Platform Perils

System Architecture

Steven Palmer
⟨palmes4⟩
Chao Ye
⟨yec6⟩

April 26, 2016

Contents

List of Tables

List of Figures

Revision History

Date	**Version**	**Notes**
January 10, 2015	1.0	Created document
January 11, 2015	1.1	Major additions to all sections
January 11, 2015	1.2	rev0 final version
April 25, 2015	1.3	rev1 final version

1 Introduction

1.1 Overview

This document is provides a synopsis of our proposed design considerations that will be used to implement our game. The document is split into two parts: the system architecture which gives a more general overview of the design goal and intended module interactions, as well as the detailed design which covers the intended implementation in greater detail.

1.2 Document Template

The Parnas template of the Module Guide and Module Interface Specification was followed in creating this document. Adherence to this template was not 100%, and in some cases the provided marking scheme was used as a supplement to modify the template.

2 Design Principle

The Model-View-Controller architectural pattern will be used as the main principle of design for this project. As such the main consideration of the design will be to fully separate the operation of the game in terms of user control, the audiovisual output, and the game model. Using this model will allow for efficient implementation and testing of functional requirements because it will enable testing of the core game code in isolation from the input and output mechanisms (the majority of functional requirements relate to the game code).

3 Anticipated Changes

3.1 Likely Changes

There are no likely changes to the game at this point.

3.2 Unlikely Changes

The following changes are unlikely to occur:

UC1 The way sound files are loaded and stored is unlikely to change.

UC2 The way object files are loaded and stored as GPU data is unlikely to change.

UC3 The way shader files are loaded and stored is unlikely to change.

UC4 The game controller is unlikely to change.

UC5 The object hierarchy is unlikely to change.

UC6 The object rendering function is unlikely to change.

4 Module Decomposition

The decomposition of the project into modules with respect to the model-view-controller design principle is given in the following subsections.

4.1 Modules

The modules that were used to implement the game are given in Table 1 and Table 2.

Table 1: List of Modules Part 1

	Module	Function	System
1.	Game	Links user input to game code; main game loop	Controller
2.	ObjGPUData	Loads and stores object gpu data	Model
3.	ObjGPUDataStore	Stores collection of loaded ObjGPUData	Model
4.	Obj	Stores game object information; base for all objects	Model
5.	PhysicsObject	Subclass of Obj for physics based objects; base for all physics objects	Model
6.	StaticObject	Subclass of PhysicsObject for static objects; base for all static objects	Model
7.	Surface	Subclass of StaticObject for surfaces that the hero can stand on; base for all surface objects	Model
8.	Platform	Platform object; subclass of Surface	Model
9.	Wall	Wall object; subclass of Surface	Model
10.	Ramp	Ramp object; subclass of Surface	Model
11.	Goal	Goal object; subclass of Surface	Model
12.	KinematicObject	Subclass of PhysicsObject for kinematic objects; base for all kinematic objects	Model
13.	Spikes	Spike hazard; subclass of KinematicObject	Model
14.	Spear	Spear hazard; subclass of KinematicObject	Model

Table 2: **List of Modules Part 2**

	Module	**Function**	**System**
15.	MovingPlatform	Moving platform object; subclass of KinematicObject	Model
16.	DynamicObject	Subclass of PhysicObject for dynamic objects; base for all dynamic objects	Model
17.	Hero	Hero object for playable character; subclass of DynamicObject	Model
18.	Boulder	Boulder hazard; subclass of DynamicObject	Model
19.	StandardObject	Subclass of Obj for non-physics objects; base for all non-physics objects	Model
20.	Skybox	Skybox object for background; subclass of StandardObject	Model
21.	Arch	Arch object; subclass of StandardObject	Model
22.	Environment	Core of the game; handles gamestate; abstract class	Model
23.	Menu	Implementation of Environment for the menu system	Model
24.	Stage	Implementation of Environment for the stages	Model
25.	StageLoader	Loads and parses level scripts	Model
26.	Camera	Stores view matrices	Model
27.	Sound	Loads and stores sounds	Model
28.	SoundStore	Stores collection of loaded sounds	Model
29.	Shader	Loads and stores shaders	Model
30.	ShaderStore	Stores collection of loaded shaders	Model
	Hardware	Outputs audiovisual	View

4.2 Class Hierarchy Diagram

All class hierarchy diagrams are contained in the MIS/MID document.

5 Traceability

5.1 Requirements

A traceability matrix showing the correspondence between requirements and modules is given in Table 3.

5.2 Changes

There are no anticipated changes at this time.

Table 3: Requirements Traceability

Requirement	Module(s)
1	23
2	23
3	23, 24, 25
4	23, 24
5	1, 17, 24
6	1, 17, 24
7	1, 17, 24
8	1, 17, 24
9	5, 16, 17
10	7, 8, 9, 10
11	N/A
12	1, 24, 26
13	1, 24, 26
14	4, 5, 6, 7, 8, 9, 10, 11, 12
15	4, 6, 7, 9
16	4, 5, 12, 13, 14, 16, 17, 18
17	17, 24
18	11, 17, 24
19	23
20	2, 3, 29, 30
21	N/A
22	2, 3, 29, 30
23	27, 28
24	N/A
25	N/A
26	N/A
27	1, 2, 3, 29, 30
28	N/A
29	23