Software Requirements Specification

Berserk Engine Rendering System

Author: Orachyov Egor
Created: 10.03.2019 14:00
Last edited: 10.03.2019 15:01

Table of Contents

Introduction

This document represent software requirements specifications for Berserk Engine subsystem named Render System (or Rendering Engine).

Purpose

The purpose of this document is to give a detailed description of the requirements for the «Berserk Engine Rendering System» module. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primary intended to be exhaustive source of information for developer for programing first version of the system and for teacher (mentor), who will accept that work.

Scope

«Berserk Engine» is software game development kit (lib) designed for creating high performance 3D graphics applications with basic physics and audio effects. It should be used by programmers and software engineers, who want to develop their own game application.

This software kit is supposed to be free to download, share and use. Also, it is open source project with shared data base, which could be downloaded from GitHub.

«Rendering System» of the Berserk Engine is a Graphics Core and the most powerful and important part of the engine. It should provide features for rendering objects and applying effects (post process effects) in soft real time mode.

Definitions, acronyms, and abbreviations

Table 1 - Definitions

Term	**Definition**
Engine / Game Engine	Software development kit - fully featured, developed as one or several .h include and .c/.cpp library files project, which provides functionality for creating graphics/physics/audio application (games, presentations, simulators, etc.).
Renderer / Rendering System	Game Engine subsystem, which is responsible for generating animated and rasterized in any way 3D/2D graphics.
User	Programmer, software engineer, in this context, user of the Engine framework and libraries.
Pipeline	Graphics generation and rasterization pipeline - the number of stages, steps, needed to generate single image with chosen quality, effects usage, time limitation and format.
Platform	Named device driver, with provides interface for access functions of hardware accelerated graphics processing unit.

OpenGL	Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.
XML	Format of creating text (human readable) files based on xml specification.
Texture	1, 2, 3 dimensional array of integer, floating point, short, signed, unsigned data, which could be interpreted as an image or other special purpose/format data. Could be created/loaded from image data, stored in common formats, such as png, bmp, jpeg, etc.
Material	Data structure, which stores properties of the abstract surface, which defines how rendered mesh will actually look in the rasterized/generated image.
Mesh	Raw vertex data: array vertices with indices, where each vertex stores data about some 2D/3D dimensional point. This data could be filled with: position, normal, tangent, bitangent, texture data of the point.
Model	An abstraction for representing 3D/2D dimensional objects in Rendering System. It stores mesh(s) and associated with them materials to perform correct representation of the model in rendered images.

References

[1] «Game Engine Architecture», 2nd ed. by Jason Gregory

[2] «Learn OpenGL», by Joey de Vries

[3] «Real-Time Rendering», 4rd ed. by Tomas Akenine-Möller, Eric Haines and Naty Hoffman

[4] «The OpenGL Shading Language», 4th ed. by David Wolff

[5] «Real-Time Graphics Rendering Engine», 1st ed. By Hujun Bao Wei Hua

[6] «ShaderX3: Advanced Rendering with DirectX and OpenGL», by Wolfgang Engel

Overview

The remainder of this document includes three chapters. The second one provides an overview of the system functionality and system interaction with other systems. Chapter also mentions the system constraints and assumptions about the product.

The third chapter provides the requirements specification in detailed terms and a description of the different system interfaces. Different specification techniques are used in order to specify the requirements more precisely for different audiences.

The fourth chapter deals with the prioritization of the requirements. It includes a motivation for the chosen prioritization methods and discusses why other alternatives were not chosen.

Overall description

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it.

Product perspective

Product functions

User characteristics

Constraints

Assumptions and dependencies

Apportioning of requirements