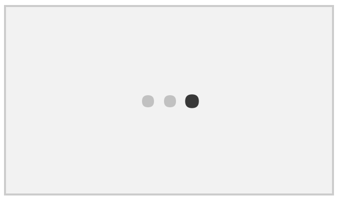
Design Document

CSCE 361 - Spring 2017 - Group 13

Noah Loos, Grahm Manley, Xiang Ma, Eric Tran

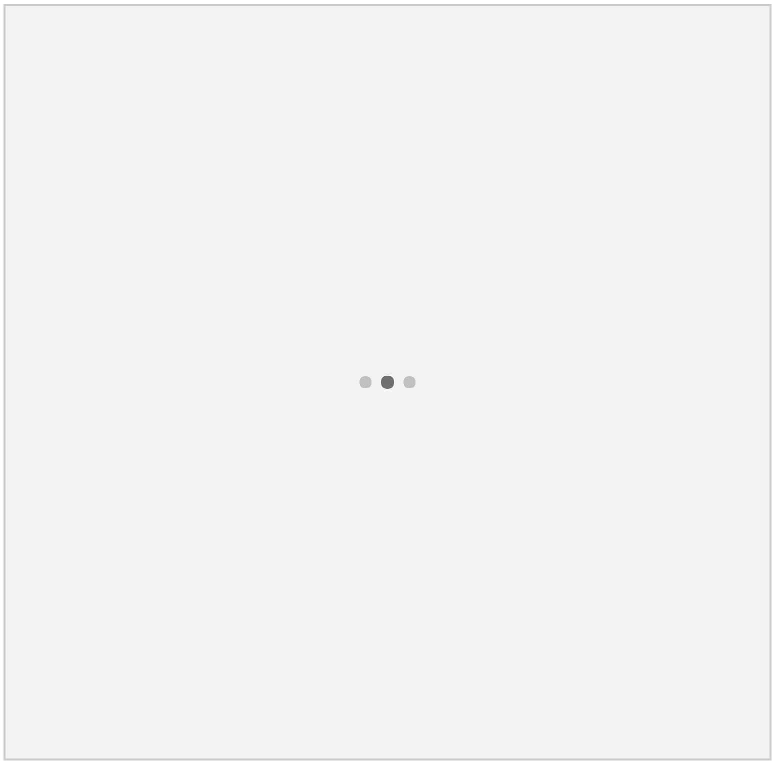
Super Ultra Mega Platformer



1. Introduction
The function of this document is to define Super Ultra Mega Platformer's systems regarding the layout of the structure. Additionally this document describes the connections between different modules within the system and the modules' interactions. It will give an overview of the design to implement the specifications outlined in the system requirements document.

2. Architecture

2.1 Introduction



The Model-View-Controller architecture is being used for this software because the architecture models how the user will send data via the controller to the view and model.

2.2 Modules

2.2.1 Controller Mapped - Keyboard Inputs
The controller module consists of a list keys that correspond to actions in the game. The module will listen for keystrokes and notify the model when a key has been pressed. The module will be responsible for the next steps.

2.2.2 View - User Graphical Interface
The view is primarily what the user will be seeing and interacting with. It is responsible for displaying data input from the model. The view provides a functional way for the user to make decisions within the structure developed.

2.2.3 Model - Unity®

The model in our game is handled by the Unity® engine. The engine will be responsible for handling logic and data that will render the game. Unity® compiles scripts with objects from the graphical user interface that allows a relationship between the model and the GUI. The model will also be responsible for the responses to keystrokes, like moving the player forward and deciding the response due to the player's location.

3. Class Diagrams

3.1 Overall Class Diagram(Please Refer To Attachment for Better Resolution)

3.2 Class Information

3.2.1 User Interface Package

This package contains elements of the User Interface that the user can view and interact with before accessing the stages of the game.

3.2.1.1 GUI

Gui provides dimensions to child classes and is the base for much of the User Interface package.

3.2.1.2 Button

Button provides a method of navigation and choice for the user between menus and the stage.

3.2.1.3 StageSelect

Main Menu of the game for which the user first sees. Buttons here leads to the ability screen. Lists stages that are currently unlocked. In agreement with usage of Unity®, a Unity Logo must be attached to the game.

3.2.1.4 AbilityScreen

Includes more buttons that chooses the abilities for the upcoming stage that the player can switch to. User can access the stage from here.

3.2.1.5 AbilityIcon

Provides an depiction of the ability and description of the ability for the user in the ability screen.

3.2.2 Stage Package

This package contains the elements that make up the stages that the character traverses through.

3.2.2.1 Stage

This is the class which aggregates all of the elements of the stage into a container.

3.2.2.2 Obstacle

This class is a superclass which all the obstacle types are derived from. The obstacles will be objects the player has to overcome within the stage. The class has the common functionality of an obstacle ID.

3.2.2.3 Pitfall

This class represents an area the character can fall down.

3.2.2.4 Portal

This class represents a portal which sends the player from one location in a stage to a different specified location.

3.2.2.5 Spike

Spike is an obstacle which kills the character on contact.

3.2.2.6 Ability Nullifier

Prevents the user from switching or using abilities in the defined field

3.2.2.7 Null-Field

Prevents the user from switching abilities.

3.2.2.8 Platform

Basic object that composes stages that the player character can walk or interact with.

3.2.2.9 MovingPlatform

MovingPlatform inherits platforms and moves on a predefined path within the stage.

3.2.2.10 Projectile

A projectile can either kill the player, sending to the beginning again, or push them away and hindering their progress.

3.2.3 PlayerCharacter

This Class is the main object the user will be associated with, and contains a variety of attributes, methods, and this class comprises the player character as a part of the stage. However this Class receives abilities from the ability screen as defined by the stage.

3.2.4 Keys

Keys are mapped to different actions the PlayerCharacter can perform such as move left, right, and jump.

3.2.4.1 AbilityKeys

Inherits Keys class and contains the mapped keys for the abilities that are different from default controls.

3.2.5 Ability Package

Ability Package contains the Abilities SuperClass, and provides an overall design about what abilities will be included in the game in the first iteration. All abilities will have cooldowns to box in the dynamics of the game.

3.2.5.1 Default

Default ability are already included with the player and are always available. Includes walking and jumping.

3.2.5.2 Blink

This ability class provides the player to cross a short span of space at near instantaneous speeds. However is stopped upon hitting a platform.

3.2.5.3 Grapple

Releases a rope that attaches to obstacles and allows the user to retract or climb up the rope and swing with it.

3.2.5.4 Bounce

Increases the player Elasticity which decreases the momentum the player loses upon hitting a surface. May protect from certain obstacles.

3.2.5.5 Sprint

Allows the character to simply move faster.

3.2.5.6 Wall Jump

Character is able to jump off of walls based on if the platform allows wall jumping.

3.2.5.7 Double Jump

Character is allowed to jump a second time before falling back to the ground.

3.2.5.8 Wall Climb

Wall Climb allows the character to walk up walls for a certain amount of time before being forced to fall.

3.2.5.9 Light
Character is about to float and fall slower than normal. May have interactions with other stage elements.

3.2.5.10 Heavy
Character will resist changes to their momentum aside from air resistance and inputs from the player.

3.2 Agile Phase
The design outlined in this document is intended for the original phase of the implementation. Upon entering the agile phase, some additional design elements are to be added. Some of the planned elements are additional stages, stage appearance enhancements, and improved character animations. The extra stages would not require design changes, while the stage appearance and character animations may. The stage appearance enhancements would require modifications to the and would require modifications to the platform class and the animations would require animation classes attached to different actions like the abilities and obstacles.