SE 3XA3: Design Document
Pong Invader

Team #10, Ben 10
Rehan Theiveehathasan `theivers`
Karnvir Bining `biningk`
Puru Jetly `jetlyp`

December 8, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
November 13, 2016	1.0	Initial changes

1 Introduction

This is the Design Documentation (DD-Rev1) for Ben 10s project Pong Invaders. This document outlines the design specifications as outlined by Dr. Spencer Smith. Complimentary documentation includes a Module Interface Specifications (MIS) document written with Javadoc. The motivation behind this document is to allow for the decomposition of systems that plan to scale into the future with updates, which allow for developers to concurrently develop components without risk of components not working with each other. The Pong Invader's project is concerned with creating a modular system design for the game that is well documented to allow for future upgrades to improve on current game mechanics. Currently the Pong Invader's system design principle is that of Model-View-Controller (MVC software design pattern) which is believed to be the best fit for allowing Pong Invader's to be modular. Potential readers of this document include:

- Designers/Developers: This document is a way to communicate how the program being designed should function in terms of coupling and meeting requirements. It allows developers to stay on the same page and be consistent with design assumptions. The breakdown of module interaction and traceability to requirements allows future developers to easily distinguish areas that can be optimized or changed.
- New Members: This documentation is an effective way to portray the project focus and aim to new members who need to be brought up to speed.

2 Anticipated and Unlikely Changes

This section outlines changes that may occur in the project. Anticipated changes are outlined in section 2.1 and unlikely changes in section 2.2.

2.1 Anticipated Changes

The following changes were identified throughout the course of project. These changes take minimal adjustment to project structure and design making them easy to implement within the given timeline constraints.

AC1: The hardware on which the software is running.

AC2: The format of the final output or display mechanism.

AC3: Changing win condition.

AC4: Additional Functionality from game menu.

AC5: The operating system the game will run with.

2.2 Unlikely Changes

The following changes were identified as requiring large adjustments to the project code and potentially requiring changes to several other components to be correctly implemented.

UC1: Input/Output devices (Input: keyboard and mouse, Output: Screen)

UC2: The Collision physics

3 Module Hierarchy

M1: Hardware-Hiding Module

M2: Input Format Module

M3: Output Module

M4: Controller Module

M5: Physics Module

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Format Module Output Module Controller Module
Software Decision Module	Physics Module

Table 2: Module Hierarchy

4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

5 Module Decomposition

Module decomposition is the process of creating a set of submodules that adhere to the principle of "information hiding." This principle of information hiding will be used as a fundamental concept for Pong Invader's object oriented development. Even though Pong Invader's is a stand-alone project without any foreseeable scalability, it will still be extremely beneficial to achieve modular design through module decomposition as Pong Invader's is created in phases.

5.1 M1: Hardware Hiding Modules (Pointing Devices, Keyboard, and Displays

Secrets: Data structures and algorithms that are implemented by device drivers, and hardware.

Services: Allows the system to access and handle input/output from the hardware and software.

Implemented By: The Java Virtual Machine, and the respective users' operating system.

5.2 M2: Input

Secrets: Data structures and data type of inputs.

Services: This module handles user keyboard input data, and then sends the input data to the correct modules to allow for event processing.

Implemented By: KeyInput

5.3 M3: Output

Secrets: Data structures of what is displayed to the screen.

Services: This module handles outputting the game state produced by all the following sub modules. The submodules are all subject to change if there are changes to the software requirements specifications.

Implemented By: GameObject, Bullet, Alien, Player

5.3.1 M4: GameObject

Secrets: Data structure concerning game states, arrays and symbolic parameters for position, and velocity.

Services: Handles game state, and delegates what to be displayed and when.

5.3.2 M5: Alien

Secrets: Data structure of alien object.

Services Sends alien data of velocity, position, and description of what to rasterize to GameObject.

5.3.3 M6: Bullet

Secrets: Data structure of bullet object.

Services: Sends bullet data of velocity, position, and description of what to rasterize to GameObject.

5.3.4 M7: Player

Secrets: Data structure of player object.

Services: Sends player data of velocity, position, and description of what to rasterize to GameObject.

5.4 M8: Controller

Secrets: Data structures and data type of inputs that are to be processed.

Services: This module handles all logic of the Pong Invader's system. All processed logic and mathematical representations of the game state are then passed to GameObject.

Implemented By: Collision

6 Traceability Matrix

Req.	Modules
R1	M1
R2	M2
R3	M8
R4	M5, M6, M7
R5	M4
R6	M3

Table 3: Trace Between Requirements and Modules

R1 : Hardware is capable of I/O handling.

R2 : Game reads input from user.

R3 : Processes and stores user input through a physics system.

R4 : Game keeps track of current display data.

R5 : Game keeps track of current game state.

R6 : Game outputs all relevant display data.

AC	Modules
AC1	M1
AC2	M3
AC3	M3, M4, M5
AC4	M4
AC5	M4

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

The hierarchy between modules that have been decomposed and mentioned throughout this document is as follows:

1. Hardware Modules (Keyboard)
2. Input (KeyInput)
3. Controller (Collision)
4. Alien / Bullet / Player
5. GameObject
6. Output
7. Hardware Modules (Display)

The uses hierarchy listed above is read as such: where 1.. 7 are numbered states in a directed acyclic graph and all numbered states are followed by the prior numbered state. Hence 1 to 2 to 3...(where "to" implies the next state) and so forth.

Figure 1: Use hierarchy among modules