10*
open source	8
closed source	2
technical aspects	*30*
programming language engine	4
programming language gamecode	7
supported platforms	6
functionality	5
architecture	8
activity of developers	*20*
frequency of updates	7
time since last update	7
presence in forums/wikis/chats	6
tool support	*16*
integration in IDEs	7
support for modelling/animation tools	5
provided tools	4
community	*17*
number of projects	5
activity of projects	6
activity in forums/wikis/chats	6

ease of learing	*33*
learning curve	9
available literature/tutorials	9
documentation	8
support by developers	7
Total:	**135**

4

Preliminary Results

In this section, the results of the cost-utility analysis are presented and discussed. For a detailed results, the tables for each engine are found in the appendix A.1.

4.1. Open Source Game Engines

The **Blender Game Engine** is an extension to the modelling tool Blender[1]. Overall it is a decent engine, but it lacks developer support and in the future it will be developed to be an tool for game prototypes, architectural walkthroughs and scientific simulators[8], which is not the way the research project is going and will be most likely a dead end.
Cafu is also a decent engine with an outdated feature set for visuals. It seems like development has not stopped, but slowed down a great deal to the point where the

[1] http://www.blender.org/

commits to the source code version control system only appear weeks apart, are small and the last official release dates back to 2012 - not a good outlook and a potential dead end. **Crystal Space** is a very similar case, both have not been used in a lot of projects and activity by developers and users is decent.

Irrlicht is probably one of the oldest still developed engines and is used in quite a few open source and commercial projects, but the last official release was in 2013, commits to their version control system have slowed down. The biggest downside is that it is mainly a graphics engine, everything else has to be provided by add-ons which might lead into a dead end if development of the plugins stopped or the interfaces in the core engine changes. Overall a solid engine, but it lacks additional tools and steady development.

jMonkey is one of the two Java-based engine in this comparison and has a good set of features with everything needed from a programmers standpoint and good platform support. Learning curve seems good and quite a few projects are active, but it seems that the development of the engine has slowed down more and more over the last months. A major downside is the lack of support for common file formats of modelling tools, only Blender models are currently supported. It is possible to run jMonkey projects in a browser, but only as a Java Applet, which got a bad reputation for being security risk due to flaws in the Java Runtime Environment.

Its competitor **libgdx**, is heavily developed with new releases every couple of weeks and an comparable set of features. It also only supports a few model data types, mostly form open source software like Blender and brings some editors with it, but mostly for minor task like creating particles. On the plus side it supports all needed platforms and due its highly active development, **libgdx** makes it into the final evaluation.

OGRE is a very old engine but still actively developed, with a big community and many tutorials and literature available. Despite having a competitive score, it will not be evaluated in the final round. The reason is that it is many a graphics engine and all additional functionality is provided via add ons. This could be fatal if they are not developed by the core-developers and interface change, which is a risk not worth taking and the over features did not bring enough to justify it.

The last of the open source engines in **Panda3D**, formerly developed closed by Disney

and now in the hands of the Carnegie Mellon University. It has a solid features set and has been used in a good number of free and commercial projects. It is a solid engine overall, but will not be considered further as it has almost no active developers. It seems like is only developed if a student of CMU uses the engine, but otherwise not much happens. And it does not stand out in any other criteria to justify a place in the final evaluation.

4.2. Closed Source Free of Charge Engines

Esenthel is very interesting engine with a feature set comparable to commercial engines and passable licencing fees if desired, but they are bound to the number of developers. The community and number of projects seems small, but is highly active as is the development of the engine. It supports all desired platforms and has a good number of tools with it. The biggest downside is that it can only be used with C/C++ code, which might be hard to convince student to use the engine, but due to its otherwise high score it is included in the final round of evaluation.

The **NeoAxis** engine is based on OGRE, but greatly enhanced by visual updates, more features and editors. It is actively developed, with major releases every six months. It is overall a solid engine, but lacks features that stand out and lacks documentation and tutorials.

Shiva 3D seem overall like a solid engine, supported platforms and a features set competitive to commercial engines stand out. The downside is that development seems rather troublesome, the next big version is overdue for 2 years with a beta version only released recently, the last major version being more than four years old with a small bugfix version released in december 2013. Despite otherwise seeming like a good engine, this could prove troublesome which also shows in the scores and therefore it does not make it into the final round.

Unity is probably one of the most used engines at the moment, from small two man teams up to large budged productions. Its feature set is unparalleled by most other engines, especially in provided tools which make almost every other software (including an IDE and modelling tools) unnecessary for small projects. It is very actively developed,

well documented and the large community provides much help and tutorials, in addition to the already huge number provided by the developer. It supports all platforms wanted and gameplay code can be written in convenient languages (C#, Boo (Python implementation) and UnityScript (JavaScript dialect). In combination, Unity achieves the highest score of all engines is therefore included in the final round of evaluation.

A bit of special place has the **Unreal Development Kit** and **Unreal Engine 4** in this comparison. Both are mostly used by big budget products with professionals of all needed crafts involved. This make for a features set even outperforming Unity, especially in graphical fidelity, where the engine is one of the most impressive on the market. To bring smaller teams to use the engine, developer Epic Games has released the UDK for free, which is based in the Unreal Engine 3. It has all the major features and a huge amount of tools provided with it. The downside is the complexity, which might be too much for small projects and students not familiar with a software this size and coded in C++. Additionally, the current Unreal Engine 4 is available for academic use for free (but still has a 5% cut of yield has to be paid for sold products, which is not the case here), but with even more functionality and complexity, it is left out of the final round of evaluation despite having the second highest score over all. But it might be a viable option in the future, especially if students already worked with it.

4.3. Results

In the final round of evaluation, three engines will be evaluated in detail: Unity (Score: 486), Esenthel (437) and libgdx (429). Every engine will be used to develop a small game to see how fast a usable result is possible and the amount of familiarization needed. This will be described in the thesis.

5

Work Plan

First discussions of the topic and ideas took place in April of 2014, followed by phase of initial research. Due to other projects and lectures, full work on the thesis started in Mid october 2014 with the evaluation of game engines, which is expected to be finished by mid november 2014. Until the end of 2014, designing the architecture of the development platform and first implementation of the prototype game and PCG-module is planned. By then it will be evaluated if the PCG-module can be used to generate content for games independent of the genre. If this is not the case, other, small prototypes of other games will be developed to cover the missing cases.
In 2015, the prototype game and PCG-module is to be finished in February. Afterwards a technical evaluation is planned and the documentation and thesis is intended to be finished by the end of march 2015.

6

Implications of Research

In parts, everything has been done already: procedural content generation, serious games, serious game RPG, adaptive content generation. The combination is new as well as the usage of a device to measure bodily functions to use with adaptive content generation.

A

Appendix

A.1. Engine Cost-Utility Analysis

Blender Game Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*40*
open source	8	5	40
closed source	2	0	0
Technical	*30*	*15*	*96*
Programming Language Engine	4	2	8
Programming Language Gamecode	7	4	28
Supported Platforms	6	3	18
Functionality	5	2	10
Architecture	8	4	32
Activity Developer	*20*	*7*	*46*
Frequency of updates	7	2	14
Time since last update	7	2	14
Activity in Forums/Wiki/Chats/etc.	6	3	18
Tool Support	*16*	*11*	*58*
Integration in IDEs	7	3	21
Support for Modelling/Animation Tools	5	5	25
provided tools	4	3	12
Community	*17*	*5*	*29*
Number of Projects	5	1	5
Activity of Projects	6	2	12
Activity in Forums/Wiki/Chats/etc.	6	2	12
ease of learning	*33*	*11*	*92*
learning curve	9	3	27
available literature/tutorial	9	3	27
documentation	8	3	24
support by developers	7	2	14
Total	**126**	**54**	**361**

Table A.1.: cost-utility analysis of Blender Game Engine

Cafu Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*40*
open source	8	5	40
closed source	2	0	0
Technical	*30*	*15*	*96*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	4	28
Supported Platforms	6	2	12
Functionality	5	4	20
Architecture	8	3	24
Activity Developer	*20*	*7*	*46*
Frequency of updates	7	3	21
Time since last update	7	1	7
Activity in Forums/Wiki/Chats/etc.	6	3	18
Tool Support	*16*	*9*	*46*
Integration in IDEs	7	2	14
Support for Modelling/Animation Tools	5	4	20
provided tools	4	3	12
Community	*17*	*8*	*46*
Number of Projects	5	2	10
Activity of Projects	6	3	18
Activity in Forums/Wiki/Chats/etc.	6	3	18
ease of learning	*33*	*12*	*97*
learning curve	9	3	27
available literature/tutorial	9	2	18
documentation	8	3	24
support by developers	7	4	28
Total	**126**	**57**	**371**

Table A.2.: cost-utility analysis of Cafu Engine

Crystal Space			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*40*
open source	8	5	40
closed source	2	0	0
Technical	*30*	*18*	*110*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	4	28
Supported Platforms	6	3	18
Functionality	5	4	20
Architecture	8	4	32
Activity Developer	*20*	*7*	*46*
Frequency of updates	7	3	21
Time since last update	7	1	7
Activity in Forums/Wiki/Chats/etc.	6	3	18
Tool Support	*16*	*5*	*27*
Integration in IDEs	7	1	7
Support for Modelling/Animation Tools	5	4	20
provided tools	4	0	0
Community	*17*	*9*	*51*
Number of Projects	5	3	15
Activity of Projects	6	3	18
Activity in Forums/Wiki/Chats/etc.	6	3	18
ease of learning	*33*	*10*	*81*
learning curve	9	1	9
available literature/tutorial	9	3	27
documentation	8	3	24
support by developers	7	3	21
Total	**126**	**54**	**355**

Table A.3.: cost-utility analysis of Crystal Space

Esenthel Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*10*
open source	8	0	0
closed source	2	5	10
Technical	*30*	*20*	*120*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	3	21
Supported Platforms	6	5	30
Functionality	5	5	25
Architecture	8	4	32
Activity Developer	*20*	*14*	*94*
Frequency of updates	7	5	35
Time since last update	7	5	35
Activity in Forums/Wiki/Chats/etc.	6	4	24
Tool Support	*16*	*13*	*69*
Integration in IDEs	7	4	28
Support for Modelling/Animation Tools	5	5	25
provided tools	4	4	16
Community	*17*	*8*	*46*
Number of Projects	5	2	10
Activity of Projects	6	3	18
Activity in Forums/Wiki/Chats/etc.	6	3	18
ease of learning	*33*	*12*	*98*
learning curve	9	2	18
available literature/tutorial	9	3	27
documentation	8	4	32
support by developers	7	3	21
Total	**126**	**72**	**437**

Table A.4.: cost-utility analysis of Esenthel Engine

Irrlicht Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*40*
open source	8	5	40
closed source	2	0	0
Technical	*30*	*15*	*90*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	3	21
Supported Platforms	6	3	18
Functionality	5	3	15
Architecture	8	3	24
Activity Developer	*20*	*7*	*46*
Frequency of updates	7	2	14
Time since last update	7	2	14
Activity in Forums/Wiki/Chats/etc.	6	3	18
Tool Support	*16*	*10*	*56*
Integration in IDEs	7	4	28
Support for Modelling/Animation Tools	5	4	20
provided tools	4	2	8
Community	*17*	*9*	*51*
Number of Projects	5	3	15
Activity of Projects	6	2	12
Activity in Forums/Wiki/Chats/etc.	6	4	24
ease of learning	*33*	*13*	*107*
learning curve	9	2	18
available literature/tutorial	9	4	36
documentation	8	4	32
support by developers	7	3	21
Total	**126**	**59**	**390**

Table A.5.: cost-utility analysis of Irrlicht Engine

jMonkey Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*40*
open source	8	5	40
closed source	2	0	0
Technical	*30*	*19*	*115*
Programming Language Engine	4	4	16
Programming Language Gamecode	7	4	28
Supported Platforms	6	4	24
Functionality	5	3	15
Architecture	8	4	32
Activity Developer	*20*	*9*	*60*
Frequency of updates	7	3	21
Time since last update	7	3	21
Activity in Forums/Wiki/Chats/etc.	6	3	18
Tool Support	*16*	*7*	*39*
Integration in IDEs	7	3	21
Support for Modelling/Animation Tools	5	2	10
provided tools	4	2	8
Community	*17*	*9*	*52*
Number of Projects	5	2	10
Activity of Projects	6	3	18
Activity in Forums/Wiki/Chats/etc.	6	4	24
ease of learning	*33*	*14*	*116*
learning curve	9	3	27
available literature/tutorial	9	4	36
documentation	8	4	32
support by developers	7	3	21
Total	**126**	**63**	**422**

Table A.6.: cost-utility analysis of jMonkey Engine

libgdx			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*40*
open source	8	5	40
closed source	2	0	0
Technical	*30*	*19*	*113*
Programming Language Engine	4	4	16
Programming Language Gamecode	7	4	28
Supported Platforms	6	5	30
Functionality	5	3	15
Architecture	8	3	24
Activity Developer	*20*	*14*	*94*
Frequency of updates	7	5	35
Time since last update	7	5	35
Activity in Forums/Wiki/Chats/etc.	6	4	24
Tool Support	*16*	*6*	*37*
Integration in IDEs	7	4	28
Support for Modelling/Animation Tools	5	1	5
provided tools	4	1	4
Community	*17*	*8*	*46*
Number of Projects	5	2	10
Activity of Projects	6	3	18
Activity in Forums/Wiki/Chats/etc.	6	3	18
ease of learning	*33*	*12*	*99*
learning curve	9	3	27
available literature/tutorial	9	3	27
documentation	8	3	24
support by developers	7	3	21
Total	**126**	**64**	**429**

Table A.7.: cost-utility analysis of libgdx

NeoAxis Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*10*
open source	8	0	0
closed source	2	5	10
Technical	*30*	*17*	*101*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	4	28
Supported Platforms	6	2	12
Functionality	5	5	25
Architecture	8	3	24
Activity Developer	*20*	*12*	*80*
Frequency of updates	7	4	28
Time since last update	7	4	28
Activity in Forums/Wiki/Chats/etc.	6	4	24
Tool Support	*16*	*12*	*61*
Integration in IDEs	7	3	21
Support for Modelling/Animation Tools	5	4	20
provided tools	4	5	20
Community	*17*	*9*	*52*
Number of Projects	5	2	10
Activity of Projects	6	3	18
Activity in Forums/Wiki/Chats/etc.	6	4	24
ease of learning	*33*	*11*	*89*
learning curve	9	3	27
available literature/tutorial	9	2	18
documentation	8	2	16
support by developers	7	4	28
Total	**126**	**66**	**393**

Table A.8.: cost-utility analysis of NeoAxis Engine

OGRE Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*40*
open source	8	5	40
closed source	2	0	0
Technical	*30*	*14*	*85*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	3	21
Supported Platforms	6	3	18
Functionality	5	2	10
Architecture	8	3	24
Activity Developer	*20*	*11*	*73*
Frequency of updates	7	4	28
Time since last update	7	3	21
Activity in Forums/Wiki/Chats/etc.	6	4	24
Tool Support	*16*	*4*	*24*
Integration in IDEs	7	2	24
Support for Modelling/Animation Tools	5	2	10
provided tools	4	0	0
Community	*17*	*14*	*28*
Number of Projects	5	4	20
Activity of Projects	6	5	30
Activity in Forums/Wiki/Chats/etc.	6	5	30
ease of learning	*33*	*15*	*123*
learning curve	9	2	18
available literature/tutorial	9	5	45
documentation	8	4	32
support by developers	7	4	28
Total	**126**	**63**	**425**

Table A.9.: cost-utility analysis of OGRE Engine

Panda3D Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*40*
open source	8	5	40
closed source	2	0	0
Technical	*30*	*14*	*85*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	4	28
Supported Platforms	6	3	18
Functionality	5	3	15
Architecture	8	3	24
Activity Developer	*20*	*6*	*39*
Frequency of updates	7	1	7
Time since last update	7	2	14
Activity in Forums/Wiki/Chats/etc.	6	3	18
Tool Support	*16*	*6*	*31*
Integration in IDEs	7	1	7
Support for Modelling/Animation Tools	5	3	20
provided tools	4	1	4
Community	*17*	*9*	*52*
Number of Projects	5	2	10
Activity of Projects	6	3	18
Activity in Forums/Wiki/Chats/etc.	6	4	24
ease of learning	*33*	*13*	*107*
learning curve	9	3	27
available literature/tutorial	9	3	27
documentation	8	4	32
support by developers	7	3	21
Total	**126**	**55**	**366**

Table A.10.: cost-utility analysis of Panda3D Engine

Shiva3D Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*10*
open source	8	0	0
closed source	2	5	10
Technical	*30*	*19*	*112*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	3	21
Supported Platforms	6	5	30
Functionality	5	5	30
Architecture	8	3	24
Activity Developer	*20*	*6*	*39*
Frequency of updates	7	2	14
Time since last update	7	1	7
Activity in Forums/Wiki/Chats/etc.	6	3	18
Tool Support	*16*	*9*	*43*
Integration in IDEs	7	1	7
Support for Modelling/Animation Tools	5	4	20
provided tools	4	4	16
Community	*17*	*8*	*46*
Number of Projects	5	2	10
Activity of Projects	6	3	18
Activity in Forums/Wiki/Chats/etc.	6	3	18
ease of learning	*33*	*11*	*90*
learning curve	9	2	27
available literature/tutorial	9	3	27
documentation	8	3	24
support by developers	7	3	21
Total	**126**	**58**	**340**

Table A.11.: cost-utility analysis of Shiva3D Engine

Unity Engine			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*10*
open source	8	0	0
closed source	2	5	10
Technical	*30*	*21*	*127*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	4	30
Supported Platforms	6	5	30
Functionality	5	5	30
Architecture	8	4	32
Activity Developer	*20*	*13*	*88*
Frequency of updates	7	5	35
Time since last update	7	5	35
Activity in Forums/Wiki/Chats/etc.	6	3	18
Tool Support	*16*	*9*	*41*
Integration in IDEs	7	0	0
Support for Modelling/Animation Tools	5	5	25
provided tools	4	4	16
Community	*17*	*15*	*85*
Number of Projects	5	5	25
Activity of Projects	6	5	30
Activity in Forums/Wiki/Chats/etc.	6	5	30
ease of learning	*33*	*17*	*142*
learning curve	9	4	36
available literature/tutorial	9	5	45
documentation	8	5	40
support by developers	7	3	21
Total	**126**	**80**	**493**

Table A.12.: cost-utility analysis of Unity Engine Engine

Unreal Development Kit			
Criteria	**Weight**	**degree of fulfillment**	**Score**
licence	*10*	*5*	*10*
open source	8	0	0
closed source	2	5	10
Technical	*30*	*20*	*121*
Programming Language Engine	4	3	12
Programming Language Gamecode	7	4	28
Supported Platforms	6	5	30
Functionality	5	5	30
Architecture	8	3	24
Activity Developer	*20*	*14*	*94*
Frequency of updates	7	5	35
Time since last update	7	5	35
Activity in Forums/Wiki/Chats/etc.	6	4	24
Tool Support	*16*	*12*	*59*
Integration in IDEs	7	2	14
Support for Modelling/Animation Tools	5	4	24
provided tools	4	5	20
Community	*17*	*14*	*80*
Number of Projects	5	4	20
Activity of Projects	6	5	30
Activity in Forums/Wiki/Chats/etc.	6	5	30
ease of learning	33	15	124
learning curve	9	2	18
available literature/tutorial	9	5	45
documentation	8	5	40
support by developers	7	3	21
Total	**126**	**80**	**486**

Table A.13.: cost-utility analysis of Unreal Development Kit

List of Tables

3.1. considered game engines in cost-utility analysis 7
3.2. Criteria for cost-utility analysis 12

A.1. cost-utility analysis of Blender Game Engine 24
A.2. cost-utility analysis of Cafu Engine 25
A.3. cost-utility analysis of Crystal Space 26
A.4. cost-utility analysis of Esenthel Engine 27
A.5. cost-utility analysis of Irrlicht Engine 28
A.6. cost-utility analysis of jMonkey Engine 29
A.7. cost-utility analysis of libgdx 30
A.8. cost-utility analysis of NeoAxis Engine 31
A.9. cost-utility analysis of OGRE Engine 32
A.10.cost-utility analysis of Panda3D Engine 33
A.11.cost-utility analysis of Shiva3D Engine 34
A.12.cost-utility analysis of Unity Engine Engine 35
A.13.cost-utility analysis of Unreal Development Kit 36

Bibliography

[1] ABT, C.C.: *Serious Games*. University Press of America, 1987 http://books.google.de/books?id=axUs9HA-hF8C. – ISBN 9780819161482

[2] *Über das Projekt - Universität Ulm*. http://www.uni-ulm.de/in/serious-games/ueber-das-projekt.html, . – Last accessed: 2014/10/28

[3] *Carl Zeiss Stiftung*. http://carl-zeiss-stiftung.de/, . – Last accessed: 2014/10/28

[4] *List of game engines - Wikipedia, the free encyclopedia*. https://en.wikipedia.org/wiki/List_of_game_engines, . – Last accessed: 2014/10/25

[5] *DevDB - Database of Game Development Resources | DevMaster*. http://devmaster.net/devdb/, . – Last accessed: 2014/10/25

[6] *DevMaster - game development news, discussions, and resources*. http://www.http://devmaster.net/, . – Last accessed: 2014/10/25

[7] DAVIS, Ray: *Unreal Engine 4 Goes Free for Academic Use*. https://www.unrealengine.com/blog/unreal-engine-4-goes-free-for-academic-use, . – Last accessed: 2014/10/25

[8] ROOSENDAAL, Ton: *Blender roadmap - 2.7, 2.8 and beyond | Blender Code*. http://code.blender.org/index.php/2013/06/blender-roadmap-2-7-2-8-and-beyond/, . – Last accessed: 2014/10/28

Glossary

IDE Integrated Development Environment. 11, 17

PCG procedurally generated content. 2, 19

Name: Michael Legner Matrikelnummer: 800817

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Michael Legner