Physics-Based Chipmunk2D Game

System Architecture

Steven Palmer
⟨palmes4⟩
Emaad Fazal
⟨fazale⟩
Chao Ye
⟨yec6⟩

January 11, 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

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

The following changes are likely to occur as the project proceeds:

AC1 The number of object classes will likely be expanded. Currently the game objects (characters, traps, platforms, etc.) are represented as generic static, dynamic, and kinematic object classes. Further class specialization may be useful.

AC2 Changes to the way in which levels are implemented in the code are likely.

AC3 Changes to the way objects interact are likely.

AC4 Additional controls will likely be added.

AC5 Changes to the scoring system are likely.

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.

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 will be used to implement the game are given in Table 1.

4.2 Uses Diagram

A uses diagram is given in Figure 1 to show how the modules will interact. This diagram has been partitioned to show how the modules fit the model-view-controller design principle.

Table 1: Proposed Modules

Module	Function	Design Principle
Game	Links user input to game code	Controller
ObjGPUData	Loads and stores object gpu data	Model
Obj	Stores game object information	Model
Environment	Core of the game; handles gamestate	Model
Sound	Loads and stores sounds	Model
Shader	Loads and stores shaders	Model
Hardware	Outputs audiovisual	View

5 Traceability

5.1 Requirements

A traceability matrix showing the correspondence between requirements and modules is given in Table 2. Please note that this table will be completed once the requirements document has been updated since our game has changed drastically (before rev0 demonstration).

Table 2: Requirements Traceability

Requirement	Module
to be completed	

5.2 Changes

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

Table 3: Changes Traceability

Change	Module
AC1	Obj
AC2	Environment
AC3	Obj
AC4	Game, Environment
AC5	Environment

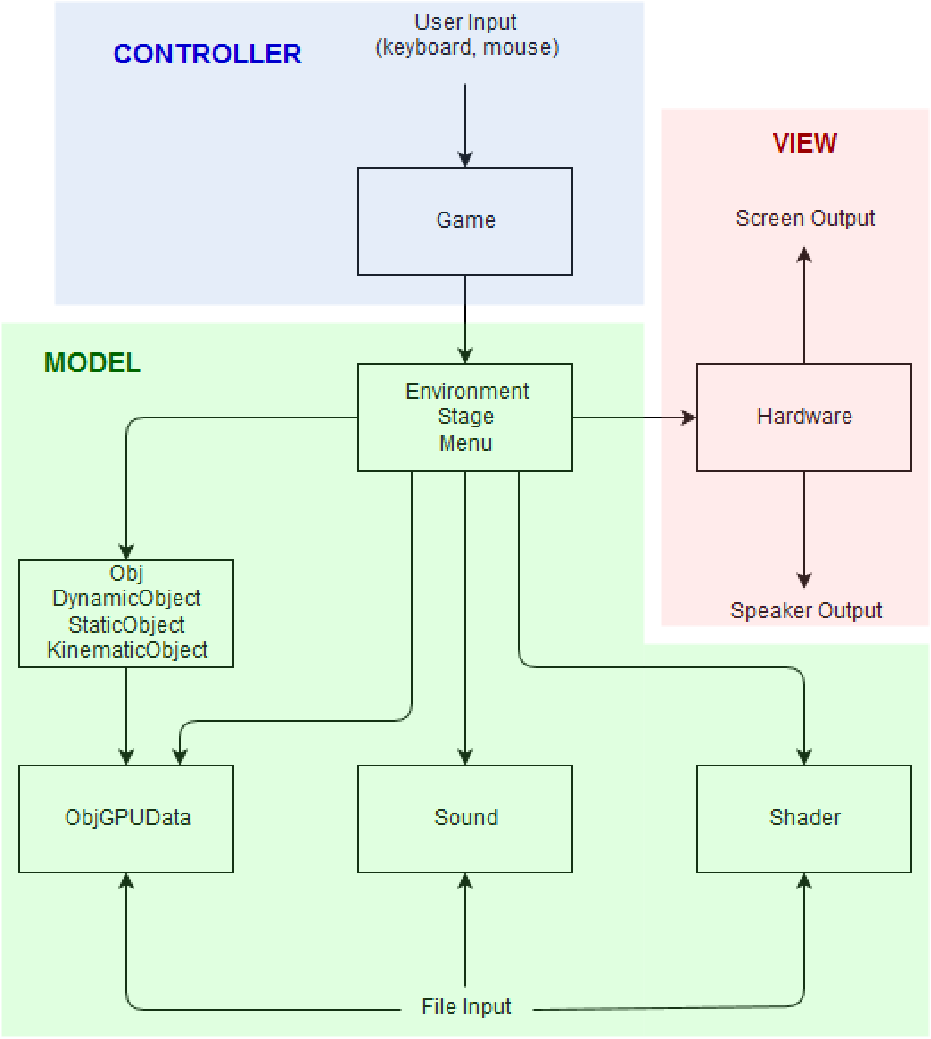

Figure 1: Uses diagram with MVC partitioning.