Part2 - Breakout Game

The part2 is a simple breakout game build with `tkinter`, no third party library used. User could click the menu bar to start a new game while they are waiting in the queue.

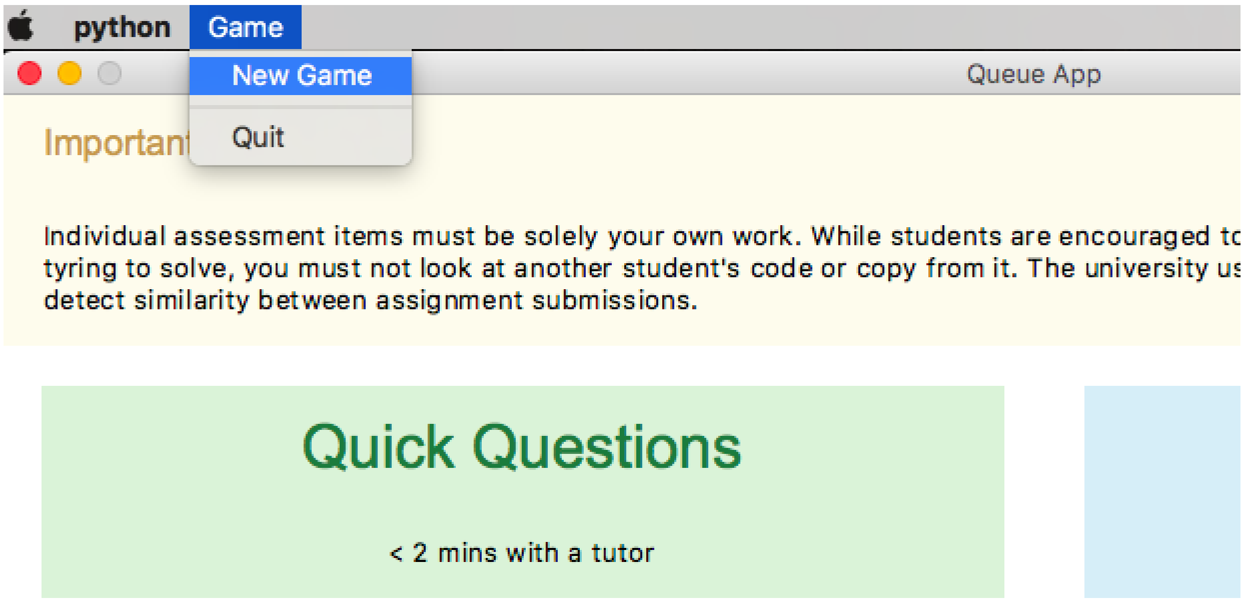
Library used

- `tkinter`

How to play:

Start New Game

When user open the Queue app, there will be a Game menu initialized at the top. Click it and choose new game to start new game, a pop up window will show then.



Importan

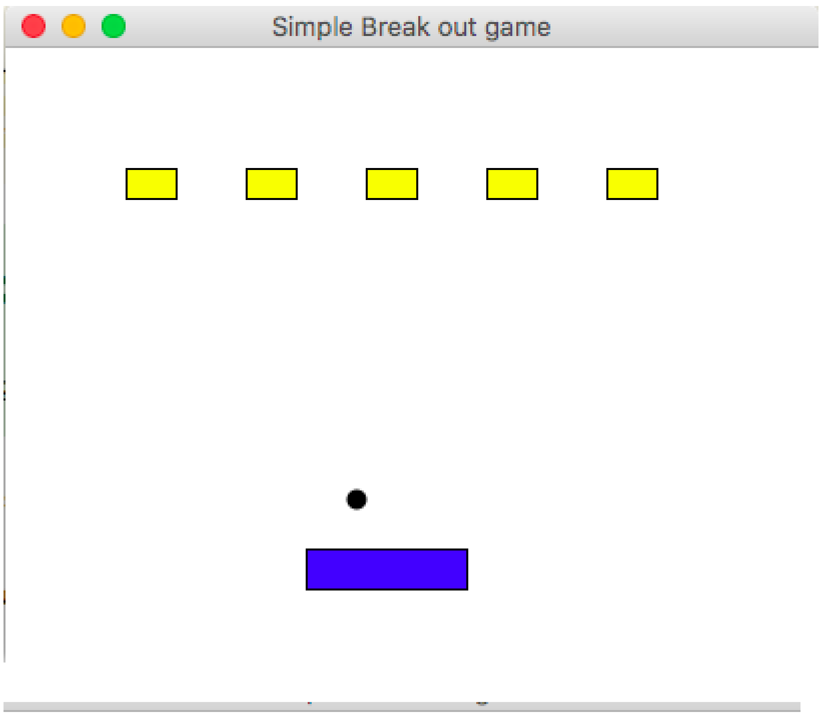
Individual assessment items must be solely your own work. While students are encouraged tc
tyring to solve, you must not look at another student's code or copy from it. The university us
detect similarity between assignment submissions.

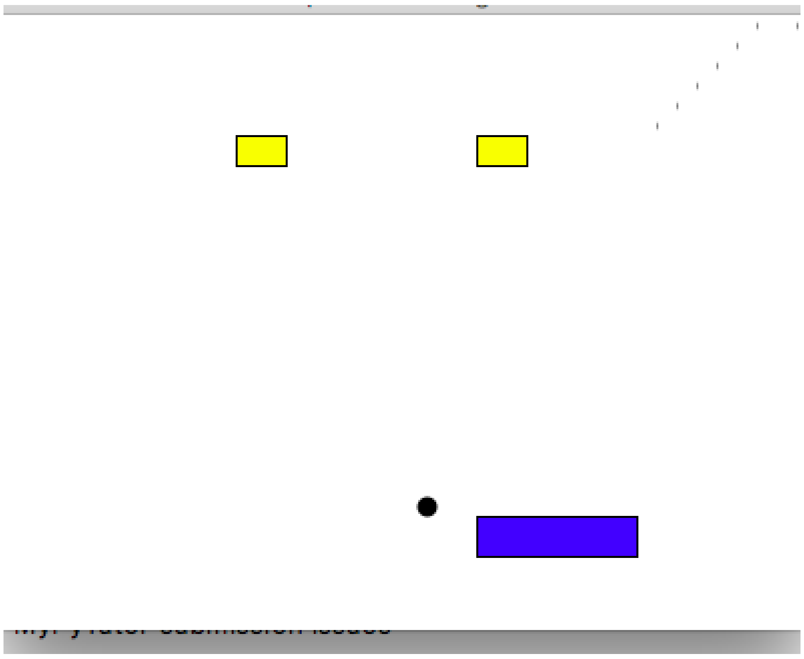
Quick Questions

< 2 mins with a tutor

Play

Once open the game, The game should first be static. The blue rectangle is your paddle, balck circle is ball and five yellow rectangles are blocks. Once the user press `f` on their keyboard, the ball will be fired, and you can press `a` and `d` to move your paddle to left and right. When the ball hit a paddle, it will be bounced. When the ball hits the block, the block will disappear, this means that block has been "killed".





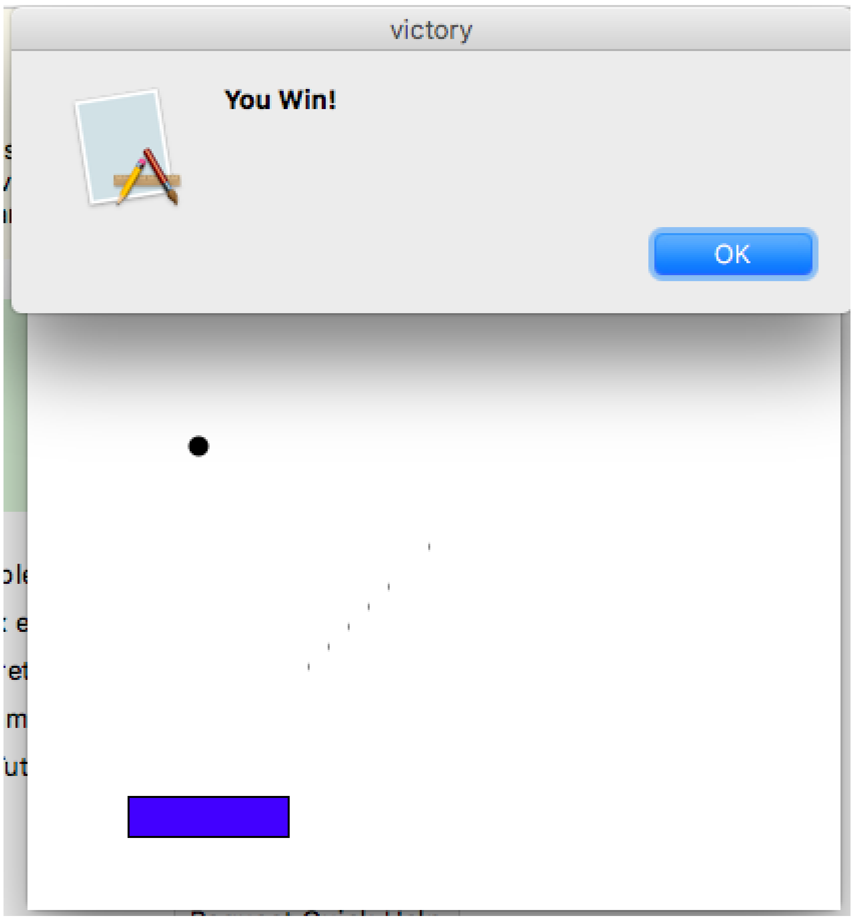
move

Game end

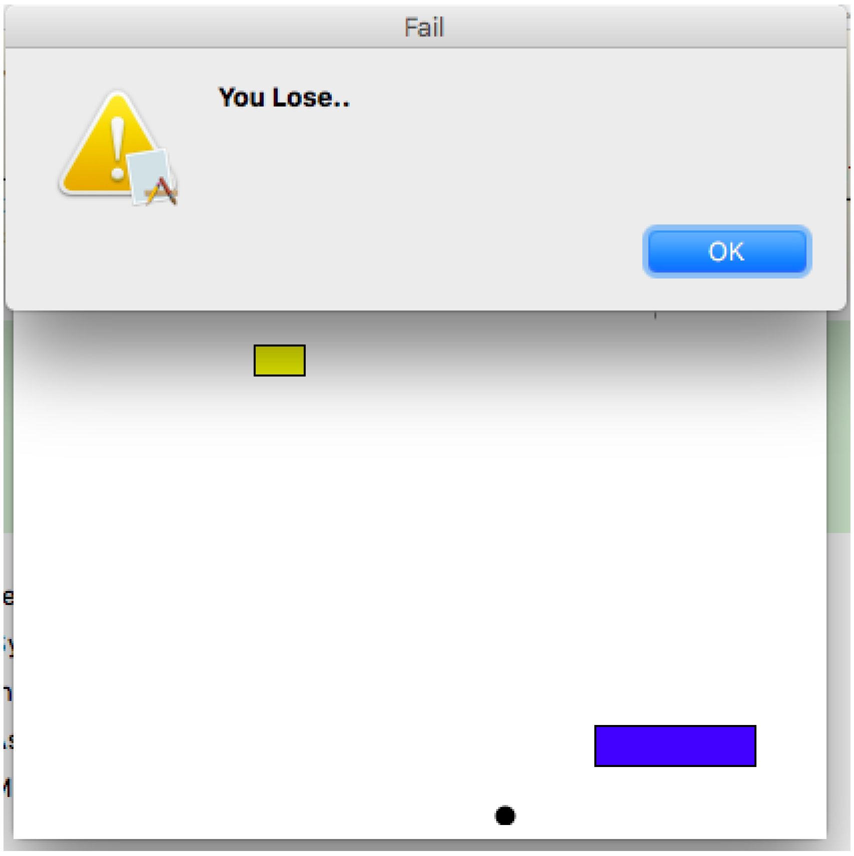
There will be two situations when the game ends:

- Killed all the blocks, win the game
- The ball hit the ground, lose the game

When the game is end, the game window will be frozen, ball will not move anymore, and a relative message window will pop up saying you win or lose



win



lose

Quit game

There are two options to quit the game:

- Click the X to close the game window

- Go to the menu and click `quit` option to quit the game.

Code approach

Menu part

For the `Menu`, i used the `tk.Menu()` to create a menu and add submenu(New game and quit) after it.

The `new_game()` method will start a new game in a pop up. For the pop up, i used `tk.Toplevel()` to generate a new pop up

The `clsoe()` method will close the game pop up window

```
class Menu:
    def __init__(self, master):
        menubar = tk.Menu(master)
        master.config(menu=menubar)

        gamemenu = tk.Menu(menubar)
        menubar.add_cascade(label="Game", menu=gamemenu)
        gamemenu.add_command(...)

    def new_game(self):
        self.game_window = tk.Toplevel()
        pass

    def close(self):
        pass
```

Game part

The game contains several classes:

- `Paddle`
- `Ball`
- `Block`
- `Game`
- `GameApp`

Paddle

The `Paddle` class contains several attribute and method:

```
class Paddle:
    def __init__(self, canvas):
        pass

    def draw_image(self):
        pass
    def move_left(self):
        pass

    def move_right(self):
        pass

    def clear(self):
        pass
```

Ball

The `Ball` class is little bit different from the `Paddle`, since it will moves all the time, an attribute callled `fired` is given to determine whether it is fired(when user press `f`)

```
class Ball:
    def __init__(self, canvas):
        # ...
        self.fired = False

    def fire(self):
        self.fired = True

    def move(self):
        pass

    def draw_image(self):
        pass

    def clear(self):
        pass

    def collide(self):
        pass

    def bound(self):
        pass

    def die(self):
        pass
```

Block

The `Block` class is similar with `Paddle`, however, the difference is that, when the ball hit it, it will be removed then, so an attribute called `alive` is given.

```
class Block:
    def __init__(self, canvas):
        # ...
        self.alive = True

    def draw_image(self):
        pass

    def kill(self):
        self.alive = False

    def react_intersects(self, rect1, rect2):
        pass

    def collide(self, o):
        pass

    def clear(self):
        pass
```

Game

The `Game` class is inherited from `tk.Canvas`. It will just be a container to be put other shape classes.

```
class Game(tk.Canvas):
    WIDTH = 400
    HEIGHT = 300

    def __init__(self, master):
        super().__init__(master, width=self.WIDTH, height=self.HEIGHT)
        self._master = master
```

GameApp

The `GameApp` class is the main app to initialize the game and deal with the logic part. It is the `Controler` part for the `MVC` model.

The basic theory is using `root.after(interval, callback)` to refresh and update all the information about shape classes.

The `callback` function will do such things:

- Check if the game is end
- Check if the ball hit paddle and block
- Clear all the objects on the canvas
- Redraw all the objects on the canvas again.

```
class GameApp(tk.Frame):
    # ... init
    self.update_all()
    # other methods

    def update_all(self):
        # check game end
        self.check_win_game()
        self.check_lose_game()
        if self.check_end():
            return
        # check collide
        self.check_all()
        # clear
        self.clear_all()
        # draw
        self.draw_all()
        self._master.after(50, self.update_all)
```

Todo

During the game, i actually need to determine if two items overlap, i used code snippet from `stackoverflow` to roughly judget the collide. However, that code is based on two rectangles. the circle and rectangle is a little bit different, I did not implement a lot in this part.

Reference

https://stackoverflow.com/questions/20846944/check-if-two-items-overlap-on-a-canvas-using-javascript