Laminin

Tender application for XGame presented by Derivco. The purpose of this document is to give an overview of why we should be awarded the project and how we aim to achieve success and complete the project.

2014

Nico Taljaard
10153285

Gerhard Smit
12282945

Martin Schoeman
10651994

TABLE OF CONTENTS

Group Information

Here is some information about the group:

We are three programmers studying at the University of Pretoria, each for a degree that falls under the computer science department. We all have a passion for programming and love a challenge. Also we all have years of experience in different programming languages like java. We also have had our fair share of playing games on different platforms and know exactly what is needed to make a fun and exciting game. Some of us have experience in tools and languages that not all students have, like CryENGINE, OpenGL, Maya and 3ds Max. There will be one hundred percent commitment from our side, from the beginning to the end of the project.

The rest of our individual information can be found in the team portfolio.

Vision & Objective

The vision of this project is to determine if a cross platform game can be created in a short period of time. This game should also have backend components, consisting of a server connection enabling users to share their game progression and compare themselves with other players. All of the systems as well as the backend should only consist of free or open source technologies to allow free usage and free modifications to be applied where needed.

As for the objective, achieving this through the use of only open source and freeware technologies, will prove that more independent creators may have alternative methods to developing their own creations. This method can be a starting point for a striving game developer to compile a portfolio of his creations. It can also be used to expand the existing open source resources and enriching the community with ideas on how to use these technologies, quickly and efficiently.

Requirements

- Do research on finding open source or freeware technologies this must include:
  - Game engine
  - All libraries used: Physics, UI, Networking, Terrain system, Import and utilize libraries.
  - Resources: Textures, Models, Music, Sound Effects.
- Develop the simplistic game that is possible to be completed within 3 months
- The Game should be cross platform with at least two different options used or a HTML application that can be run from the browser.
- OAuth integration must be used for the login system connecting the user to Facebook, Twitter or G+ for achievement updates.
- Custom login:
  - If anonymity is desired.
  - Encrypt password when stored.

  - Retrieve password feature.
- A live leader board must be integrated for real time standing for more competitive users.

Game Proposal

We have decided that for this project, on a simplistic game that can be done in rapid development. Where we prefer having a good looking game with all the features of a good game and without cutting down on depth and game play time too much. Our idea is to implement a '3D slasher' game which will only consist of one 'open world' map focusing on the environment. This route is decided upon for implementing multiple maps, but does not drastically increase difficulty but rather more on scope and time consumption. Doing this we will be able to explore a wider variety of technologies as well as implementations to demonstrate the effects of these technologies.

We are aiming for a minimalistic game that will have all the various fascists of a game, where all extended features and content will be seen as a nice to have which is listed below. It will start out with a single character to be used by the player. When logging in the character level, current position on the map and current experience will be sent to the client to start the game. The 'open world' will have a sense of exploration, up to a degree, mobs will not have an overwhelming presence to promote admiration of the environment. The size of the map/maps will be dependant on the memory available, this will be decided at a later stage. This world will consist of multiple mobs from humanoid to animals, which will be able to attack and kill you upon detection. All animations will be kept to a minimal to save time for implementing a better looking environment, game play and cleaner implementation.

A levelling system will be implemented, that will be updated to the leader board when enough experienced is gained to level. The experience will be calculated according to the amount of health you have lost during an encounter, what level the mob is and how fast you were able to finish it off, the exact formula will be experimented with during testing. This experience value will be update after each kill, sending it to the back-end server, which will keep track of all user level and experience. From this the leader board can be pulled in game to see your standing. At each level up you will be prompted if you would like to post your achievement to either Facebook, Twitter or G+ depending if you are logged on through OAuth.

All values of your character will be calculated at a later stage to determine effectiveness this will include: health, attack strength and defence. Through levelling these values will be increased accordingly to be able to face higher level mobs.

Nice-To-Have (During and after project completion):

- Multiple maps.
- Improvement of animations.
- Improvement on character textures and modelling.
- Further the achievement system with can be shared on social media consisting of:
  - Killing on of each mob.
  - Killing amounts of curtain mobs.
- Attribute system that will allow users to increase character values according to there game play.

- Gear system for characters.
- Looting system for mob kills to be sold or equipped.
- Look into adding multi-player support through SpiderMonkey.

This is the basic overview of what we aim to accomplish during this project. Any further requirements and ideas will be determined if we are selected to take up this project and have met with the client to go over the proposal, if this is an acceptable approach and idea. We are excited to take up this project as we are game enthusiasts who would appreciate it if we were to be picked, to undertake this research experiment with you.

Software Research

When we were conducting our research we where astonished of how many open source and freeware technologies/resource are available for use. A new independent developer has a hard time deciding on which of these will be most suitable for the extent and limitations of his skill, requirements and preference. All of this might be a bit overwhelming but we have decided upon the following software and resources that we will be using to implement the game:

Game Engine

We have decide to use the following:
jMonkeyGameEngine 3

The reason for this is that most open source engines and freeware engines are C# and most of engines that use C++ or Java do not support cross platform or is not free to use. This engine is developed in Java and makes use of Java for implementation, it can be downloaded as a Netbeans package to simplify installation and gives an interface that is more commonly used by first time developers. It is pre-compiled and outfitted with most of the libraries and accessories that you would require to develop your game all installed or can be downloaded as plugins. The reason for choosing Java above C++, even though C++ is stronger and faster than Java, is due to the requirement of rapid development. To be able to achieve this Java offers a lot more open libraries free of use, mostly seeing as this is the first game we will be undertaking the ease of not needing to do memory management will save even more time. Another reason we decided on the use of Java, is that when looking at possibilities of implementing design patterns it will be more time efficient. To further the reasons as to why we chose this engine, compared to others is that this engine does support OpenGL 1.x up to 4.x which is not frequently seen or catered for on these type of engines. This allows us to use all the new features that was rolled out with OpenGL 4 like the freedom of shader usage.

Design Engine

We have decide to use the following:
Blender & Gimp

The reason for choosing Blender is that it is freeware software that does everything you would expect from a graphic design engine. This engine will be used to edit all textures, models and perform all the animations that we will require. It is a world wide used software by independent developers and ever small developer companies, that can not afford a high end game engine like Unreal, CryEngine as well as the use of the Autodesk Suite for all their development needs.

Libraries

- Bullet or jBullet:
  jMonkey 3 has the libraries added to the SDK. Having this highly rated physics lib available as open source allows us to dive into the source and see how the underbelly works. Using this, we are able to do all physics calculations on Nvidia's CUDA cores instead of older physics engines that still requires CPU processing. It will still be decide between Bullet and jBullet seeing as both run on Windows as well as Android.
- Stack-Alloc:
  This library is used by Bullet and jBullet to simplify the amount of memory utilization and waist there of for the physics component of the game, seeing as it needs to run on Android devices as well.
- Spidermonkey:
  As a requirement we need to implement a backend system to which users connect to during gameplay. To achieve this we will be using Spidermonkey to handle all the client as well as server side programming and management. The reason for this library is that, it is also included as part of the jMonkey 3 SDK.
- Audio:
  Other possible choses for the audio component can be OpenAL or JOgg which is also supported in jMonkey 3.
- jME3-Core:
  This is the basis of jMonkey 3 containing most of the required libraries required:
  - MatDefs: Pre-created shaders.
  - ShaderLib: Libraries used in user created glsl shaders to display visual effect without having to recreate e.g. Bump, fog, hdr, lighting, math, shadows, water.
  - Animation: Handles all animation that you will require.
  - Audio: Controlling file loading, playing and setting.
  - Bounding: Adding bounding boxes for collision detecting.
  - Cinematic: If you require making cut scenes for your game.
  - Collision: Support for collisions detection and results there of.
  - Effect: Particle systems.
  - Loading: Handles all import and export.
  - Input: Retrieves user input.
  - Light: Lighting point variety.
  - Math: All mathematics required when implementing games.
  - Post: Post processing features.
  - Render: Camera and rendering options.
  - Scene: Scene management.
  - Shader: Controller of all OpenGL programs used.
  - Shadow: Filters, controllers and renders.
  - System: System monitoring
  - Textures: Support of all texture types and buffers required.
  - Util: Utilities to improve safety and overall usage.
  - Converters: Loading models.

  - Optimize: Optimization of storage and organisation of data e.g. geometry, trees, textures and triangles.
  - SaveGame: To maintain position and data.
  - Shader: Shader debugging and validation.

- jME3-effects:
  This adds extra means to improve the aesthetics of the game through:
  - Post processing: Bloom, Depth Of Field, FXAA, Fog, Gamma Correction, Light Scattering.
  - SSAO
  - Water

- jME3-niftygui:
  The user interface library that we will be using that is supported by Windows and Android.

- Other:
  Any other library natively supported in Java is accessible and usable during this project.

Resources

The resources that we will be using during this project will consist of: Textures, 3D / 2D Models, Music and Sound Effects. All of these need to be freeware and this may seem as a difficult task at first but there are various sources available:

- Open Game Art
- Archive 3d
- Texture Mate
- Blender Models
- Flash Kit
- Mayang Textures
- CG Textures