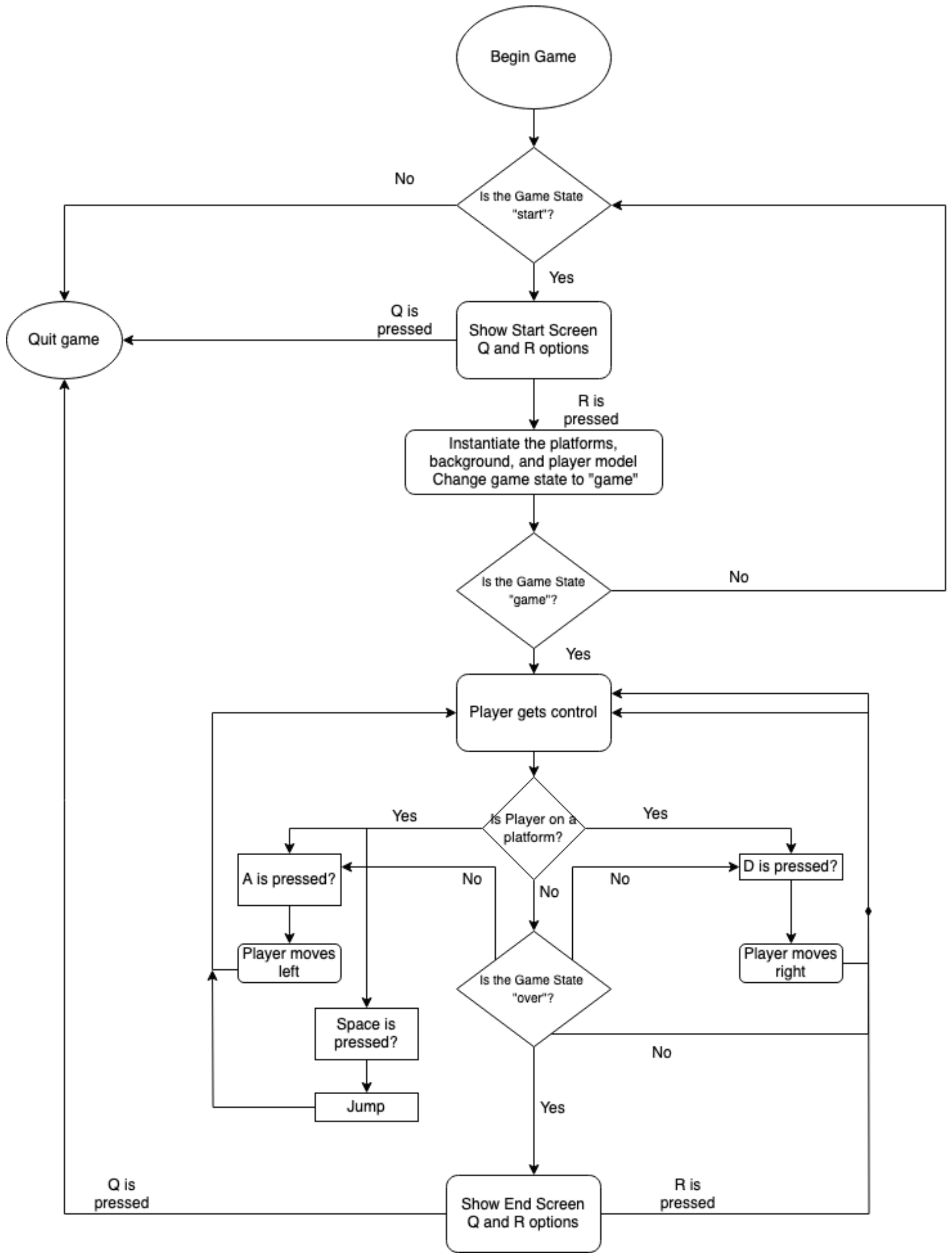
John Botonakis
5/8/2023

Final Project

Write a one paragraph description of what your project will do. (Think of this as a summary of your program or an 'app store' description.)

Jump! Is an interactive Doodle Jump inspired game, where the player controls a small green rectangle, and has to jump to various platforms in order to avoid death at the bottom of the screen. As well as precise movement and jump height, players will also encounter different platform types.

Use Microsoft Word (or some other program) to create a flowchart for your program. (Another student in the class should be able to take your flowchart and implement a controller program for your classes.)


Begin Game
Is the Game State "start"?
No
Yes
Quit game
Q is pressed
Show Start Screen Q and R options
R is pressed
Instantiate the platforms, background, and player model Change game state to "game"
Is the Game State "game"?
No
Yes
Player gets control
Is Player on a platform?
Yes
Yes
A is pressed?
No
No
No
D is pressed?
Player moves left
Player moves right
Is the Game State "over"?
Space is pressed?
No
Jump
Yes
Q is pressed
Show End Screen Q and R options
R is pressed

John Botonakis
5/8/2023

Final Project

Use Microsoft Word (or some other program) to create a class diagram for each class that you're going to use. (Another student in the class should be able to take your diagrams and implement your classes based on your descriptions.)

PLAYER	**PLATFORM**	**SAND PLATFORM**
Image: string	Image: String	Image: String
Score: int	Point: bool	Point: bool
Acceleration: (tuple)	———————————	————————————
Velocity: (tuple)		Sand(): bool
Position: (tuple)		
Jumping status: bool		
———————————————		
Move (self)		
Reset(self)		
Update(self)		
Jump(self)		
Cancel Jump(self)		

EXPLANATION OF CLASSES:

The PLATFORM class is the main framework for all of the following platforms that are generated using the self.plat_gen() command inside of the MAIN class. The platforms consist of an image and a points variable. It also makes it possible to register a collision when a player lands on it, as well as gaining points per NEW platform stepped on

The SAND PLATFORM class is a subclass of the main PLATFORM class. It is similar in that it also has an image and a points variable. Where it differs, is the actions that happen to the player while on a sand platform. It slows the player down by adding negative speed to the player's X velocity, making it harder to line up an otherwise easy jump!

The PLAYER class is the main controllable player character in the game. It handles things such as Points, basic X value movement, player updating, Jumping (as an applied action, rather than a player action), and hit detection for platforms. It also sets the platform point variable after jumping to ensure you can't double jump on one platform and rack up points.

The MAIN class is the controller for the GUI, and the first instantiation for every other class and subclass. Holding most of the game logic, the MAIN class handles Starting, Game Over, Points, Platform Generation, and surprisingly, Jumping for the player character.

John Botonakis
5/8/2023

Final Project

<u>ABOVE AND BEYOND</u>

- ALL of the art related to this project
  - Sprites
  - Background image
  - Game Start
  - Game Over
- The project itself
  - While I followed a guide, it was not only for a much simpler and smaller scale game, it was more or less useless as it had several problems that needed to be addressed not only for my game in particular but for the code to work as a whole
- Settings file
  - Ability to easily scale up and down the project using the settings txt file, as well as keeping key variables outside of the main program file for safety.
- Random level generation
  - While the beginning to most games is somewhat standardized, once the screen is scrolled upward, the level is completely randomly generated
- PyGame
  - Makes use of the PyGame engine, something I had to learn brand new specifically for this project.