Project Ngeen

Supplementary Specification

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 3/20/2016 | 1.0 | Initial Commit | Daian Dragos |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Testability 4

2.5 Usability 4

3. Design Constraints 4

Supplementary Specification

# Introduction

There are two main possible ways this project can take. Either become open source or a proprietary project.

If it were to be open source, it would benefit from the community, and evolve at a rapid pace. It would then get it's income from plugins that the creator of it(who know it best) could make and that people would trust. Once the community is build, it should also be easy to advertise different things to it.

If it were to be proprietary, the plan would most likely be monthly payments for continuation of use. There would be different plans offering more as you pay more.

The system to be built will work firstly on desktops with all the features it supports. It will also run on devices, but only support release features, or some debug features if possible. The system is also intended to be run on browsers, that being if the proprietary choice is made for it, because such an option implies costs for servers, domains, etc.

The project will be written in Java language, and it will support all major systems, including Windows, Linux and Mac, and other devices such as Android, IOS and Windows Phone.

# Non-functional Requirements

The project is to be used to aid game development. This is important because game development takes a lot of time

The project is to be used to aid game development. This is important because game development can take a very long time until the game sees the light of release. Also, it does so that it is also easier to test and debug and scale the project later on.

The project was thought of after seeing other similar products that had different limitations.

One was the fact that the whole engine had to be ran on your own pc and it was after a while slow. That is, it had all the tools needed for development, but such tools could be split. For example, the tools needed for building are not needed for all the systems when working on Windows, etc.

This project is intended to be as light and fast as possible for development and to contain all the things one would need at a time.

As a response, this project is thought with the consideration that it will make production as fast as possible by making it so it doesn't have any recompilation time and that it can be ran from anywhere.

While true, the system doesn't offer that many more to the actual people using competing software, the tactics are to build also a small demo with it in order to offer people a glimpse of what the engine can do.

## Availability

The system is available to everyone by default, and if the non open source plan is chosen, paying users will receive extra features. That means new users will be able to test and evaluate for free the product for as long as they wish, but with some limitations included.

## Performance

The Entity Component System Design is slowing down a little the end result, so if you want your game to be as fast as possible, you should write it without any library or framework whatsoever. That being said, the tradeoff is worth it, for it makes the development time smaller, and organizes parts of your product by data and logic.

## Security

The project offers no security whatsoever, but other tools can be used to obstruficate your code.

## Testability

The project will be intensively tested in order to give the user an idea of what it should and should not do. The testing will be done after each release using JUnit. It will be done on all supported devices, in order to find out different limitations, if any.

## Usability

The system is to be used by anyone that is intending to develop a video game and needs a complete solution for rendering, physics and networking. It can also be used in conjunction with other libraries or frameworks.

# Design Constraints

The project is already to be build with Java for class platform reasons and libGDX, box2d and Bullet to aid into that and to minimize development time. Also, the Entity Component System Design Pattern was chosen and is defining part of the project, for it is the thing that helps users of it organize their code and write less code. It was chosen because it makes more sense to model the project with this design pattern than with other. That being said, every game can be modeled using this.

Java was chosen mainly because the Android device, which is a huge market right now, uses Java. Thus, it is easier to directly develop in Java and export to Android. Other devices or Operating Systems can also run Java, which also assures cross platform.