ZED

Software Requirements Specification

Open Game Developers

[Version: 0.0.0.1] [Date: 10|10|2012]

Change Log

Version	Author	Changes
0.0.0.0	Rico	Initial revision Created document structure
0.0.0.1	Rico	Overall Description section has had its preliminary sweep More detail added to the Introduction section Changed Product Functions to Product Features in the Overall Description section

Table of Contents

Preface

This document will describe the ZED engine in terms of what the engine will be implementing in terms of requirements.

Early class names shall be described here.

Exact API information will not be described.

ZED

Introduction

Purpose

ZED is intended to provide a variety of components to aid in game development. Components which are supported by ZED are as follows:

- System Services
- 2D and 3D Mathematics
- 2D and 3D Graphics Rendering
- Data Types
- Resource Management
- Asset Handling
- 3D Animation
- Audio
- Input
- Networking
- Artificial Intelligence
- Physics
- Scripting

While these functions are provided by ZED, they are intended to be extended by the developer for their specific development needs.

ZED will be implemented on a wide variety of hardware. As not all hardware will be capable of accomplishing the same task, the API will need to accommodate these lesser hardware platforms. In addition to hardware limitations, some hardware will be more esoteric and will have to be handled appropriately.

Abstracting away all known APIs from the developer is the ultimate goal of ZED. As the engine is intended to be used by programmers who are not working on lower-level details such as which graphics API to use or how to correctly apply a texture to a mesh.

Scope

As there are a wide variety of components involved in the creation of the ZED engine, the scope of each will expand as needed. In the initial version of ZED, the API will offer a minimal set of functionality for developers to extend as they see fit. ZED will be available for Linux, Windows, Xbox, Pandora, and BlackBerry PlayBook initially.

Definitions, Acronyms, Initialisms, and Abbreviations

Definitions

Engine An Engine is a conglomeration of software libraries which combine to provide a common layer of abstraction for accessing a computer's resources.

Acronyms

ZED – The engine which this document is intended to put forth the requirement effort involved

Initialisms

API	Application Programming Interface
STL	Standard Template Library

Abbreviations

Sockets – Windows or Berkeley Sockets API
WinSock – Windows Sockets API

System Overview

ZED is intended to be as lightweight as possible and as optimised for each platform it supports. Modularity is important for the future expansion of the engine, thus there will be a heavy focus on keeping the system well documented and open for developers.

Depicted from a coarse view; see Figure 1.1.

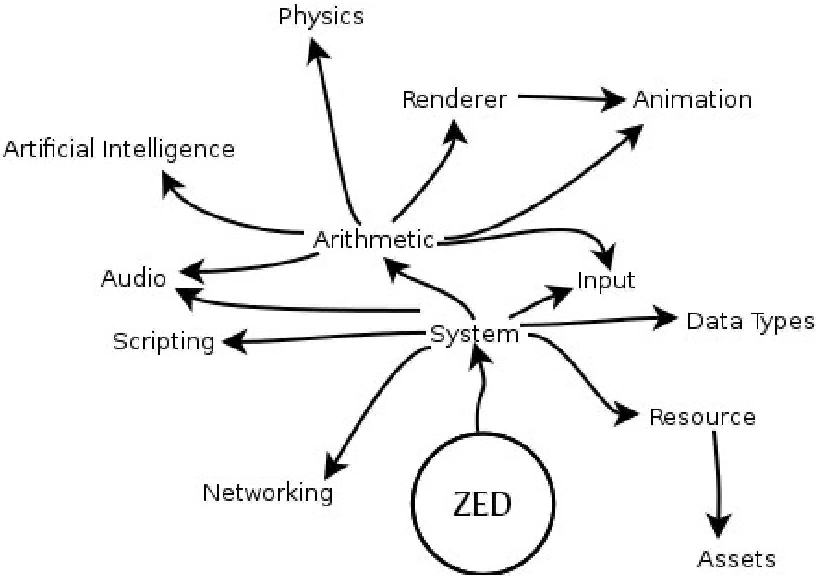

Figure 1 - An overview of the main engine components and their dependencies

Following are a series of narratives on each system provided by ZED:

System Services

Providing memory management, processor information, memory information, multi-threading, and platform information.

2D and 3D Mathematics

Vectors, Matrices, Quaternions, Rays, Lines, Planes, Polygons, Axis-Aligned Bounding Boxes, Orientated Bounding Boxes, Bounding Spheres, Bounding Ellipsoids, and Bounding Volumes.

2D and 3D Graphics Rendering

Supporting Direct3D, OpenGL and OpenGL | ES to apply shaders to primitives and the screen, render primitives, and applying textures to primitives.

Data Types

Intended to replace the STL; the data types cover arrays, lists, queues, maps, and sets. Mainly provided to ensure that there is a set of standard data types available on all platforms with a consistent interface.

Resource Management

Loading content and avoiding loading the same piece more than once is the aim of good resource management. Pushing resources out of memory to make way for new resources and keeping count of how many times a resource is referenced will aid in deciding which resources have to be pushed out and which can stay. Flagging resources as “resident” can be useful for resources which need to always be present, such as pause menus and a character's arms for a first-person shooter.

Asset Handling

Default assets supported by ZED. Model, shader, animation, audio, texture, script.

3D Animation

Skeletal animtion.

Storage

Reading/Writing files and directories. Listing files and directories.

Audio

Multi-channel audio, playback controls, playback audio manipulation.

Input

Keyboard, Mouse, Gamepad, Motion Controller.

Networking

Supporting Modem and Ethernet devices. Providing Multicasting, Latency Simulation, Packet Handling, IPv4, IPv6, Peer-to-Peer, and Client-Server.

Artificial Intelligence

Finite State Machines, Behaviour Trees, Pathfinding, Navigation Meshes and Waypoints, Steering Behaviours, and Decision Trees.

Physics

Rag Dolls, Collision, Particles, Soft and Hard Bodies.

Scripting

ZED Scripting Language, compiler, hot-swappable code.

References

ZED

Overall Description

Product Perspective

ZED is a set of software libraries for abstracting low-level implementation details of a game. To lessen as many dependencies as possible and maintain a consistent level of cross-platform behaviour, libraries from the Standard C Library and the C++ Standard Template Library are duplicated to ensure developers will not have to use each platform's specific implementation of both sets of standard functions and template classes. Abstracting the underlying hardware will be the main objective of the engine, while also providing access for developers whom are using the engine to lessen development time, yet still wish to access the lower-level functionality.

Product Features

Following is a system-by-system breakdown of the features

System Services

- Provides information about the processor in the system:
    - SIMD Support
    - CPU type (x86, ARM, SPARC, SH)
    - Cache count and size
    - Physical and logical processor count
    - Vendor
    - Model
    - Frequency
    - Esoteric hardware information, such as AMD and Intel Virtualisation types.
- Multi-threading functions and primitives for the processor
- Memory information:
    - Available/Used/In-use memory statistics
    - Vendor *
    - Model *
    - Frequency *
    - Individual module capacity *
    - Type (DDR, SDRAM) *
- Memory management
- Operating System information:
    - Version

    - ○ Vendor

2D and 3D Mathematics

- Vector [Common]
    - Addition
    - Subtraction
    - Multiplication (Scalar and Vector)
    - Dot Product
    - Normalise
    - Direction and Direction Squared
    - Magnitude and Magnitude Squared
    - Vector-*n*D-by-matrix-*n*x*n* multiplication (Row vector by matrix multiplication)
    - Matrix-*n*x*n*-by-Vector-*n*D multiplication (Matrix by column vector multiplication)
- Vector 2D
    - Perpendicular Product
    - Perpendicular Dot Product
- Vector 3D
    - Cross Product

2D and 3D Graphics Rendering

Data Types

Resource Management

Asset Handling

3D Animation

Storage

Audio

Input

Networking

Artificial Intelligence

Physics

Scripting

* = May require high privilege level

User Characteristics

Constraints

Assumptions and Dependencies

Dependencies

ZED requires the Microsoft Visual C++ Compiler for the following:

Windows 2000	Microsoft Visual C++ 2003
Windows XP	Microsoft Visual C++ 2003, 2005, 2008, or 2010
Windows 7	Microsoft Visual C++ 2005, 2008 or 2010
Xbox	Microsoft Visual C++ 2003

For GNU/Linux x86 systems, any version of GCC 4.4 and higher will be tested against.

Windows and GNU/Linux will use the OpenGL and OpenAL libraries.

Pandora development libraries and compiler is correctly installed.

The Microsoft Xbox, SEGA Dreamcast and SEGA Saturn will require their respective SDKs.

The Nintendo GameCube and Nintendo DS will be built against their respective devkitPro SDKs.

The Open2X Toolchain will be required for GP2X development.

BlackBerry's Native SDK will be required for BlackBerry PlayBook development. PlayBook OS 2.1 and higher will be targeted, deprecating unsupported versions of the NDK as necessasry.

Assumptions

Developers will have adequate knowledge of the systems they are targeting and the compilers being used for development. Target hardware is available to the developer and any necessary SDKs outlined in the dependencies section above are correctly installed.

Specific Requirements

External Interface Requirements

User Interfaces

Hardware Interfaces

Software Interfaces

Communication Interfaces

Functional Requirements

Performance Requirements

Standards

Hardware Limitations

Design Constraints

Availability

Security

Maintainability

Other Requirements