

**COMPUTER PROJECT**

**ON**

**“PYGAME”**



***SHIV ASHISH SCHOOL***

**SUBMITTED TO :** ***Honey Shivnani***

**SUBMITTED BY :** ***Dev Radadia* *and g Dheer Banker***

**CLASS :** ***XII (Science)***

**BATCH : *2020-21***



# ping-pong-xiiProj

PingPong - A project made by Dev Radadia and Dheer Banker

The aim of the project is to create a ping-pong game with basic 1v1 options and a smooth, intuitive gameplay.

Salient Features:

1) The project is divided into 5 packages:

i) image - Contains the images used in the project

ii) r - Contains the screen-wise strings, resources and font styles used in the project

iii) screens - Contains the logic for every screen in the project, with a module for every screen

iv) sound - Contains the sounds used in the project

v) sprites - Contains the different pygame drawables that have been used frequently in the project, like ball, paddle, button, etc.

2) All the GUI Elements in the project are developed solely using pygame, from scratch, in order to keep a consistent GUI

3) The project also includes various sounds in it, which are played, for example, when a button is clicked, or when the ball bounces.

4) There are a total of 6 screens in the project

i) About - Tells the user about the developers and the basic controls of the game

ii) Main Menu - The main screen that has options to go to the other screens.

iii) Player Names - Where players can enter their names and choose their colours

iv) Game - The game screen, where the players play.

v) Pause - The screen which comes up when players choose to pause the game

vi) EndGame - The screen which declares the winner of the game just played

5) All the screens are then bound and controlled using "The Game.py", the main controller code of this project.

Procedural Design

1. To start the application, click on **The Game.py** or run **The Game.py** via a Python console.
2. The main screen appears, which has three options:
   1. **Start**: Takes the user to the **Player Names** screen (details mentioned below).
   2. **About**: Takes the user to the **About** screen, which contains information about the project and its basic controls.
   3. **Quit**: Exits the application.
3. **Player Names** screen: Here, the players can set their respective names and choose their colours (default White). From here they can either **Return to Main Menu** or press **Enter** to move to the **Game** screen.
4. **Game** screen: After an initial countdown from 3 to 1, the game starts. Players can:
   1. Move their paddles using the respective controls (mentioned in **About** screen).
   2. Pause the game using the **Pause** button or by pressing **P**.
   3. Go to the main screen by pressing **Esc**.
5. **Pause** screen: Shows the current scores and has the options to **Resume, Return to Main Menu** or **Quit**.
6. **Endgame** screen: The screen which follows when the game ends, showing the winner’s name, and the options to **Play Again, Return to Main Menu,** or **Quit**.

The possible screens and transitions are shown in the image below.



# Modules Used

1. r.about
2. r.colors
3. r.endgame
4. r.font\_size
5. r.game
6. r.main
7. r.pause
8. r.playernames
9. screens.about
10. screens.endgame
11. screens.game
12. screens.main\_menu
13. screens.pause
14. screens.playernames
15. sprites.Ball
16. sprites.Border
17. sprites.Button
18. sprites.Label
19. sprites.Paddle
20. sprites.PauseButton
21. sprites.Textbox

# Functions Used

1. main
2. start\_menu
3. player\_details
4. start\_game
5. pause\_game
6. launch\_endgame
7. launch\_about
8. \_\_init\_\_
9. show\_menu
10. showAbout
11. setAboutText
12. setDisplay
13. PlayerNames
14. handleColorClick
15. getPlayer1Name
16. getColor1
17. getPlayer2Name
18. getColor2
19. setColorButtons
20. playersReset
21. gameReset
22. collides
23. play
24. countdown
25. setGameObjective
26. setMovables
27. setPaddleMargin
28. setPaddleSpeed
29. setBallResetMargin
30. setBounceBias
31. setPlayer1Name
32. setPlayer2Name
33. setBounceAcceleration
34. setPlayerColors
35. getWinnerColor
36. getWinnerName
37. getScores
38. pause\_game
39. setScores
40. setWinnerColor
41. setWinnerName
42. showEndScreen
43. bounce
44. same\_dir
45. crossed
46. ballReset
47. update
48. setResetMargin
49. setBallSpeed
50. getXSpeed
51. moveUp
52. moveDown
53. rectangle
54. draw
55. test\_surface
56. image
57. rect
58. setHighlightable
59. stayHighlighted
60. staysHighlighted
61. getTextRgb
62. handle\_event
63. define\_event
64. getText
65. setText

***The Game.py***

***r.\_\_init\_\_***

from . import game

from . import main

from . import endgame

from . import colors

from . import font\_size

from . import pause

from . import playernames

from . import about

\_\_all\_\_=['main','game', 'endgame', 'colors', 'font\_size', 'pause', 'playernames','about']

***r.about***

text\_about="This is a Pygame-Based Class-12 Project made by Dev Radadia and Dheer Banker.\n\n\nControls :-\nW and S : To move Left paddle Up and Down\nUp and Down (Arrows) : To move Right paddle Up and Down\n\n\nSet your name, choose your paddle colour, and PLAY ON !!! "

txt\_return\_btn="Return to Main Menu"

***r.colors***

BLACK = (0,0,0)

WHITE = (255,255,255)

BLUE = (19,244,239)

PINK = (240,0,255)

GREEN = (104,255,0)

YELLOW = (250,255,0)

RED =(255,0,92)

GOLD = (255,215,0)

SILVER = (192,192,192)

***r.endgame***

win\_statement=" Wins!"

play\_again\_btn\_txt="Play Again"

return\_to\_mainmenu\_button\_txt = "Return to Main Menu"

quit\_button\_txt = "Quit”

***r.font\_size***

xxxs = 20

xxs = 30

xs = 40

s = 50

m = 60

l = 80

xl = 100

xxl = 120

xxxl = 135

xxxxl = 200

***r.game***

BLACK = (0,0,0)

WHITE = (255,255,255)

BLUE = (106, 159, 181)

FPS = 60

SCREEN\_WIDTH = 900

SCREEN\_HEIGHT = 670

SCORE\_MARGIN = 70

PADDLE\_WIDTH = 12

PADDLE\_HEIGHT = 120

PADDLE\_MARGIN = 20

PADDLE\_BOUNCE\_BIAS = 80

PADDLE\_SPEED = 15

BALL\_WIDTH = 21

BALL\_HEIGHT = 21

BALL\_RESET\_Y\_MARGIN = 50

BALL\_BOUNCE\_ACC = 1.05

game\_obj\_txt = "Race to 10"

***r.main***

r\_title\_label\_txt = "Ping-Pong"

r\_start\_button\_txt = "Start"

r\_quit\_button\_txt = "Quit"

r\_about\_button\_txt = "About”

***r.pause***

paused\_label\_txt = "PAUSED"

resume\_button\_txt = "Resume"

return\_to\_mainmenu\_button\_txt = "Return to Main Menu"

quit\_button\_txt = "Quit"

***r.playernames***

playernames\_label\_txt = "Player Names"

p1\_label\_txt = "Player 1"

p2\_label\_txt = "Player 2"

name\_label\_txt = "Name :"

color\_blue\_label\_txt = "Blue"

color\_green\_label\_txt = "Green"

color\_yellow\_label\_txt = "Yellow"

color\_pink\_label\_txt = "Pink"

color\_red\_label\_txt = "Red"

return\_to\_mainmenu\_button\_txt = "Return to Main Menu"

enter\_button\_txt = "Enter"

***screens.\_\_init\_\_***

from . import game

from . import pause

from . import endgame

from . import main\_menu

from . import playernames

from . import about

\_\_all\_\_ = ['game', 'pause', 'endgame', 'main\_menu', 'playernames', 'about']

***screens.about***

import pygame

import pygame.freetype

from sprites.Border import \*

from sprites.Label import \*

from sprites.Button import \*

import r

CB\_RETURN=0

CB\_QUIT=-1

class AboutScreen():

def \_\_init\_\_(self,screen,abouttext,screen\_dimen,bg\_color,fg\_color,fontsize=r.font\_size.xxs, bg=None):

self.screen=screen

self.screen\_dimen=screen\_dimen

self.bg\_color=bg\_color

self.fg\_color=fg\_color

self.abouttext=abouttext

self.bgimg=bg

self.font=pygame.font.Font(None,fontsize)

self.clock=pygame.time.Clock()

self.setDisplay()

def showAbout(self):

exitw=False

buttons=[self.return\_btn]

while not exitw:

mouse\_up=False

for event in pygame.event.get():

if event.type==pygame.QUIT:

exitw=True

return CB\_QUIT

if event.type == pygame.MOUSEBUTTONUP and event.button == 1:

mouse\_up = True

keys=pygame.key.get\_pressed()

if keys[pygame.K\_ESCAPE]:

return CB\_RETURN

self.screen.fill(self.bg\_color)

if self.bgimg is not None:

self.screen.blit(self.bgimg,(0,0))

for button in buttons:

button\_action=button.update(pygame.mouse.get\_pos(),mouse\_up)

if button\_action is not None:

return button\_action

button.draw(self.screen)

border = Border()

border.rectangle(self.screen)

self.aboutLabel.draw()

pygame.display.flip()

self.clock.tick(30)

def setAboutText(self,abouttext):

self.abouttext=abouttext

self.aboutLabel=Label(self.screen,pygame.Rect(40,40,self.screen\_dimen[0]-80,self.screen\_dimen[1]-80),self.fg\_color,self.bg\_color,self.font,text=self.abouttext,lineSpacing=6)

def setDisplay(self):

self.return\_btn=Button((self.screen\_dimen[0]-220,self.screen\_dimen[1]-40),

text=r.about.txt\_return\_btn,

font\_size=r.font\_size.xxs,

bg\_rgb=self.bg\_color,

text\_rgb=self.fg\_color,

action=CB\_RETURN)

self.setAboutText(self.abouttext)

***screens.endgame***

import pygame

from sprites.Border import \*

from sprites.Label import \*

from sprites.Button import \*

import r

CB\_RETURN = 0

CB\_PAUSE = 1

CB\_ENDGAME = 2

CB\_QUIT = -1

CB\_PLAY = 4

class EndgameScreen():

def \_\_init\_\_(self, screen, win, screen\_dimen, bg\_color, fontsize = r.font\_size.xxl, bg=None):

self.screen = screen

self.screen\_dimen = screen\_dimen

self.bg\_color = bg\_color

self.win = win

self.font = pygame.font.Font("r\\font\_styles\Courier Bold.ttf",fontsize)

self.winnerName = ""

self.winnerColor = r.colors.WHITE

self.bgimg = bg

self.setDisplay()

def setWinnerColor(self, winnerColor):

self.winnerColor = winnerColor

self.winner\_label = Button(

center\_position=(r.game.SCREEN\_WIDTH/2, 100),

font\_size=r.font\_size.xxl,

bg\_rgb=r.colors.BLACK,

text\_rgb=self.winnerColor,

text=self.winnerName,

)

self.win\_label = Label(self.screen, pygame.Rect(225, 185, 1000 ,1000), self.winnerColor, self.bg\_color, self.font, text=self.win)

def setWinnerName(self, winnerName):

self.winnerName = winnerName

self.winner\_label = Button(

center\_position=(r.game.SCREEN\_WIDTH/2, 100),

font\_size=r.font\_size.xxl,

bg\_rgb=r.colors.BLACK,

text\_rgb=self.winnerColor,

text=self.winnerName,

)

self.win\_label = Label(self.screen, pygame.Rect(225, 185, 1000 ,1000), self.winnerColor, self.bg\_color, self.font, text=self.win)

def showEndScreen(self):

self.winner\_label.setHighlightable(False)

buttons = [self.winner\_label, self.play\_btn, self.return\_to\_mainmenu\_btn, self.quit\_btn]

while True:

mouse\_up = False

for event in pygame.event.get():

if event.type == pygame.QUIT:

return CB\_QUIT

if event.type == pygame.MOUSEBUTTONUP and event.button == 1:

mouse\_up = True

self.screen.fill(r.colors.BLACK)

if self.bgimg is not None:

self.screen.blit(self.bgimg,(0,0))

for button in buttons:

button\_action = button.update(pygame.mouse.get\_pos(), mouse\_up)

if button\_action is not None:

return button\_action

button.draw(self.screen)

border = Border()

border.rectangle(self.screen)

keys=pygame.key.get\_pressed()

if keys[pygame.K\_ESCAPE]:

return CB\_QUIT

self.win\_label.draw()

pygame.display.flip()

def setDisplay(self):

self.play\_btn = Button(

center\_position = (r.game.SCREEN\_WIDTH/2, 390),

font\_size = r.font\_size.m,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = r.endgame.play\_again\_btn\_txt,

action = CB\_PLAY

)

self.return\_to\_mainmenu\_btn = Button(

center\_position = (r.game.SCREEN\_WIDTH/2, 490),

font\_size = r.font\_size.m,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = r.endgame.return\_to\_mainmenu\_button\_txt,

action = CB\_RETURN

)

self.quit\_btn = Button(

center\_position = (r.game.SCREEN\_WIDTH/2, 590),

font\_size = r.font\_size.m,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = r.endgame.quit\_button\_txt,

action = CB\_QUIT,

)

***screens.game***

***screens.main\_menu***

***screens.pause***

import pygame

import r

from sprites.Border import \*

from sprites.Label import \*

from sprites.Button import \*

CB\_RETURN = 0

CB\_PLAY = 4

CB\_QUIT = -1

class PauseScreen():

def \_\_init\_\_(self, screen, pause, screen\_dimen, bg\_color, fg\_color, fontsize1 = r.font\_size.xxl, fontsize2 = r.font\_size.m, bg=None):

self.screen = screen

self.screen\_dimen = screen\_dimen

self.bg\_color = bg\_color

self.fg\_color = fg\_color

self.pause = pause

self.font1 = pygame.font.Font("r\\font\_styles\Courier Italic.ttf", fontsize1)

self.font2 = pygame.font.Font("r\\font\_styles\Courier.ttf", fontsize2)

self.scores = (0,0)

self.setDisplay()

self.bgimg = bg

def pause\_game(self):

buttons = [self.resume\_btn, self.quit\_btn, self.return\_to\_mainmenu\_btn]

while True:

mouse\_up = False

for event in pygame.event.get():

if event.type == pygame.MOUSEBUTTONUP and event.button == 1:

mouse\_up = True

self.screen.fill(r.game.BLACK)

if self.bgimg is not None:

self.screen.blit(self.bgimg,(0,0))

for button in buttons:

button\_action = button.update(pygame.mouse.get\_pos(), mouse\_up)

if button\_action is not None:

return button\_action

button.draw(self.screen)

keys = pygame.key.get\_pressed()

if keys[pygame.K\_r]:

return CB\_PLAY

border = Border()

border.rectangle(self.screen)

self.pause\_label.draw()

self.score\_label.draw()

pygame.display.flip()

def setDisplay(self):

self.pause\_label = Label(self.screen, pygame.Rect(230, 70, 1000 ,1000), self.fg\_color, self.bg\_color, self.font1, text=self.pause)

self.score\_label = Label(self.screen, pygame.Rect(360, 250, 1000 ,1000), self.fg\_color, self.bg\_color, self.font2, text=str(self.scores[0])+" : "+str(self.scores[1]))

self.resume\_btn = Button(

center\_position = (r.game.SCREEN\_WIDTH/2, 385),

font\_size = r.font\_size.m,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = r.pause.resume\_button\_txt,

action=CB\_PLAY,

)

self.return\_to\_mainmenu\_btn = Button(

center\_position = (r.game.SCREEN\_WIDTH/2, 485),

font\_size = r.font\_size.m,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = r.pause.return\_to\_mainmenu\_button\_txt,

action=CB\_RETURN,

)

self.quit\_btn = Button(

center\_position = (r.game.SCREEN\_WIDTH/2, 585),

font\_size = r.font\_size.m,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = r.pause.quit\_button\_txt,

action = CB\_QUIT,

)

def setScores(self,a):

self.scores=a

self.score\_label = Label(self.screen, pygame.Rect(360, 250, 1000 ,1000), self.fg\_color, self.bg\_color, self.font2, text=str(self.scores[0])+" : "+str(self.scores[1]))

***screens.playernames***

***sprites.\_\_init\_\_***

from . import Ball

from . import Paddle

from . import Border

from . import Button

from . import Textbox

from . import Label

\_\_all\_\_ = ['Ball', 'Paddle', 'Border', 'Button', 'Textbox', 'Label']

***sprites.Ball***

import pygame

import random

import math

ball\_color\_default = (255,255,255)

class Ball(pygame.sprite.Sprite):

def \_\_init\_\_(self, ball\_dimen, screen\_dimen, paddle\_dimen, score\_margin):

super().\_\_init\_\_()

self.image = pygame.Surface([ball\_dimen[0] , ball\_dimen[1]], pygame.SRCALPHA)

pygame.draw.circle(self.image, ball\_color\_default, (ball\_dimen[0]//2, ball\_dimen[0]//2),ball\_dimen[0]//2)

self.rect = self.image.get\_rect()

self.speed = 0

self.x = 0

self.y = 0

self.px = 0

self.py = 0

self.direction = 0

self.bounce\_bias = 0

self.reset\_margin = 0

self.ball\_dimen = ball\_dimen

self.screen\_dimen = screen\_dimen

self.paddle\_dimen = paddle\_dimen

self.score\_margin = score\_margin

self.bounceSound=pygame.mixer.Sound('sound/bounce1.wav')

self.crossedSound=pygame.mixer.Sound('sound/bounce2.wav')

self.ballReset()

def bounce(self,b\_param):

p\_dir=self.direction

self.direction = (180-self.direction)%360

if not self.same\_dir(p\_dir,self.direction + (b\_param/self.paddle\_dimen[1])\*self.bounce\_bias):

self.direction += (b\_param/self.paddle\_dimen[1])\*self.bounce\_bias

def same\_dir(self,dir1,dir2):

dir1=dir1%360

dir2=dir2%360

r1=-1

r2=-1

if (dir1<=90 and dir2>=0) or (dir1>=270 and dir1<=360):

r1=1

if (dir2<=90 and dir2>=0) or (dir2>=270 and dir2<=360):

r2=1

return r1==r2

def crossed(self,xcor):

return (((self.x+self.px)/2-xcor)\*((self.px+self.ppx)/2-self.getXSpeed()-xcor) < 0)

def ballReset(self):

self.speed = 4.0

self.y = random.randrange(self.reset\_margin + self.score\_margin , self.screen\_dimen[1] - self.reset\_margin)

self.x = self.screen\_dimen[0]/2 - self.ball\_dimen[0]/2

self.py = self.y

self.px = self.x

self.direction = random.randrange(-45,45)

if random.randrange(2) == 0:

self.direction += 180

def update(self):

rads = math.radians(self.direction)

self.ppx=self.px

self.ppy=self.py

self.px=self.x

self.py=self.y

self.x += math.cos(rads) \* self.speed

self.y -= math.sin(rads) \* self.speed

if self.x < -self.ball\_dimen[0]\*5 or self.x > self.screen\_dimen[0] + self.ball\_dimen[0]\*5:

self.crossedSound.play()

self.ballReset()

self.rect.x = int(self.x)

self.rect.y = int(self.y)

if self.y <= self.score\_margin:

self.bounceSound.play()

self.direction = (360-self.direction)%360

self.y = 1 + self.score\_margin

if self.y >= self.screen\_dimen[1] - self.ball\_dimen[1]:

self.bounceSound.play()

self.direction = (360-self.direction)%360

self.y = self.screen\_dimen[1] - self.ball\_dimen[1] - 1

def setBounceBias(self, bias):

self.bounce\_bias = bias

def setResetMargin(self, margin):

self.reset\_margin = margin

def setBallSpeed(self, speed):

self.speed = speed

def getXSpeed(self):

return math.cos(math.radians(self.direction)) \* self.speed

***sprites.Border***

import pygame

import r

class Border(pygame.sprite.Sprite):

def rectangle(self, screen):

pygame.draw.line(screen, r.colors.WHITE, [0,0],[r.game.SCREEN\_WIDTH,0], 5)

pygame.draw.line(screen, r.colors.WHITE, [r.game.SCREEN\_WIDTH,0],[r.game.SCREEN\_WIDTH,r.game.SCREEN\_HEIGHT], 5)

pygame.draw.line(screen, r.colors.WHITE, [r.game.SCREEN\_WIDTH,r.game.SCREEN\_HEIGHT],[0,r.game.SCREEN\_HEIGHT], 5)

pygame.draw.line(screen, r.colors.WHITE, [0,r.game.SCREEN\_HEIGHT],[0,0], 5)

***sprites.Button***

import pygame

from pygame.sprite import Sprite

from pygame.rect import Rect

def text\_surface(text, font\_size, text\_rgb, bg\_rgb):

pygame.init()

font = pygame.freetype.SysFont("Courier", font\_size, bold=True)

font.pad=True

surface, \_ = font.render(text=text, fgcolor=text\_rgb, bgcolor=bg\_rgb)

return surface.convert\_alpha()

class Button(Sprite):

def \_\_init\_\_(self, center\_position, text, font\_size, bg\_rgb, text\_rgb, action=None):

self.mouse\_over = False

self.text\_rgb=text\_rgb

default\_image = text\_surface(text=text, font\_size=font\_size, text\_rgb=text\_rgb, bg\_rgb=None)

selected\_image = text\_surface(text=text, font\_size=font\_size \* 1.2, text\_rgb=bg\_rgb, bg\_rgb=text\_rgb)

highlighted\_image = text\_surface(text=text, font\_size=font\_size \* 1.2, text\_rgb=text\_rgb, bg\_rgb=None)

self.images = [default\_image, selected\_image, highlighted\_image]

self.rects = [default\_image.get\_rect(center=center\_position),

selected\_image.get\_rect(center=center\_position),

highlighted\_image.get\_rect(center=center\_position)]

self.action = action

self.highlightable = True

self.stay\_highlighted=False

super().\_\_init\_\_()

@property

def image(self):

if self.stay\_highlighted:

return self.images[1]

if self.mouse\_over:

return self.images[2]

return self.images[0]

@property

def rect(self):

if self.stay\_highlighted:

return self.rects[1]

if self.mouse\_over:

return self.rects[2]

return self.rects[0]

def update(self, mouse\_pos, mouse\_up):

if not self.highlightable:

return

if self.rect.collidepoint(mouse\_pos):

self.mouse\_over = True

if mouse\_up:

click = pygame.mixer.Sound('sound/click3.wav')

click.play()

return self.action

else:

self.mouse\_over = False

def draw(self, surface):

surface.blit(self.image, self.rect)

def setHighlightable(self, highlightable):

self.highlightable = highlightable

def stayHighlighted(self,stay):

self.stay\_highlighted=stay

def staysHighlighted(self):

return self.stay\_highlighted

def getTextRgb(self):

return self.text\_rgb

***sprites.Label***

import pygame

class Label():

def \_\_init\_\_(self,screen,limitRect,fg\_color,bg\_color,font,text="",lineSpacing=-2):

self.limitRect=pygame.Rect(limitRect)

self.screen=screen

self.fg\_color=fg\_color

self.bg\_color=bg\_color

self.font=font

self.text=text

self.lineSpacing=lineSpacing

def draw(self):

fh=self.font.size("Tg")[1]

blit\_list=[]

j=0

temptxt=self.text

i=1

flag=True

while temptxt and (len(blit\_list)\*(fh+self.lineSpacing)<=self.limitRect.height):

if temptxt[0] == '\n':

for x in range(1,len(temptxt)):

if temptxt[x]!='\n':

break

blit\_list.append("")

temptxt=temptxt[x:]

continue

i=1

flag=True

while self.font.size(temptxt[:i])[0] < self.limitRect.width and i < len(temptxt) and flag:

i+=1

if temptxt[i-1]=='\n':

flag=False

if i<len(temptxt) and flag:

i=temptxt.rfind(" ",0,i)+1

if not flag:

i-=1

blit\_list.append(temptxt[:i])

temptxt=temptxt[i:]

y=self.limitRect.top

for st in blit\_list:

image=self.font.render(st,1,self.fg\_color)

self.screen.blit(image,(self.limitRect.left,y))

y+=fh+self.lineSpacing

def setText(self,text):

self.text=text

***sprites.Paddle***

import pygame

class Paddle(pygame.sprite.Sprite):

def \_\_init\_\_(self, screen\_dimen, paddle\_dimen, score\_margin, color):

super().\_\_init\_\_()

self.image = pygame.Surface([paddle\_dimen[0],paddle\_dimen[1]])

self.image.fill(color)

self.screen\_dimen=screen\_dimen

self.paddle\_dimen=paddle\_dimen

self.score\_margin=score\_margin

self.rect = self.image.get\_rect()

def moveUp(self,pixels):

self.rect.y -= pixels

if self.rect.y < self.score\_margin + 3:

self.rect.y = self.score\_margin + 3

def moveDown(self,pixels):

self.rect.y += pixels

if self.rect.y > self.screen\_dimen[1] - self.paddle\_dimen[1]:

self.rect.y = self.screen\_dimen[1] - self.paddle\_dimen[1]

***sprites.PauseButton***

import pygame

from pygame.sprite import Sprite

from pygame.rect import Rect

from r.game import \*

class PauseButton(Sprite):

def \_\_init\_\_(self, action = None):

pygame.init()

self.pausebutton = pygame.image.load('image\image.png')

self.pb = pygame.transform.scale(self.pausebutton, (50,50))

self.mouse\_over = False

self.rect = self.pb.get\_rect(center = (SCREEN\_WIDTH/2, 35))

self.action = action

self.clickSound=pygame.mixer.Sound('sound/click3.wav')

super().\_\_init\_\_()

def update(self, mouse\_pos, mouse\_up):

if self.rect.collidepoint(mouse\_pos):

self.mouse\_over = True

if mouse\_up:

self.clickSound.play()

return self.action

else:

self.mouse\_over = False

def draw(self, surface):

surface.blit(self.pb, self.rect)

***sprites.Textbox***

import pygame

import r

pygame.init()

class Textbox:

def \_\_init\_\_(self, x, y, width, height, fontsize=r.font\_size.xxs+1, maxlength=12, text='', textcolor=r.colors.BLACK, inactivebordercolor=r.colors.SILVER, activebordercolor=r.colors.GOLD):

self.rect = pygame.Rect(x, y, width, height)

self.color = inactivebordercolor

self.inactivecolor = inactivebordercolor

self.textcolor = textcolor

self.activecolor = activebordercolor

self.maxlength = maxlength

self.text = text

self.fontsize = fontsize

self.font=pygame.font.Font(None, self.fontsize)

self.txt\_surface = self.font.render(text, True, self.color)

self.txt\_surface.set\_alpha(0)

self.active = False

self.repeater\_count={}

self.nr\_init=400

self.nr\_inter=35

self.clock=pygame.time.Clock()

def handle\_event(self, events):

for event in events:

self.define\_event(event)

for k in self.repeater\_count:

self.repeater\_count[k][0] += self.clock.get\_time()

if self.repeater\_count[k][0] >= self.nr\_init:

self.repeater\_count[k][0] = (self.nr\_init - self.nr\_inter)

e\_key, e\_uni = k, self.repeater\_count[k][1]

pygame.event.post(pygame.event.Event(pygame.KEYDOWN, key=e\_key, unicode=e\_uni))

self.text\_surface = self.font.render(self.text, True, self.textcolor)

self.clock.tick()

def define\_event(self, event):

if event.type == pygame.MOUSEBUTTONDOWN:

if self.rect.collidepoint(event.pos):

click = pygame.mixer.Sound('sound/click3.wav')

click.play()

self.active = True

self.color = self.activecolor

else:

self.active = False

self.color = self.inactivecolor

if self.active:

if event.type == pygame.KEYDOWN:

if event.key not in self.repeater\_count:

self.repeater\_count[event.key]=[0,event.unicode]

if event.key == pygame.K\_DELETE:

self.text = ''

elif event.key == pygame.K\_BACKSPACE:

self.text = self.text[:-1]

elif event.key in [pygame.K\_TAB, pygame.K\_ESCAPE]:

pass

else:

if len(self.text) < self.maxlength:

self.text += event.unicode

elif event.type==pygame.KEYUP:

del self.repeater\_count[event.key]

def draw(self, screen):

pygame.draw.line(screen, r.colors.WHITE, (self.rect.x+2,self.rect.y+self.rect.height/2), (self.rect.x+self.rect.width-2,self.rect.y+self.rect.height/2), 30)

pygame.draw.rect(screen, self.color, self.rect, 4)

screen.blit(self.text\_surface, (self.rect.x+5, self.rect.y+5))

def getText(self):

return self.text



























