Inkball Report

For my inkball project, I have the main “App” class that runs the PApplet logic, a “Level” class that represents a level file, and separate classes that represent the game objects, such as the “Wall”, “Ball”, and “Hole” classes.

The main “App” class doesn’t have any inheritance hierarchy. It creates the game board, instantiates game objects, and draws the game objects on the board.

The “Level” class stores the configurations of each level.

All the game objects (such as “Wall”, “Ball”, or “Hole”) inherit from the common parent class “GameObject”, which stores the x and y coordinates of an object’s position. This is because every game object needs an x and y coordinate to specify its location on the screen when its drawn.

Directly inheriting “Game Object” are two classes, “Tile” and “Ball”.

The “Ball” class contains the logic for the ball game object. The reason why the “Ball” and “Tile” classes are implemented separately is because ball objects move, while tile objects are stationary. Therefore the “Ball” class requires methods that involve updating the ball’s position, detecting walls and holes around the ball, colliding with lines, etc.

The “Tile” class is the parent class of three classes, “Wall”, “Spawner”, and “Hole”. These are stationary objects in the game that don’t require updating positions. Every “Tile” has four corners (top left, top right, bottom left, bottom right) represented by four x-y coordinate pairs, which are used for bounding box collision detection with the ball.

The “Hole” class inherits from “Tile” and implements its own methods for handling attracting and capturing the ball when the ball is within a certain distance.

The “Spawner” class also inherits from “Tile”, but it doesn’t implement any of its own methods. It is instead just a placeholder to indicate that a specific tile is a spawner so that the “App” class can randomly select one spawner for ball spawning.

The “Wall” class is another class inheriting from “Tile” and defines the general behavior for handling collision with the ball. These include changing the ball’s color to the wall’s color and reflecting the ball’s velocity correctly upon collision. The “Wall” class has a single child class “ColorRestrictingWall”.

The “ColorRestrictingWall” class is the extension of this project. It allows balls of a certain color to pass through while bouncing back all other colors. It inherits from “Wall” and defines its own four corners, as the rectangular sprite for the color restricting walls have different dimensions from the normal walls. It has an additional field that marks the color restricting wall as either vertical or horizontal, and it overrides the “Wall” class’s collision method to allow balls of the same color to pass through.