

**BINUS UNIVERSITY**

**BINUS INTERNATIONAL**

**Assignment Cover Letter**

**(Individual Work****)**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | |  | | |  | | |
| **Student Information**: **Surname** | | | | | **Given Names**  **Krishita** | | | **Student ID Number**  **2101716773** | | |
| 1. | | **Sukhani** |  | |
|  |  |
| **Course Code** | **: COMP6335** |  |  | | **Course Name** | | | **: Introduction to Programming** | | |
| **Class** | **: L1AC** |  |  | | **Name of Lecturer(s)** | | | **:** 1. Bagus Kerthyayana | | |
|  |  |  |  | |  | | | 2. Tri Asih Budiono | | |
| **Major** | **: CS** |  |  | |  | | |  | | |
| **Title of Assignment**  (if any) | : brick breaker | |  |  | | |  | | |  | |
| **Type of Assignment**    **Submission Pattern** | **: Final Project** |  |  | |  | | |  | | |
| **Due Date** | **: 6-11-2017** |  |  | | **Submission Date** | | | **: 6-11-2017** | | |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

Binus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

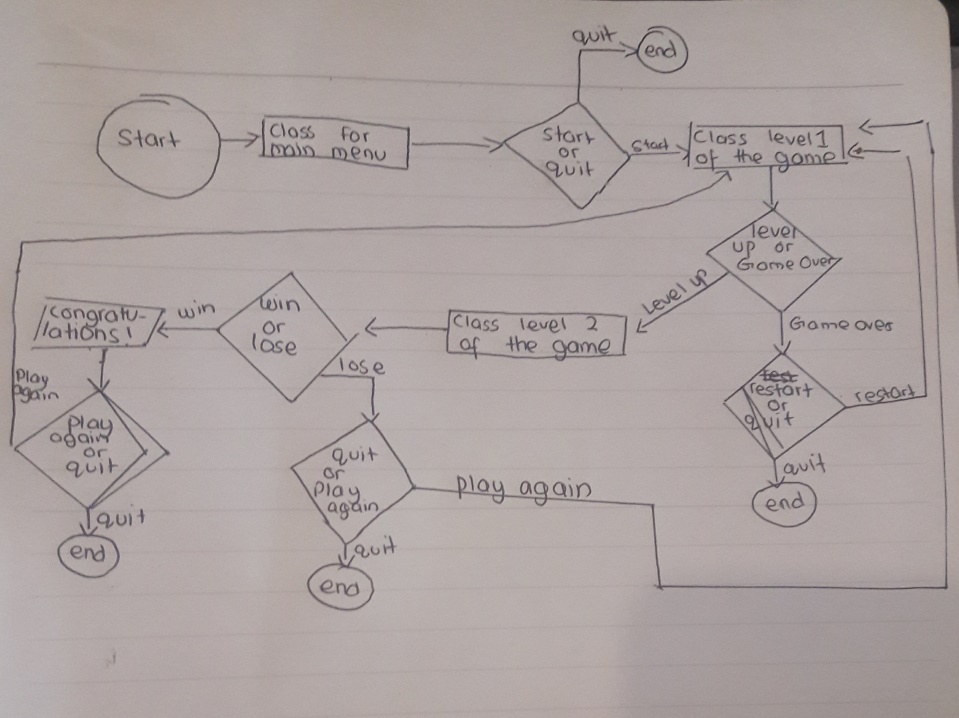
By signing this assignment, I understand, accept and consent to Binus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student) Krishita Sukhani

**“Brick Breaker”**

**Name :Krishita Sukhani**

**ID : 2101716773**

****

1. **Description**

**The function of this program:**

This is a classic Brick breaker game, the point of the game is to break all the bricks(blocks) and catching the ball with the paddle without the ball falling out of the game as that would immediately end the game.

There are 2 levels of the game the 1st one has more spaces between the bricks making it easier as well as a slower ball movement making it easier for players to catch it and a longer distance between the brick and the paddle

# what was implemented and how it works

1. **the menu():**

* is a function that works by giving you 2 options
* if you choose to play the game it calls the level 1 of the game
* if you choose to quit then it calls the pygame.quit()

1. **level 1 and level 2(is a class) :**

* this holds the characteristics such as the height width color of the Paddle (used to catch the ball), the ball( used to break the bricks) , the bricks also called as a block, also holds the direction and speed as well as swap which is later on used for the “ballUpdates”
* this holds createBlocks which tells the program where to locate the x axis and y axis of the blocks in this level of the game as well as the width and the height of the blocks
* holds the ball update that tells the system what to do with the ball when the ball bounces back from the paddle after it is caught it includes the balls speed, the balls angles and the direction or what to do with the ball if it is not caught by the paddle, the ball update also has the score i.e if it reaches its maximum score it will move on to the next level or else if the ball falls off the game area then the system will show the gameover function
* holds the paddleUpdate which tells the system what to do when the user is dragging the paddle by the keypad and it updates the paddles every move
* the main, this holds the while loop for the game and the background as well as the sound

1. **The end**

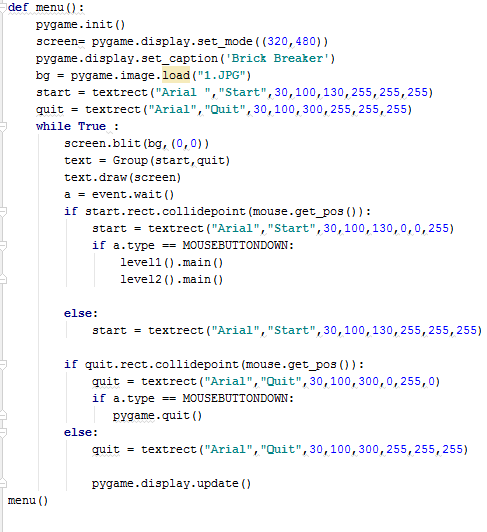
* The end function tells the system what to do if the user had lost already , users loose when they do not catch the ball
* Either allows you to restart which will call the level1 class and lets you play again
* Or it either allows you to exit the system if you no longer wish to play

1. **WINNER**

* Is a function that is called when the user have completed both the levels
* It prints a congratulation note

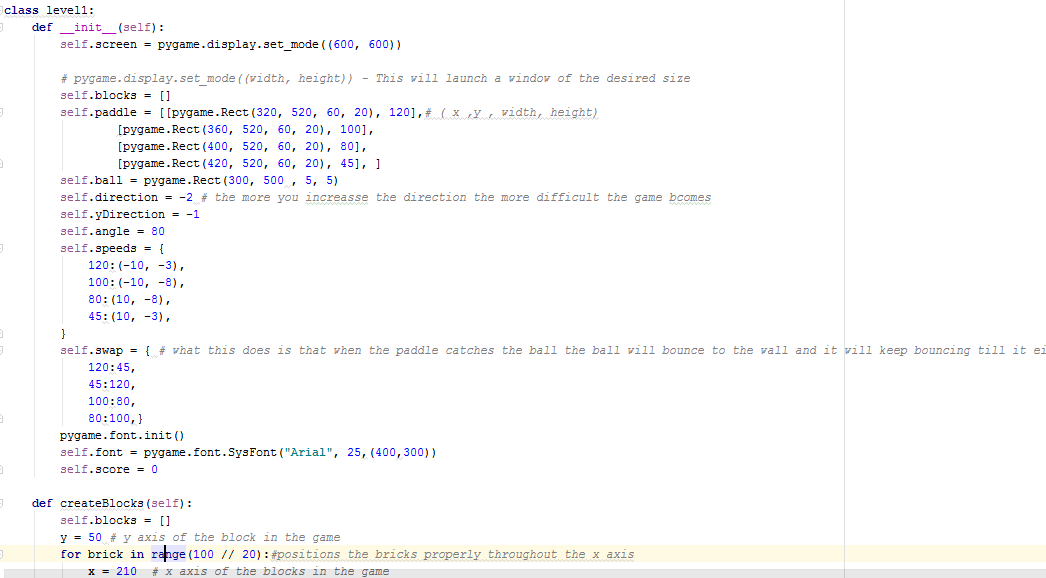
# A brief teaser on the game

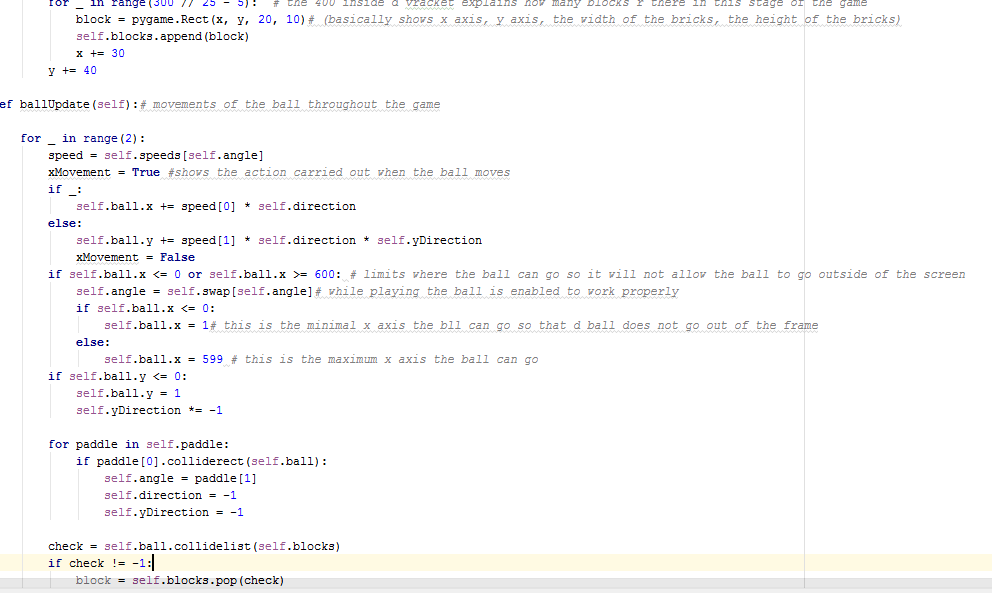
1. Menu

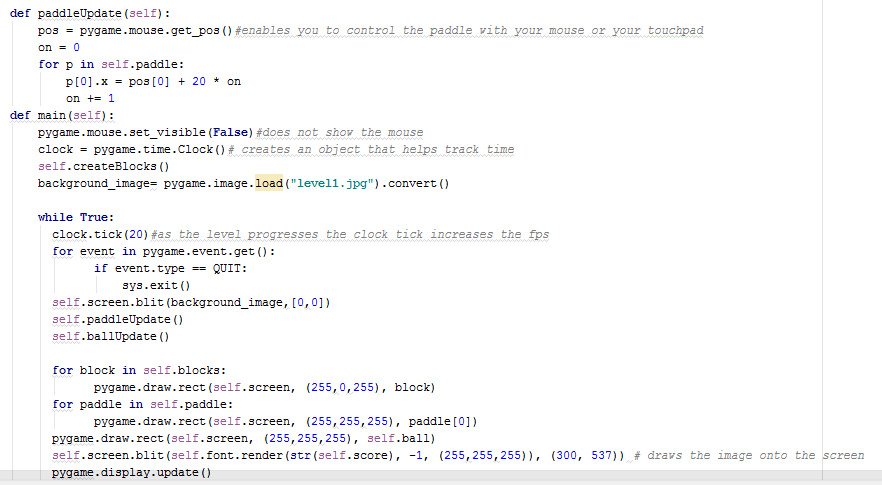
:

2.Level 1





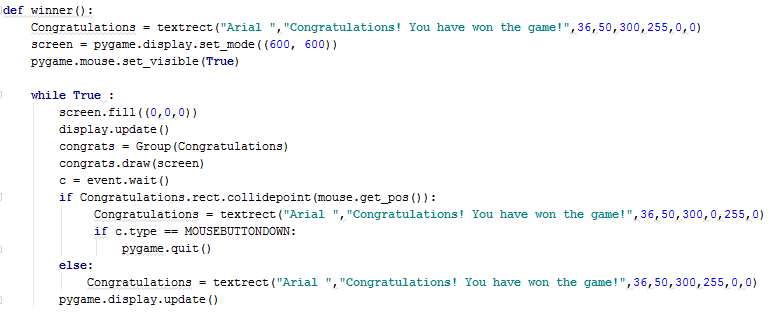




3.The end (what shows when the player has lost)



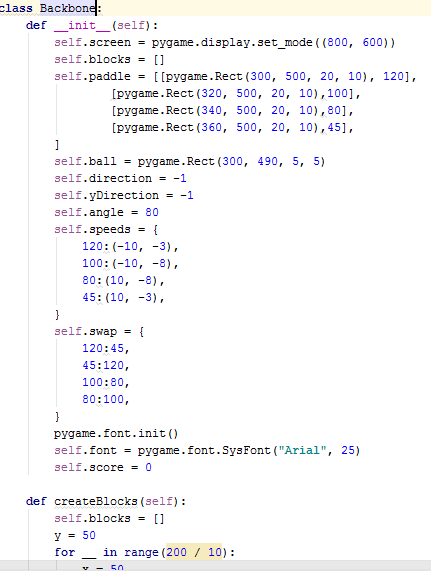
4.The winner(shows what happens when the player has defeated both levels)



Documentation

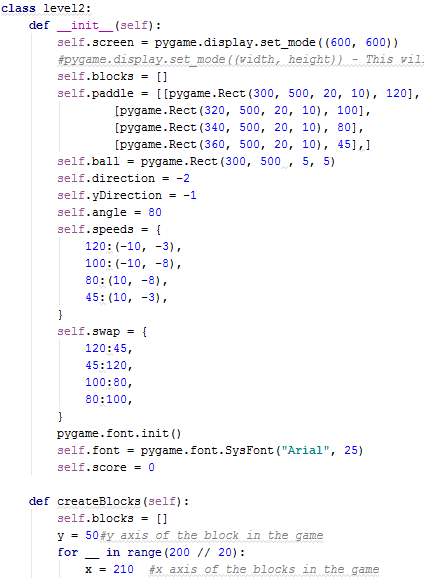
1. 27 October 2017: *(basic game)*

I have created the very basic functioning class for my game, for now I call my class BackBone and what it does is it breaks the bricks when it collides with the ball and the game would immediately end if the paddle does not catch the ball



1. 28 October 2017: *(level 1 and 2 of the game)*

Made a level 2 for my brick breaker program, I tried making a super class that is the backbone and 2 separate classes as a child class called level1 and level2 however I decided to just make 2 separate classes for each level

1. 
2. 30 October 2017: *(creating menu and end )*

Every game needs a menu so for today I created a menu function for my game, it is a very simple menu as it has 2 options that is to start the game or to quit

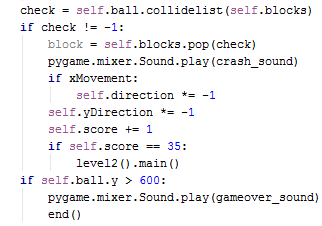


After creating the menu I realized that every beginning must have an ending therefore I created another function for end, this will occur if the paddle does not catch the ball. I also modified both my classes so if the ball falls out of its respective frames then the game would immediately end

1. 1 November 2017: (sound effects)

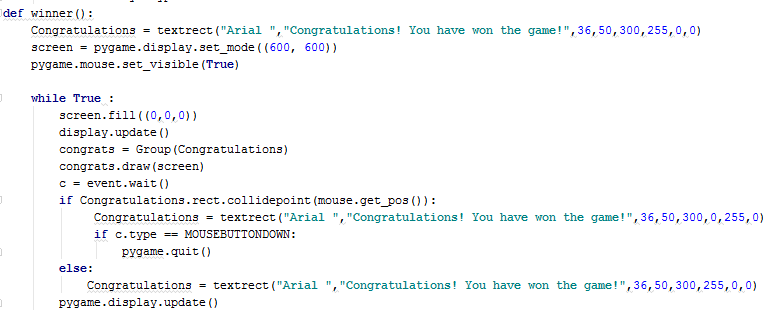
Used pygame.mixer.Sound to play a sound when the ball collides with the block as well as when the game dies





1. 3 November 2017: (the winner)

This is a function that is called once all the blocks from both the levels have been popped



Once the game is done the user an press the congratulations note and the game would end

1. 4 November 2017:

Today I have updated the backgrounds for both game levels as well as the menu as it was too simple



**SOURCE CODE**

**import** pygame  
**from** pygame.locals **import** \*  
**import** sys  
**from** pygame.sprite **import** \*  
**from** pygame **import** \*  
pygame.init()  
crash\_sound = pygame.mixer.Sound(**"break.WAV"**)  
gameover\_sound= pygame.mixer.Sound(**"die.WAV"**)  
**class** textrect(Sprite):  
 **def** \_\_init\_\_(self, fontstyle , text , fontsize , xpos , ypos ,R, G , B ):  
 Sprite.\_\_init\_\_(self)  
 self.font= pygame.font.SysFont(fontstyle,fontsize)  
 self.image= self.font.render(text, **False**,(R,G,B))  
 self.rect = self.image.get\_rect()  
 self.rect.x = xpos  
 self.rect.y = ypos  
**def** winner():  
 Congratulations = textrect(**"Arial "**,**"Congratulations! You have won the game!"**,36,50,300,255,0,0)  
 screen = pygame.display.set\_mode((600, 600))  
 pygame.mouse.set\_visible(**True**)  
  
 **while True** :  
 screen.fill((0,0,0))  
 display.update()  
 congrats = Group(Congratulations)  
 congrats.draw(screen)  
 c = event.wait()  
 **if** Congratulations.rect.collidepoint(mouse.get\_pos()):  
 Congratulations = textrect(**"Arial "**,**"Congratulations! You have won the game!"**,36,50,300,0,255,0)  
 **if** c.type == MOUSEBUTTONDOWN:  
 pygame.quit()  
 **else**:  
 Congratulations = textrect(**"Arial "**,**"Congratulations! You have won the game!"**,36,50,300,255,0,0)  
 pygame.display.update()  
  
  
  
  
**def** end():  
 go = textrect(**"Arial "**,**"Game Over"**,50,200,300,0,255,0)  
 restart = textrect(**"Arial"**,**"Restart"**,50,200,200,0,0,255)  
 screen = pygame.display.set\_mode((600, 600))  
 pygame.mouse.set\_visible(**True**)  
  
 **while True** :  
 screen.fill((0,0,0))  
 options = Group(go,restart)  
 options.draw(screen)  
 display.update()  
 b = event.wait()  
 **if** restart.rect.collidepoint(mouse.get\_pos()):  
 restart = textrect(**"Arial"**,**"Restart"**,50,200,200,0,0,255)  
 **if** b.type == MOUSEBUTTONDOWN:  
 level1().main()  
  
  
 **else**:  
 restart = textrect(**"Arial"**,**"Restart"**,50,200,200,255,0,0)  
  
 **if** go.rect.collidepoint(mouse.get\_pos()):  
 go = textrect(**"Arial"**,**"GameOver"**,50,200,300,255,0,255)  
 **if** b.type == MOUSEBUTTONDOWN:  
 pygame.quit()  
 **else**:  
 go = textrect(**"Arial"**,**"GameOver"**,50,200,300,0,255,0)  
  
 pygame.display.update()  
  
**class** level1:  
 *#reference list* **def** \_\_init\_\_(self):  
 self.screen = pygame.display.set\_mode((600, 600))  
  
 *# pygame.display.set\_mode((width, height)) - This will launch a window of the desired size* self.blocks = []  
 self.paddle = [[pygame.Rect(300, 500, 20, 10), 120],  
 [pygame.Rect(320, 500, 20, 10), 100],  
 [pygame.Rect(340, 500, 20, 10), 80],  
 [pygame.Rect(360, 500, 20, 10), 45], ]  
 self.ball = pygame.Rect(300, 500 , 5, 5)  
 self.direction = -2  
 self.yDirection = -1  
 self.angle = 80  
 self.speeds = {  
 120:(-10, -3),  
 100:(-10, -8),  
 80:(10, -8),  
 45:(10, -3),  
 }  
 self.swap = {  
 120:45,  
 45:120,  
 100:80,  
 80:100,  
 }  
 pygame.font.init()  
 self.font = pygame.font.SysFont(**"Arial"**, 25)  
 self.score = 0  
  
 **def** createBlocks(self):  
 self.blocks = []  
 y = 50 *# y axis of the block in the game* **for** \_\_ **in** range(100 // 20):  
 x = 210 *# x axis of the blocks in the game* **for** \_ **in** range(300 // 25 - 5):  
 block = pygame.Rect(x, y, 20, 10)  
 self.blocks.append(block)  
 x += 30  
 y += 40  
  
 **def** ballUpdate(self):  
 **for** \_ **in** range(2):  
 speed = self.speeds[self.angle]  
 xMovement = **True  
 if** \_:  
 self.ball.x += speed[0] \* self.direction  
 **else**:  
 self.ball.y += speed[1] \* self.direction \* self.yDirection  
 xMovement = **False  
 if** self.ball.x <= 0 **or** self.ball.x >= 600: *# limits where the ball can go so it will not allow the ball to go outside of the screen* self.angle = self.swap[self.angle]  
 **if** self.ball.x <= 0:  
 self.ball.x = 1  
 **else**:  
 self.ball.x = 599  
 **if** self.ball.y <= 0:  
 self.ball.y = 1  
 self.yDirection \*= -1  
  
 **for** paddle **in** self.paddle:  
 **if** paddle[0].colliderect(self.ball):  
 self.angle = paddle[1]  
 self.direction = -1  
 self.yDirection = -1  
 **break** check = self.ball.collidelist(self.blocks)  
 **if** check != -1:  
 block = self.blocks.pop(check)  
 pygame.mixer.Sound.play(crash\_sound)  
 **if** xMovement:  
 self.direction \*= -1  
 self.yDirection \*= -1  
 self.score += 1  
 **if** self.score == 35:  
 level2().main()  
 **if** self.ball.y > 600:  
 pygame.mixer.Sound.play(gameover\_sound)  
 end()  
  
 **def** paddleUpdate(self):  
  
 pos = pygame.mouse.get\_pos()  
 on = 0  
 **for** p **in** self.paddle:  
 p[0].x = pos[0] + 20 \* on  
 on += 1  
 **def** main(self):  
 pygame.mouse.set\_visible(**False**)  
 clock = pygame.time.Clock()  
 self.createBlocks()  
 background\_image= pygame.image.load(**"level1.jpg"**).convert()  
 **while True**:  
 clock.tick(20)*#as the level progresses the clock tick increases the fps* **for** event **in** pygame.event.get():  
 **if** event.type == QUIT:  
 sys.exit()  
 self.screen.blit(background\_image,[0,0])  
 self.paddleUpdate()  
 self.ballUpdate()  
  
 **for** block **in** self.blocks:  
 pygame.draw.rect(self.screen, (255,0,255), block)  
 **for** paddle **in** self.paddle:  
 pygame.draw.rect(self.screen, (255,255,255), paddle[0])  
 pygame.draw.rect(self.screen, (255,255,255), self.ball)  
 self.screen.blit(self.font.render(str(self.score), -1, (255,255,255)), (300, 550))  
 pygame.display.update()  
**class** level2:  
 **def** \_\_init\_\_(self):  
 self.screen = pygame.display.set\_mode((600, 600))  
 *#pygame.display.set\_mode((width, height)) - This will launch a window of the desired size* self.blocks = []  
 self.paddle = [[pygame.Rect(300, 500, 20, 10), 120],  
 [pygame.Rect(320, 500, 20, 10), 100],  
 [pygame.Rect(340, 500, 20, 10), 80],  
 [pygame.Rect(360, 500, 20, 10), 45],]  
 self.ball = pygame.Rect(300, 500 , 5, 5)  
 self.direction = -2  
 self.yDirection = -1  
 self.angle = 80  
 self.speeds = {  
 120:(-10, -3),  
 100:(-10, -8),  
 80:(10, -8),  
 45:(10, -3),  
 }  
 self.swap = {  
 120:45,  
 45:120,  
 100:80,  
 80:100,  
 }  
 pygame.font.init()  
 self.font = pygame.font.SysFont(**"Arial"**, 25)  
 self.score = 0  
  
 **def** createBlocks(self):  
 self.blocks = []  
 y = 50*#y axis of the block in the game* **for** \_\_ **in** range(200 // 20):  
 x = 210 *#x axis of the blocks in the game* **for** \_ **in** range(200 // 20 - 5): *#the 200 inside d bracket explains how many blocks r there in this stage of the game* block = pygame.Rect(x, y, 25, 10)  
 self.blocks.append(block)  
 x += 27  
 y += 12  
  
 **def** ballUpdate(self):  
 **for** \_ **in** range(2):  
 speed = self.speeds[self.angle]  
 xMovement = **True  
 if** \_:  
 self.ball.x += speed[0] \* self.direction  
 **else**:  
 self.ball.y += speed[1] \* self.direction \* self.yDirection  
 xMovement = **False  
 if** self.ball.x <= 0 **or** self.ball.x >= 600:  
 self.angle = self.swap[self.angle]  
 **if** self.ball.x <= 0:  
 self.ball.x = 1  
 **else**:  
 self.ball.x = 599  
 **if** self.ball.y <= 0:  
 self.ball.y = 1  
 self.yDirection \*= -1  
  
 **for** paddle **in** self.paddle:  
 **if** paddle[0].colliderect(self.ball):  
 self.angle = paddle[1]  
 self.direction = -1  
 self.yDirection = -1  
 **break** check = self.ball.collidelist(self.blocks)  
 **if** check != -1:  
 block = self.blocks.pop(check)  
 pygame.mixer.Sound.play(crash\_sound)  
 **if** xMovement:  
 self.direction \*= -1  
 self.yDirection \*= -1  
 self.score += 1  
 **if** self.score == 50:  
 winner()  
  
 **if** self.ball.y > 600:  
 pygame.mixer.Sound.play(gameover\_sound)  
 end()  
  
 **def** paddleUpdate(self):  
  
 pos = pygame.mouse.get\_pos()  
 on = 0  
 **for** p **in** self.paddle:  
 p[0].x = pos[0] + 20 \* on  
 on += 1  
 **def** main(self):  
 pygame.mouse.set\_visible(**False**)  
 clock = pygame.time.Clock()  
 self.createBlocks()  
 background\_image= pygame.image.load(**"LEVEL2.jpg"**).convert()  
 **while True**:  
 clock.tick(30)  
 **for** event **in** pygame.event.get():  
 **if** event.type == QUIT:  
 sys.exit()  
 self.screen.blit(background\_image,[0,0])  
 self.paddleUpdate()  
 self.ballUpdate()  
  
 **for** block **in** self.blocks:  
 pygame.draw.rect(self.screen, (255,0,255), block)  
 **for** paddle **in** self.paddle:  
 pygame.draw.rect(self.screen, (255,255,255), paddle[0])  
 pygame.draw.rect(self.screen, (255,255,255), self.ball)  
 self.screen.blit(self.font.render(str(self.score), -1, (255,255,255)), (400, 550))  
 pygame.display.update()  
  
**def** menu():  
 pygame.init()  
 screen= pygame.display.set\_mode((320,480))  
 pygame.display.set\_caption(**'Brick Breaker'**)  
 bg = pygame.image.load(**"1.JPG"**)  
 start = textrect(**"Arial "**,**"Start"**,30,100,130,255,255,255)  
 quit = textrect(**"Arial"**,**"Quit"**,30,100,300,255,255,255)  
 **while True** :  
 screen.blit(bg,(0,0))  
 text = Group(start,quit)  
 text.draw(screen)  
 a = event.wait()  
 **if** start.rect.collidepoint(mouse.get\_pos()):  
 start = textrect(**"Arial"**,**"Start"**,30,100,130,0,0,255)  
 **if** a.type == MOUSEBUTTONDOWN:  
 level1().main()  
 level2().main()  
  
 **else**:  
 start = textrect(**"Arial"**,**"Start"**,30,100,130,255,255,255)  
  
 **if** quit.rect.collidepoint(mouse.get\_pos()):  
 quit = textrect(**"Arial"**,**"Quit"**,30,100,300,0,255,0)  
 **if** a.type == MOUSEBUTTONDOWN:  
 pygame.quit()  
 **else**:  
 quit = textrect(**"Arial"**,**"Quit"**,30,100,300,255,255,255)  
  
 pygame.display.update()  
menu()  
  
  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 level1().main()  
  
*#reference list: GitHub. (2017). Max00355/Breakout. [online] Available at: https://github.com/Max00355/Breakout [Accessed 6 Nov. 2017].*