SOFTWARE REQUIREMENTS SPECIFICATION

for

GET
2d Game Engine Toolbox

Version 1.0 approved

Prepared by Kier Lindsay and Léo Abiguime

CMPT-376W

April 6, 2020

Contents

Revision History

Name	Date	Reason For Changes	Version
Kier Lindsay	April 6, 2020	Inital Copy	1

1 Introduction

1.1 Purpose

This 2d game engine is designed to bridge a gap between existing graphics library and large complete 3d game engines. It will extend the capabilities of graphics library and provide common systems and feature needed to develop a game without forcing the user into a large complex engine.

1.2 Document Conventions

This document follows the IEEE SRS template. It also makes use of the KISS mentality in order to make things easy to parse.

1.3 Intended Audience and Reading Suggestions

This document is intended for the developers who will be implementing the game engine. We would also like to consider users of the engine as this document may offer a high level overview of what the product offers and its design philosophy.

1.4 Project Scope

This project is meant to complement an existing graphics library providing the core functionalists all games require and tools for developers to use. It is meant to provide a more advanced and highly customization starting point for game developers who want to have full control over there game. It is also meant to be modulate so that developers are not forced into using our designs and have the freedom to incorporate as many or as few of the features we provide.

1.5 References

SFML IMGUI SDL2 RayLib

Minimal programing guides.

others as we write them or references to documentation for implementation

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information

so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

2 Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

2.2 Product Functions

Engine Timing - Setups at maintian a core game loop Allows users to use real dt or fix it so each tick is a consistent time. Allows users to set tickrate and framerate independently Allows users to hook into update and render calls

States - Manage switching between game states Allow users to Bind a number of game states to an engine each with its own update and render functions Allow users to toggle states.

Assets - Manages the loading and management of game assets such as fonts and textures. Supports Fonts Supports Sprites Supports Textures Supports Animations

Physics (Optional) - Incorporates Physics Module Into update Camera (Optional) - Incorporates Camera Module and camera control function into engine ie setting camera to fallow a Body in the engine.

Physics Camera UI - Adds the ability to make and control UI's Util - extra utility functions that extend GL or offer useful functions Extend Math for GL's vectors. Add Extra bodys and objects such as partical systems

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

2.3 User Classes and Characteristics

Full users - these users will use this software as their starting point and build there game around our components and tools.

Partial users - These users will user the core graphics library as their starting point and may incorporate some components of our software into there project.

2.4 Operating Environment

Our Target environment for this project would be to extend SFML and we should expect the same environment that sfml does.
OS: Linux, OSX, Windows
Depandancys: SFML, OpenGL, Imgui
Language Target: C++

2.5 Design and Implementation Constraints

SFML does not support 3d inherently so it should be expected that the protect also does not support 3d.

2.6 User Documentation

API Reference
Setup Guide
Basic examples

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3 External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4 System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1: REQ-2:

4.2 System Feature 2 (and so on)

5 Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6 Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

6.1 Appendix A: Glossary

computer is a programmable machine that receives input, stores and manipulates data, and provides output in a useful format. 12

graphics library is a library the provides a window and functions to draw to the window. Examples of graphics libraries are SFML and SDL2. 5, 12

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

6.2 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship a single project in each SRS.>

6.3 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

6.4 Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>