Michael Remley
Skye Ozga

MP4 Write-Up

Project Overview

For our project, we decided to make a simple side-scrolling platformer game in a minimalist art style. The player's avatar is a red square that can jump, wall-jump, and hang onto the ceiling to reach the end of the stages posed to them. If they miss a jump, they must start over.

Results

We created a game that does not take very long to play; the player is presented with a random combination of three, prebuilt stages in a row each time they run the game. The game automatically moves the screen to the right, so the player must match this pace. Failing to keep up or falling off ends the game. Reaching the end of the three stages wins the game.

There are eight stages from which the game selects to build the whole level. There are three basic ones that demonstrate jumping, wall jumping, and ceiling grabbing. The rest are more challenging by requiring the player to use combinations of the game mechanics. Figure 1 shows one of these stages.

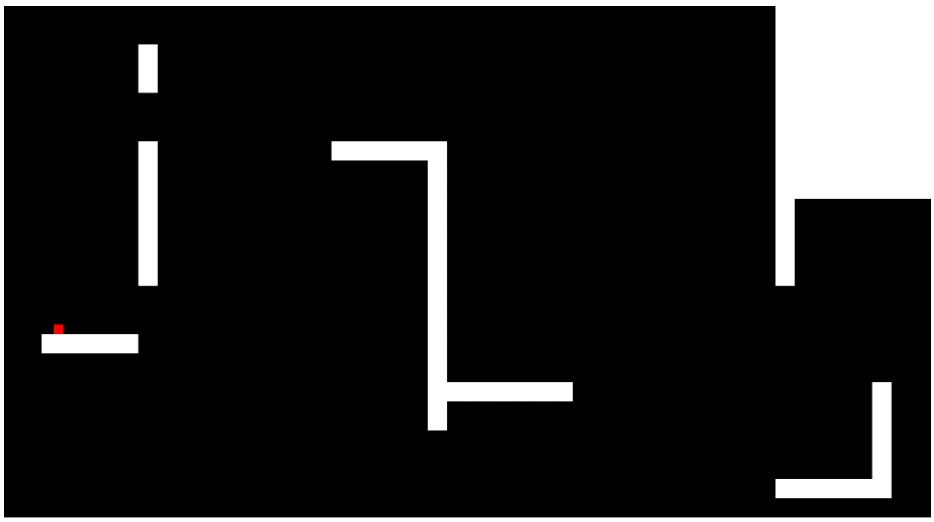


Figure 1. One of the stages created for our game. The player's avatar is seen on the left as a red square.

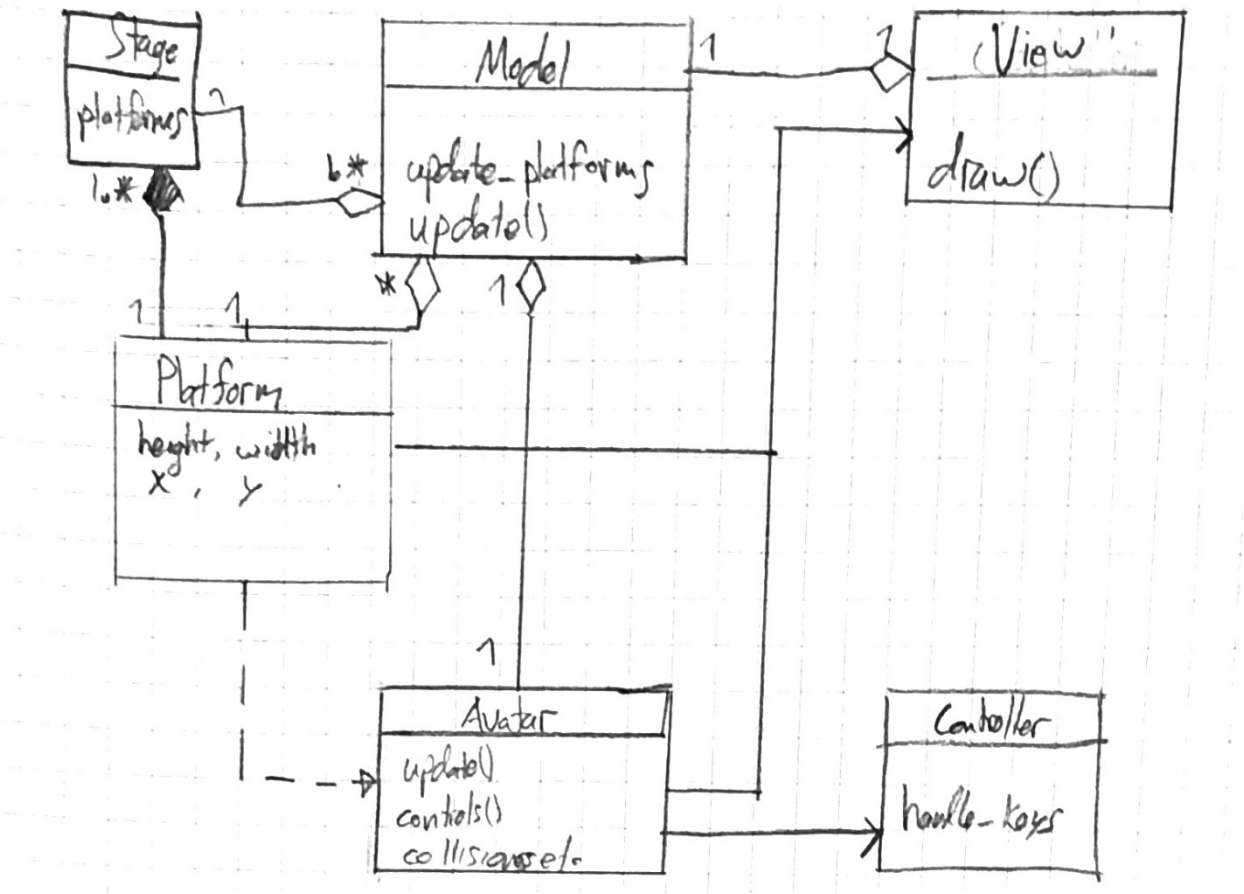
Michael Remley
Skye Ozga

MP4 Write-Up

Implementation

We implemented the model-view-controller organization for our project. The model contains the player's avatar, platforms, stages, and the overall model. The stages are lists of platforms, and the player class receives a list of these platforms from the overhead model class. With this and inputs from the controller, the avatar class decides where the avatar can move based off the arrangement of platforms and the player's inputs. All of this information is given to the view class, which then generates the view for the player.

The controller class initially used pygame events to directly make decisions on player movement, but we decided to instead give the model a list of current inputs and allow it to make all of those decisions. This makes the code to be more concise and readable. With the game being in full screen, the player can quit by pressing 'q' or 'p'. Other controls are 'a' and left arrow to move left, 'd' and right arrow to move right, and 'w', up arrow, or space to jump.



Michael Remley
Skye Ozga

MP4 Write-Up

Reflection

From a process point of view, the project went well in that we progressively developed the game, dealing with bugs and adding features along the way. Our project was scoped about right in terms of our goals; we did not reach our individual stretch goals, but we did implement wall-jumping. We wish we had known the failings of pygame, specifically events and collision framework.

We divided the work by class and pair programmed sometimes; when we neared the end, we divided and conquered to produce the final documentation. Generally, we handled bugs together if we were stuck on them for more than a couple minutes, but much of our work meshed together nicely. Next time, we would write the documentation as we go so that we each understand each other's code with each push.