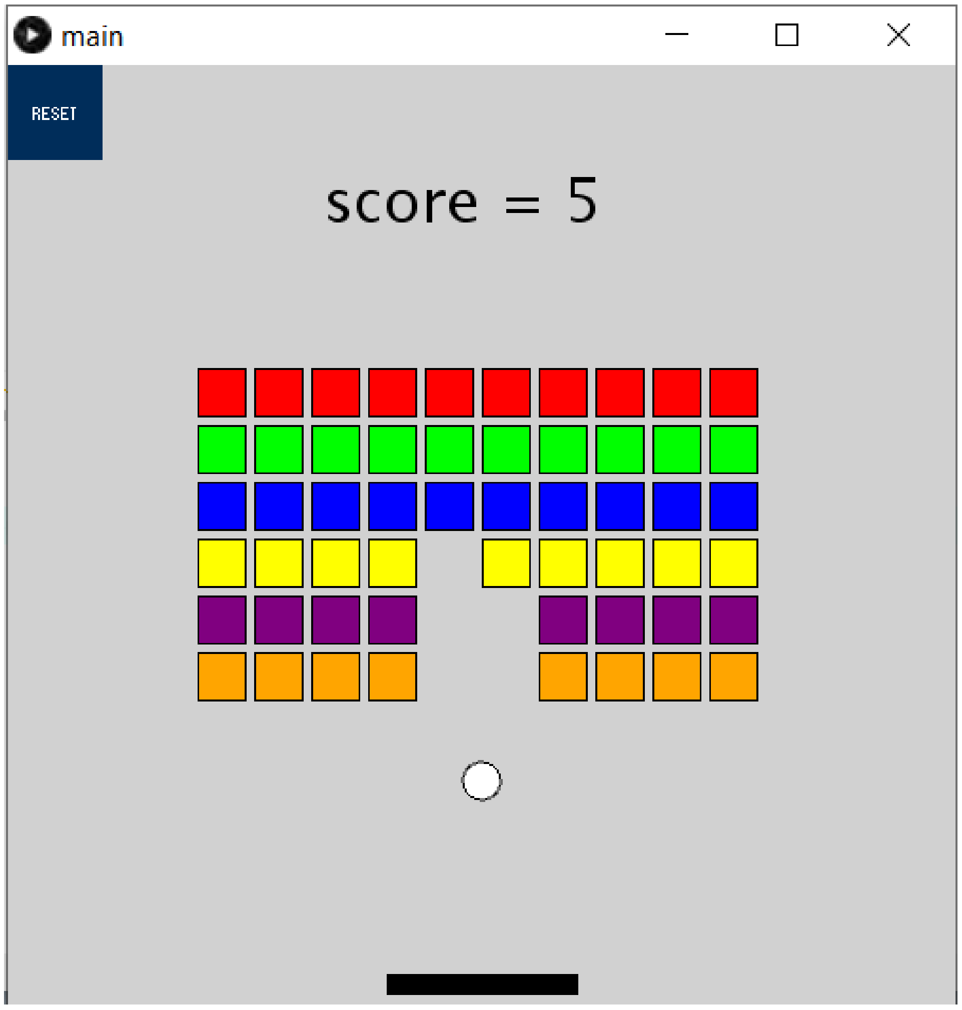
Project 4 - Breakout

Overview

For this project, you are going to implement a simple version of the game Breakout. The core concept in this assignment is an introduction to collisions and movement.

Description

For this assignment, you are going to create a program that looks something like the following:



The game will have the following UI components (use Control P5 where necessary):

- Reset button – Resets the game (Restores blocks and sets score back to 0)
- Text field: Displays current score, “You Lose.”, or “You Win!”
- 60 boxes to represent (10 columns x 6 rows)
    - Each row represents a different color
    - Each color represents a different scoring value
- One circle to represent the moving ball
- One rectangle to represent the moving paddle

The game will have 3 main parts:

- Movement:
    - Ball movement via velocity
    - Paddle movement via keyboard input
- Collisions
    - Circle-box
    - Circle-edge/wall
- Scoring
    - (**see Scoring details below**)

How to Play:

- The game starts off with 60 boxes. The ball starts from the paddle and moves towards the boxes.
- Every time a box is hit by the ball, the box should disappear and point(s) are added to the score.
- The ball can bounce off the left, top, and right walls. The ball can also bounce off the top side of the paddle.
- If the ball collides with the bottom, the player loses.
- **Goal:** Hit all 60 boxes using the walls and paddle
    - If the ball hits all 60 boxes, the player wins.

Scoring:

- Row 1 (closest to the Paddle): 1 point
- Row 2: 2 points
- Row 3: 3 points
- Row 4: 4 points
- Row 5: 5 points
- Row 6 (furthest away from the Paddle): 6 points

Keyboard Input:

- ‘a’ = slide paddle left
- ‘d’ = slide paddle right
- ‘p’ = pause game

Class Structure

You should create 5 classes for this project:

- Main
    - Runs the game. Initializes objects and calls functions necessary to run the game.
- GameState
    - Brains of the game. Handles how the game is run especially interactions between the Ball, Paddle, and Box objects.
- Ball
    - Represents a circle, which also has movement via velocity.
    - Knows how to collide with the Paddle and Box objects
- Paddle
    - Represents a rectangle, which also has movement via keyboard input.
- Box
    - Represents a rectangle (also a square), which disappears when the ball collides into it

Note: Skeleton classes with comments are provided to get you started. Each class should know how to ***update()*** its state and then ***draw()*** itself.

Main

The main.pde file creates and initializes a GameState object. Also, the setup(), update(), and draw() functions are invoked here. The screen window should be 800 x 800 pixels.

Class: GameState

The GameState object created through the main class will hold the variables pertaining to the objects playing the game (ball, paddle, and boxes). The update function will handle object changes and positions (motion/velocity, collisions, drawing, scoring, etc.).

Class: Box

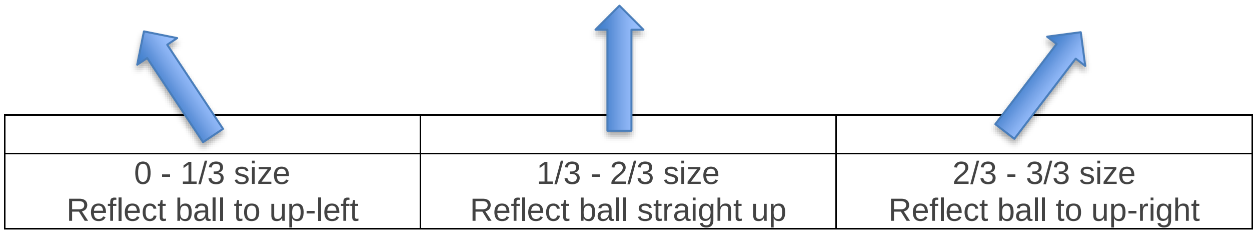
The box object represents one box, which disappears when the ball collides into it. The box is a rectangle (shaped like a square). Multiple of these objects will need to be created to display the 60 boxes for the game. Each box should be 25 x 25 pixels. Each row of boxes should have a distinct color (see picture on first page). Colors are your choice.

Class: Paddle

The paddle is a rectangle. It should be 150 x 20 pixels. When the “a” key is pressed, the paddle should move LEFT along the x-axis. When the “d” key is pressed, the paddle should move RIGHT along the x-axis. It should stay constant on the y-axis near the bottom of the screen (paddle does not move up or down).

The direction of the ball’s bounce is based on the location of the ball’s collision on the paddle.

- 0 - 1/3 size of Paddle: Reflect ball to up-left
- 1/3 - 2/3 size of Paddle: Reflect ball straight up
- 2/3 - 3/3 size of Paddle: Reflect ball to up-right



Paddle Reflection Diagram

A simple of these velocities be a value of 1 or 0 in either the X or Y components. “Up and to the left” could be an X and Y value of -1 each, up and to the right could be an X and Y value of 1 and -1, respectively, and straight up could be (0, -1). (Remember that in Processing the top of the screen is a Y value of 0.)

Normalizing vectors: A vector of (1, -1) is technically “faster” than a vector of (0, -1)—that is, it has a higher magnitude. This could cause issues in your program, so you should use the normalize() function of the PVector class to change the magnitude to 1.

Class: Ball

The ball is a circle (you could use an ellipse shaped like a circle as well). It should have a position, velocity, diameter, and speed. The diameter of the ball should be 20. All other ball values are your choice. This would be a great place to implement collision detection (ball-box, ball-paddle, and ball-outside-walls). When collision is detected, the ball should bounce and change direction appropriately.

Sometimes this may be multiplying the velocity by -1, other times it may be changing only the x or y component. For example, if the something is moving straight up with a velocity of (0, 1), then bouncing and reversing direction should result in a velocity of (0, -1). The two side walls should change the x direction of the ball, while the top should change the y direction. When hitting a block, it depends where the ball was coming from relative to the block.

There are many resources online to help with implementing collision detection. Here's one of many you could use for guidance:
https://yal.cc/rectangle-circle-intersection-test/

Extra Credit

- Create at least 2 distinct "power-up" boxes (replaces a normal box).
  - A power up can be anything you think will help the player have an advantage at winning the game more efficiently
  - Examples:
    - A larger ball
    - A larger paddle
    - Spawning additional balls
    - Stretching the size of the paddle
    - Changing the speed of the paddle
    - Firing projectiles from the paddle
    - A ball that doesn't bounce after colliding with a block
  - Use an image to represent each distinct power up
- Randomly distribute the 2 [or more] distinct power-ups within the 60 boxes.
  - There should be 10 power-up boxes out of 60 boxes total.
- Write a text file called "YOURNAME-Proj04-Extra-Credit.txt" that describes what each power-up accomplishes. Also, describe the image used with each power up. Make it easy for graders to understand what is going on.

Submissions

Create a .zip file with any code files you created for this project (in Processing they are files with the .pde extensions). Also, include your extra credit text file if you choose to do so. No extra credit will be given without the submission of an extra credit text file.

Feel free to be as creative as you please with this project. Just be sure to meet the minimum requirements stated throughout this document.

Tips

1. Plan your design ***before*** writing any code.
2. Write and test code in small increments.
3. It may be useful to place objects in the direction of the velocity to test collision.
4. It may be useful to have separate testing scenarios where you make several of these objects and several balls so you can visually and thoroughly test your collision logic.
5. Use keyPressed() and keyReleased() to help detect keyboard input

Grading

Item	Description	Possible Points
UI	Reset button, score display, 60 boxes, ball, and paddle.	20
Paddle-Ball collision	Correctly collide and reflect the ball.	20
Box-Ball collision	Correctly collide and reflect the ball.	20
Wall-Ball collision	Correctly collide and reflect the ball.	20
Paddle movement	‘**a**’ and ‘**d**’ keys control paddle	10
Ball movement	Ball moves reasonably	10
Pause button	Pressing ‘**p**’ pauses the game	10
Reset Button	Resets the game	10
Paddle Reflection Direction	Reflect ball in correct direction after collision	20
Text Display	Show correct score, winner, and loser messages within Text Display.	10
	Total	150
Extra Credit	10 Points for each power-up implemented (max 20 points)	20

Partial Credit

Partial credit will be given for features which work, but have some sort of functional defect. For example, if collision with the paddle works most of the time, but occasionally works incorrectly, the points for that section will be reduced by 50%.