1. Introduction

The breakoutgame is a game developed in 1976 by Steve Wozniak that become a commercial success selling over 3500 arcades cabinets (ref 1). While it is relatively simple game it the game is composed by objects in a GUI that represent a ball, a paddle and bricks, that can be destroyed and represent a score. The principal component in the game is the ball and the bricks, that when destroyed take up a certain score that is track and can changes the game play style.

However, due to difficulty to avoid repetitions is important to create function that keep track of the bricks. Finally, after a score then needs to be marked to adress the rules imposed.

2. Game rules explain

In Breakout, the initial configuration of the world appears as shown on the right. The colored rectangles in the top part of the screen are bricks, and the slightly larger rectangle at the bottom is the paddle. The paddle is in a fixed position in the vertical dimension but moves back and forth across the screen along with the mouse until it reaches the edges of the window.

A complete game consists of three turns. On each turn, a ball is launched from the center of the window toward the bottom of the screen at a random angle. That ball bounces off the paddle and the walls of the world, in accordance with the physical principle generally expressed as "the angle of incidence equals the angle of reflection" (which turns out to be very easy to implement as discussed later in this handout). Thus, after two bounces—one off the paddle and one off the right wall—the ball might have the trajectory shown in the second diagram. (Note that the dotted line is there only to show the ball's path and does not actually appear on the screen.) As you can see from the second diagram, the ball is about to collide with one of the bricks on the bottom row. When that happens, the ball bounces just as it does on any other collision, but the brick disappears. The third diagram shows what the game looks like after that collision and after the player has moved the paddle to put it in line with the oncoming ball. (REF 2)

3. Methods

We developed the breakout game with the turtle library of tkinter. The first function we use was to import Screen and store it in variable. The with setup adjust the width and heigh. Then use bgcolor to change to black and title the window "Breakout Game". Next use tracer(0) to remove turtle animations and screen.exitonclick() to loop the screen

Next we create a no python file to create the paddle. For this initiate a super.init method for object construction and start define the shape, size and color of the paddle. We the object done, create a function for setting the initial position with the goto() method. Use screen.listen() to see it display

The paddle is going to be moved by pressing the arrows keys. For this we use the screen.onkey() method that reacts with a function pressed key. Create a function to go to right and other for the left where in there set a variable equal to paddle position with self.xcor() and add 30 pixies to the movement then with self.goto() send the object to this new position. Call this functions in main.

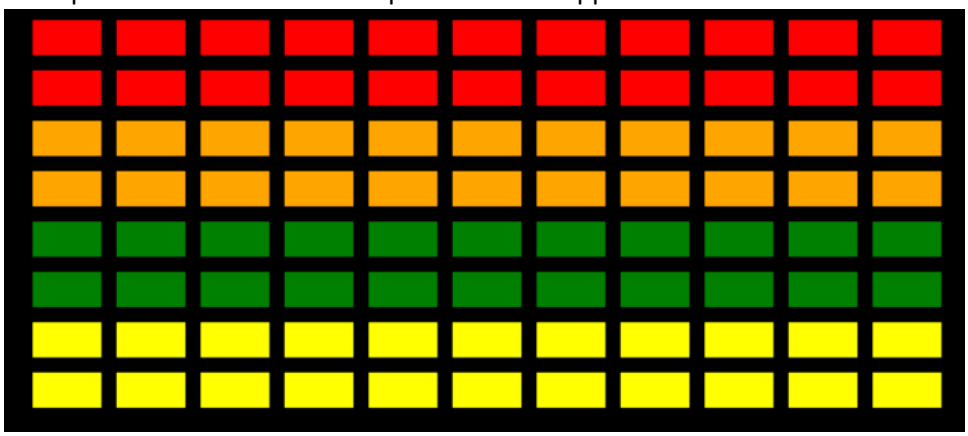
Create another python file where you execute a class init method for the ball and set up its

Object shape, color and two variables equal to one for the vector's direction. Next create a while loop and call this method in main. Create a function for the movement set a function in ball to subtract to the inticial position and call goto() just like in previous with the paddle but here multiply by the set variable in the super.init.

Now in the main while loop start the conditional operations, where you set the limit of the ball coordinates to then make the ball bounce of this limits. To do this all we need to do is to in ball file create a function that multiples the set variables by minus 1 to x and y depending on the limits set. Do a similar process for the ball bouncing on paddle but instead of the conditional statement use distance(ball) < 50. That activates if the ball is near 50 pixies.

To make the brick of the wall create another python file and define all the positions in the window in a form of tuple inside a dictionary that has a list for red bricks, orange, green and yellow.

Next create a class and init and set a variable for a list where we can later save the bricks in here. Then create and four functions for each colored brick, where you define the in a variable each color and a for loop where you will loop all the positions inside the bricks color list. Then create a function with two variables where you create the turtle objects of the bricks shape and color and set the positions with the loop list and append it to the list in the init. (imag1)



Next for the detection we created a method for detection of coordinates and tracking of the bricks inside the list. In the while loop create eight conditional statements that detect if the ball y coordinates are inside the range of the colour bricks. Then make a function to send a number that represents the level of the line bricks.

Inside this function create a eleven conditional statements to see if the ball.distance is near the bricks inside the wall list. To get the number of the list multiply the variable number of the line by 11 (number of bricks on each line) and subtract by a distance. Repeat this condition to set all the brick in each line.

Next to make the wall disappear create a function in the wall with goto() that send that brick to outside limits of the window then call a function that passes the line of the wall and determines the score of the destroyed wall.

Finally, this last step was to create a score and display it during the game without counting twice. This is done by create a separate function that receives the variable line of wall and if color brick range call the proper score points for that color brick. Create another python file and here create a class super init, with variables for the score and lives and the go to positon and a updfate function. In this update function use clear to clear text and write a turtle with the lives and score.

In the end create function for the brick color point that add to the score the points and updates function. Also create a game over sequence