Software Design Document for Automaton Demonstrating Advanced Movement

Prepared by:

Joseph Lipinski

Patrick Nwanah

Matthew Freihofer

Cory Cellucci

Sarah Kushner

Zachary Lopez

February 5, 2016
Version 1

Revision History

Date	Description	Version	Author
February 5, 2016	Created the document	1	All

Table of Contents

1 Introduction

1.1 Purpose

This document is to describe the implementation of the game: Automaton Demonstrating Advanced Movement, or A.D.A.M. for short. A.D.A.M. is a 2-D side-scrolling platformer with shooter elements.

1.2 Scope

The scope of A.D.A.M.'s design will encompass programming the base game including these elements: Physics, Player control, AI logic, and User Interface. A.D.A.M. will also import artwork and level design created by us outside of the base program.

2 Design Overview

2.1 Description of Problem

Often, people can become disinterested in the monotony of their daily lives and seek out the refuge of a fantasy world to avoid the problems and challenges that they face in real life. We seek to provide our users with a similar experience. Users will platform and shoot their way through levels in order to progress forward.

2.2 Technologies Used

Python
Pygame
PyTMX
Pyscroll
Visual Studio
Windows OS

2.3 System Architecture

Figure 1 depicts the high-level system architecture.

Figure 1

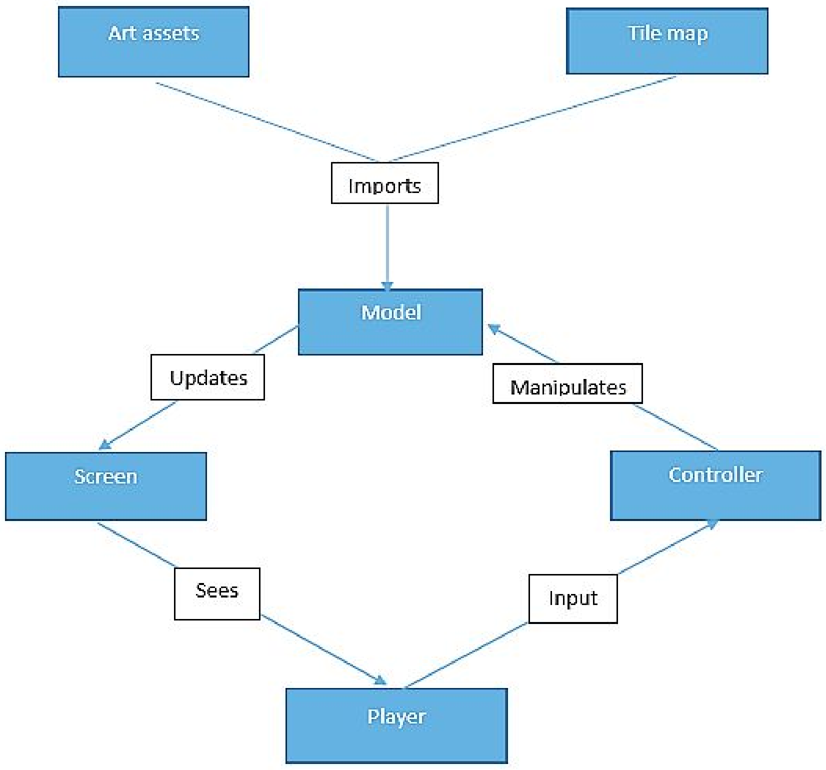

Figure 1: A.D.A.M. Architecture

2.4 System Operation

Figure 2 is the sequence of events that occur during execution of the game A.D.A.M.

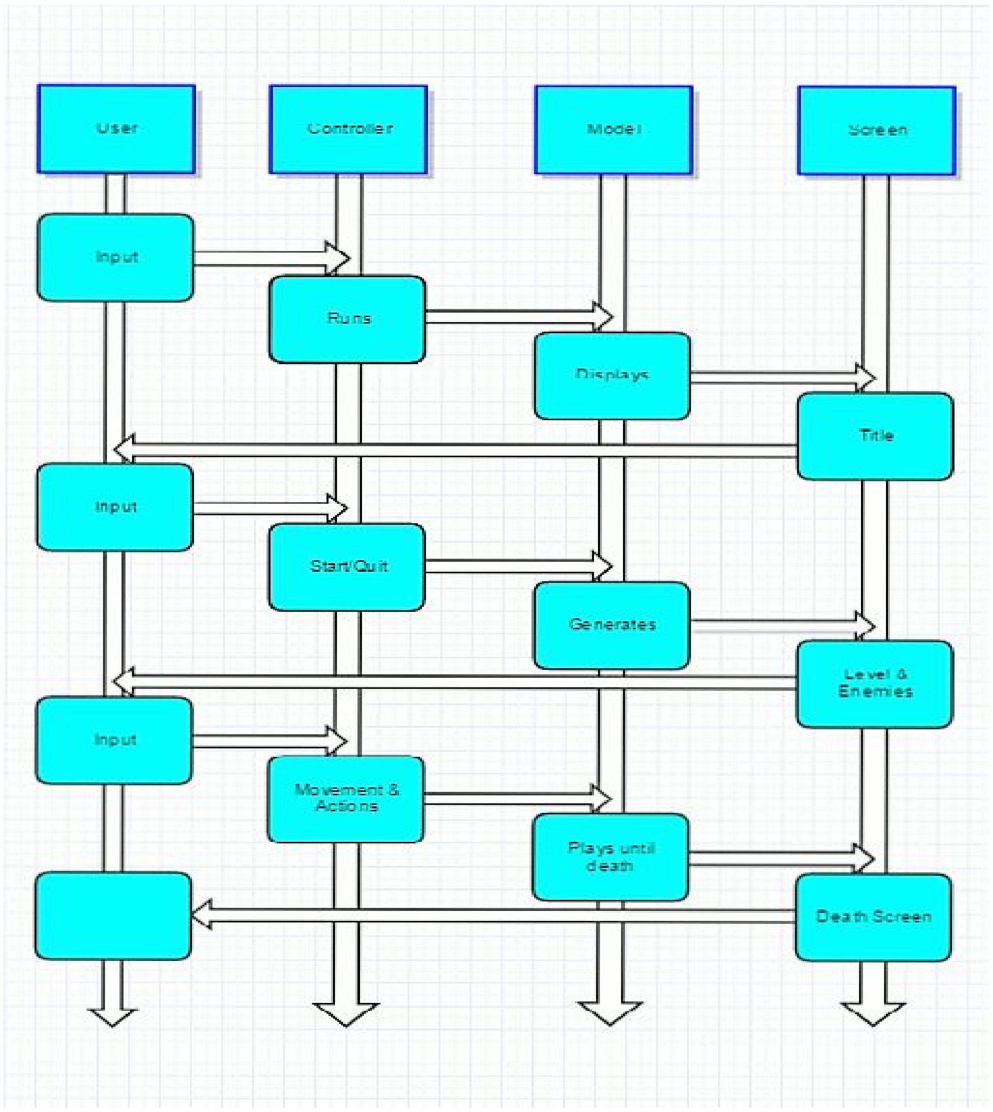

Figure 2: A.D.A.M. Sequence Diagram

3 Requirements Traceability

Requirement	Description	Design Reference
R1	Player	Figure 1,2,3,5
R2	Physics Engine	Figure 3
R3	Enemies	Section 4
R4	Level Design	Figure 4,8
R5	User Interface	Figure 4
R6	Art Assets	Figure 5,6,7,8

4 Game UML

4.1 Game Class Diagram

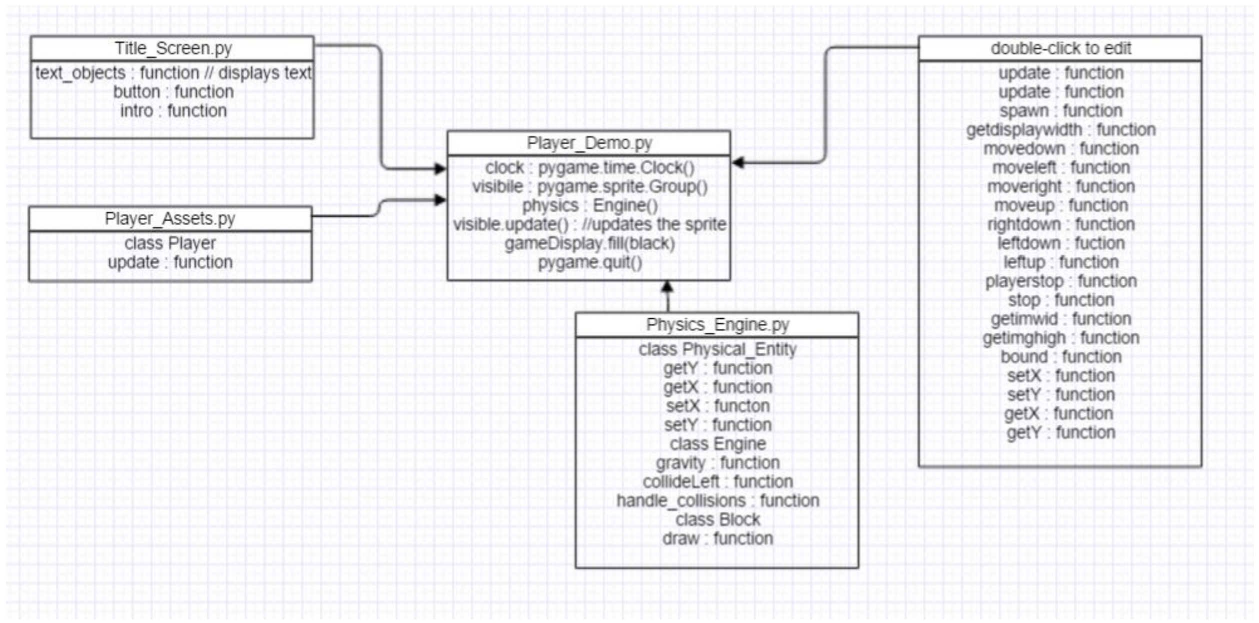

Figure 3: Class diagram for A.D.A.M.

4.2 Operational Overview

The Player_Demo is the main focus of our model. This is where the current state of the game will be stored. The other classes are imported into the Player_Demo so that their functions and capabilities can be accessed. The Player_Demo will function as the main interfacing component to the player

- The player, and enemies will be treated as physical Objects
- Collisions should be detected and handled by the physics engine
- All of the classes should flow together so that the player's experience is continuous

5 User Interface

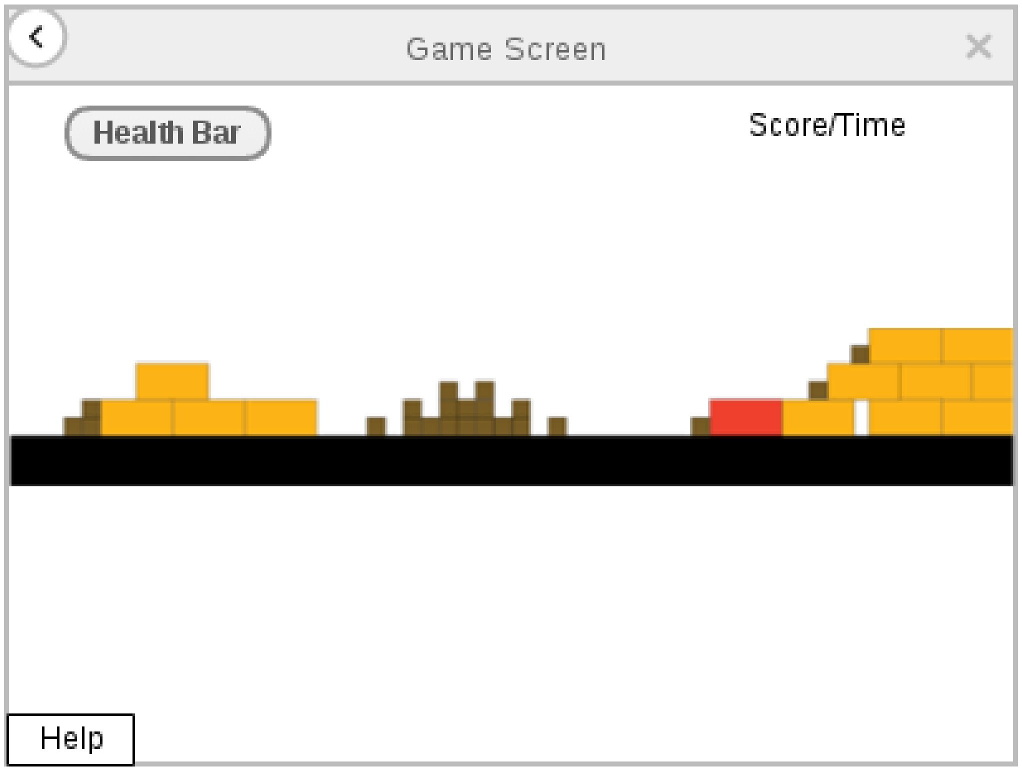

Figure 4: Mock-up for the main UI in A.D.A.M.

The main game playing screen will have a help button, shown at the bottom left, which brings up keyboard controls and other instructions for the game. The back button will bring the user to the main start screen. The user can navigate through the level using the correct keyboard controls. Their score, time remaining, and health will be displayed at the top, changing throughout the gameplay according to the users' performance.

6 Art

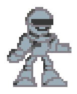
Figure 5: Main player standing

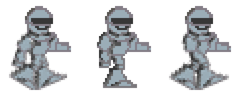
Figure 6: Main player walking

Figure 7: Main player shooting

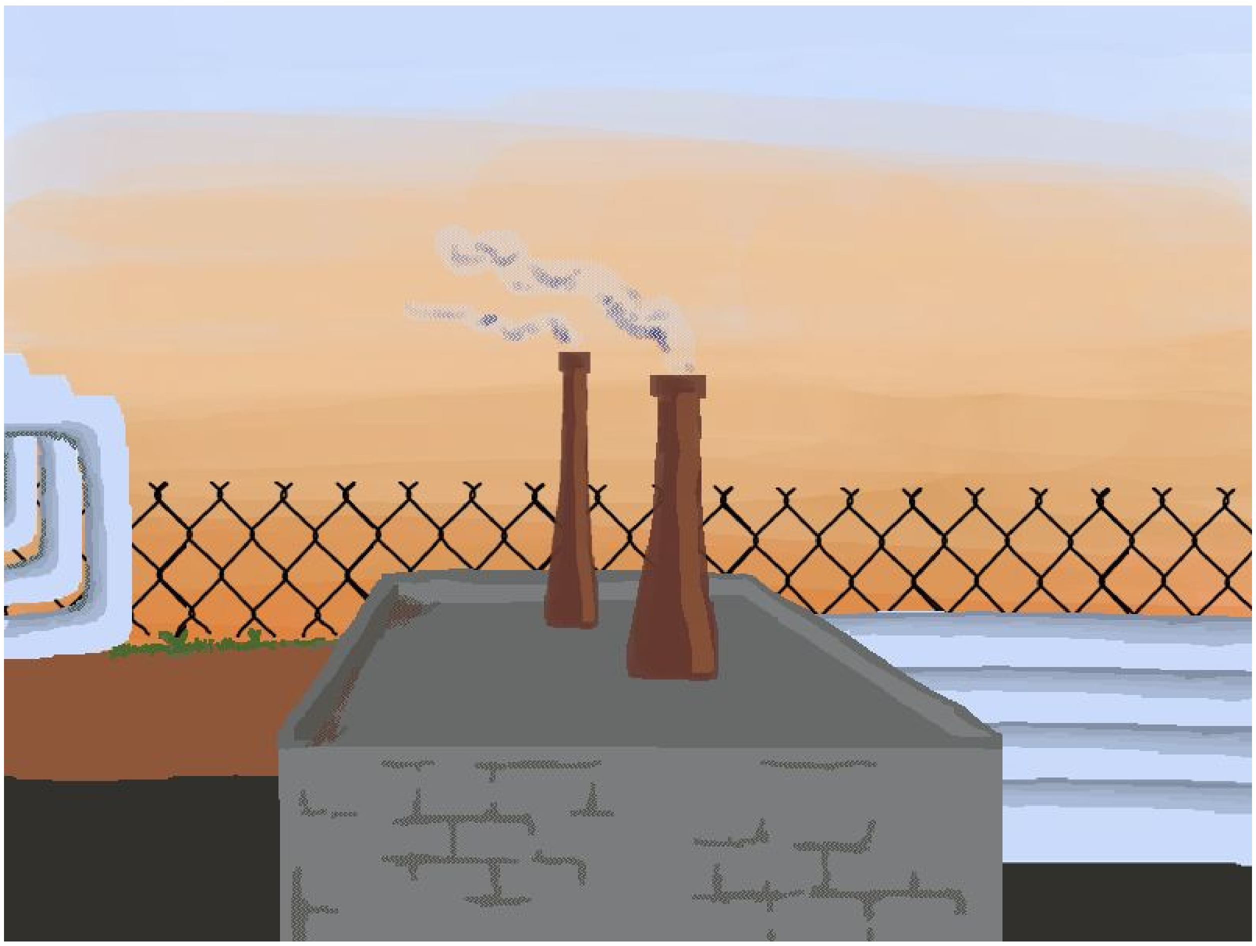
Figure 8: Factory level first draft art

7 References

[1] Pygame.org, "pygame", 2016. [Online]. Available: http://pygame.org/hifi.html. [Accessed: 10- Feb- 2016].

[2] Docs.python.org, "Overview — Python 3.3.6 documentation", 2016. [Online]. Available: https://docs.python.org/3.3/. [Accessed: 10- Feb- 2016].

[3] Pygame.org, "Tags", 2016. [Online]. Available: http://pygame.org/tags/tiled. [Accessed: 10- Feb- 2016].

[4] Gliffy, "Gliffy | Online Diagram and Flowchart Software", 2016. [Online]. Available: https://www.gliffy.com/. [Accessed: 10- Feb- 2016].