

**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***



***Index***

* Certificate
* Acknowledgement
* Python Introduction
* System Requirements
* Game Logo
* Project Overview
* Procedural Design
* Modules Used
* Functions Used
* Source Code
* Screenshots
* Limitations
* Bibliography

***Certificate***

***Acknowledgement***

***Python Introduction***

* ****What is Python?****

**Python is a popular programming language. It was created by Guido van Rossum, and released in 1989.**

***Interesting fact* : Python is named after the comedy television show “Monty Python’s Flying Circus”. It is not named after the Python snake.**

**It is used for :**

* web development (server-side),
* software development,
* mathematics,
* system scripting
* data analysis,
* game development,
* desktop applications
* **What can Python do?**
* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.
* **Why Python?**
* Python works on different platforms (Windows, Mac, Linux, etc.)
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-oriented way or a functional way.
* **Python Syntax compared to other programming languages**
* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.
* **Python-Pluses**

1. Easy to Use
2. Expressive Language
3. Interpreted Language
4. Free and Open Source
5. Cross-Platform Language
6. Extensive Libraries
7. Object-Oriented Language
8. Improvised Productivity
9. Advanced features
10. Automatic memory management

* **Python-Minuses (So Human Like)**

1. Not the Fastest Language
2. Lesser Libraries than C, Java, Perl
3. Not Strong on Type­Binding
4. Not Easily Convertible
5. High Memory Consumption
6. Weak in Mobile Development
7. Underdeveloped Database Access Layers

***System Requirements***

* **Hardware Requirements**
* *Operating System* : Supports Linux, Windows (95, 98, ME, 2000, XP, Vista, 64-bit Windows), Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX
* *RAM* : 500MB
* *Graphics Card* : AMD Radeon HD 5450
* *CPU* : Intel Pentium 4 2.00GHz
* *HDD Space* : 50MB
* **Software Requirements**
* *IDE* : A Python IDE (eg. Python IDLE, Spyder, Pycharm, Atom, etc.)
* *Python Modules* : Pygame, Numpy, Random and Math

***Game Logo***



### ***Project Overview***

PingPong - A project made by Dev Radadia and Dheer Banker

The aim of the project is to create a ping-pong game with two game-modes – ‘1 Player’ and ‘2 Players’ with a smooth, intuitive gameplay.

* ***Salient Features* :**

1. The project is divided into 6 packages:
   1. ***ai*** – Contains the code governing the AI used in 1 player game-mode
   2. ***image*** – Contains the images used in the project
   3. ***r*** – Contains the screen-wise strings, resources and font styles used in aaathe project
   4. ***screens*** – Contains the code governing every screen in the project, aaaaaaaaawith a module for every screen
   5. ***sound*** – Contains the sounds used in the project
   6. ***sprites*** – Contains the different pygame drawables that have been aaaaaaaaused in the project
2. All the GUI Elements in the project are developed solely using pygame from scratch, in order to keep the GUI consistent throughout the application.
3. The project also includes various sounds in it, which are played when a button is clicked, or when the ball bounces with the paddle or the wall, etc.
4. There are a total of 6 screens in the project :
   1. ***About*** – Tells the user about the developers and the basic controls of aaaaaaaathe game
   2. ***Main*** ***Menu*** – The main screen that has options to go to the other aaaaaaaaaaaaascreens
   3. ***PlayerNames*** – The screen where players can enter their names and aaaaaaaaaaaaaaachoose their paddle colours
   4. ***Game*** – The game screen, where the players can play
   5. ***Pause*** – The screen which comes up when the game is paused, by aaaaaaaapressing ***Pause button*** or by pressing ***P***
   6. ***EndGame*** – The screen which declares the winner of the game that aaaaaaaaaaa was just played
5. Finally, all the screens are bound together by the controller code “***The Game.py***”.

***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 aaaaaaabelow).
   2. **About** : Takes the user to the **About** screen, which contains aaaaaaaainformation 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 aaaaaaaaaaaaaaaaaaaaachoose their colours (default White). From here they aaaaaaaaaaaaaaaaaaaaacan either **Return to Main Menu** or press **Enter** to aaaaaaaaaaaaaaaaaaaaamove 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***

1. **Pause** screen : Shows the current scores and has the options to **Resume, aaaaaaaaaaaaReturn to Main Menu** or **Quit**.
2. **Endgame** screen : The screen which follows when the game ends, showing aaaaaaaaaaaaaaaaathe winner’s name, and the options to **Play Again, aaaaaaaaaaaaaaaReturn to Main Menu,** or **Quit**.

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

***Modules Used***

*User-defined Modules*

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

*Built-In Modules*

1. pygame
2. numpy
3. random
4. math

***Functions Used***

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

***Source Code***

***The Game.py***

import pygame

from enum import Enum

import r

import screens

def main():

global game, main\_menu, player\_names, pause\_screen, endgame\_screen, about\_screen

pygame.mixer.pre\_init(22050, -16, 1, 512)

pygame.init()

pygame.mixer.quit()

pygame.mixer.init(22050, -16, 1, 512)

pygame.display.set\_caption(r.main.r\_title\_label\_txt)

default\_bg=pygame.image.load("image\\bg\_default.jpg")

screen = pygame.display.set\_mode((r.game.SCREEN\_WIDTH, r.game.SCREEN\_HEIGHT), pygame.FULLSCREEN)

game\_screen = Screen.MENU

game=screens.game.GameScreen(screen, (r.game.SCREEN\_WIDTH, r.game.SCREEN\_HEIGHT), r.colors.BLACK, r.colors.WHITE, r.game.SCORE\_MARGIN, r.game.FPS)

game.setPaddleMargin(r.game.PADDLE\_MARGIN)

game.setPaddleSpeed(r.game.PADDLE\_SPEED)

game.setBallResetMargin(r.game.BALL\_RESET\_Y\_MARGIN)

game.setBounceBias(r.game.PADDLE\_BOUNCE\_BIAS)

game.setBounceAcceleration(r.game.BALL\_BOUNCE\_ACC)

game.setGameObjective(r.game.game\_obj\_txt)

game.setMovables(r.game.BALL\_HEIGHT, (r.game.PADDLE\_WIDTH, r.game.PADDLE\_HEIGHT), r.colors.WHITE, r.colors.WHITE)

main\_menu=screens.main\_menu.MainMenuScreen(screen, r.main.r\_title\_label\_txt, (r.game.SCREEN\_WIDTH, r.game.SCREEN\_HEIGHT), r.colors.BLACK, r.colors.WHITE, bg=default\_bg)

player\_names=screens.playernames.PlayerNamesScreen(screen, r.playernames.playernames\_label\_txt, r.playernames.p\_label\_txt, r.playernames.ai\_label\_txt, r.playernames.p1\_label\_txt, r.playernames.p2\_label\_txt, r.playernames.name\_label\_txt, (r.game.SCREEN\_WIDTH, r.game.SCREEN\_HEIGHT), r.colors.BLACK, r.colors.WHITE, bg=default\_bg)

pause\_screen=screens.pause.PauseScreen(screen, r.pause.paused\_label\_txt, (r.game.SCREEN\_WIDTH, r.game.SCREEN\_HEIGHT), r.colors.BLACK, r.colors.WHITE, bg=default\_bg)

endgame\_screen=screens.endgame.EndgameScreen(screen, r.endgame.win\_statement, (r.game.SCREEN\_WIDTH, r.game.SCREEN\_HEIGHT), r.colors.BLACK, bg=default\_bg)

about\_screen=screens.about.AboutScreen(screen, r.about.text\_about, (r.game.SCREEN\_WIDTH, r.game.SCREEN\_HEIGHT), r.colors.BLACK, r.colors.WHITE, bg=default\_bg)

while True:

if game\_screen == Screen.MENU:

game\_screen = start\_menu(screen)

if game\_screen == Screen.PLAYER1:

game\_screen = player1\_details(screen)

if game\_screen == Screen.PLAYERS2:

game\_screen = players2\_details(screen)

if game\_screen == Screen.PLAYGAME:

game\_screen = start\_game(screen, game)

if game\_screen == Screen.PAUSE:

game\_screen = pause\_game(screen)

if game\_screen == Screen.ENDGAME:

game\_screen = launch\_endgame(screen)

if game\_screen == Screen.ABOUT:

game\_screen = launch\_about(screen)

if game\_screen == Screen.QUIT:

pygame.quit()

return

def start\_menu(screen):

new\_screen=main\_menu.show\_menu()

game.gameReset()

if new\_screen == screens.main\_menu.CB\_QUIT:

return Screen.QUIT

if new\_screen == screens.main\_menu.CB\_1PLAYER:

return Screen.PLAYER1

if new\_screen == screens.main\_menu.CB\_2PLAYERS:

return Screen.PLAYERS2

if new\_screen == screens.main\_menu.CB\_ABOUT:

return Screen.ABOUT

return Screen.QUIT

def player1\_details(screen):

new\_screen=player\_names.Player1\_Name()

if new\_screen == screens.playernames.CB\_PLAY\_AI:

game.setPlayer1Name(player\_names.getPlayer1Name())

game.setPlayer2Name(player\_names.getPlayer2Name())

game.enableAi()

game.setMovables(r.game.BALL\_HEIGHT, (r.game.PADDLE\_WIDTH, r.game.PADDLE\_HEIGHT), player\_names.getColor1(), player\_names.getColor2())

return Screen.PLAYGAME

if new\_screen == screens.playernames.CB\_RETURN:

return Screen.MENU

return Screen.QUIT

def players2\_details(screen):

new\_screen=player\_names.Players2\_Names()

if new\_screen == screens.playernames.CB\_PLAY:

game.setPlayer1Name(player\_names.getPlayer1Name())

game.setPlayer2Name(player\_names.getPlayer2Name())

game.setMovables(r.game.BALL\_HEIGHT, (r.game.PADDLE\_WIDTH, r.game.PADDLE\_HEIGHT), player\_names.getColor1(), player\_names.getColor2())

game.disableAi()

return Screen.PLAYGAME

if new\_screen == screens.playernames.CB\_RETURN:

return Screen.MENU

return Screen.QUIT

def start\_game(screen,game):

new\_screen = game.play()

if new\_screen == screens.game.CB\_PAUSE:

return Screen.PAUSE

elif new\_screen == screens.game.CB\_ENDGAME:

return Screen.ENDGAME

elif new\_screen == screens.game.CB\_RETURN:

return Screen.MENU

elif new\_screen == screens.game.CB\_QUIT:

return Screen.QUIT

return Screen.MENU

def pause\_game(screen):

global game,pause\_screen

pause\_screen.setScores(game.getScores())

new\_screen = pause\_screen.pause\_game()

if new\_screen == screens.pause.CB\_QUIT:

return Screen.QUIT

if new\_screen == screens.pause.CB\_PLAY:

return Screen.PLAYGAME

if new\_screen == screens.game.CB\_RETURN:

return Screen.MENU

return Screen.MENU

def launch\_endgame(screen):

global endgame\_screen,game

endgame\_screen.setWinnerName(game.getWinnerName())

endgame\_screen.setWinnerColor(game.getWinnerColor())

new\_screen=endgame\_screen.showEndScreen()

if new\_screen==screens.endgame.CB\_PLAY:

return Screen.PLAYGAME

if new\_screen==screens.endgame.CB\_RETURN:

return Screen.MENU

if new\_screen == screens.pause.CB\_QUIT:

return Screen.QUIT

return Screen.MENU

def launch\_about(screen):

global about\_screen

new\_screen=about\_screen.showAbout()

if new\_screen==screens.about.CB\_RETURN:

return Screen.MENU

if new\_screen==screens.about.CB\_QUIT:

return Screen.QUIT

return Screen.MENU

class Screen(Enum):

QUIT=-1

MENU=0

PLAYGAME=1

PAUSE=2

ENDGAME=3

PLAYER1=4

PLAYERS2=5

ABOUT=6

if \_\_name\_\_=="\_\_main\_\_":

main()

***ai.\_\_init\_\_***

from . import fnn

from . import qlearner

\_\_all\_\_=['fnn','qlearner']

***ai.fnn***

#FeedForward Neural Network

import numpy as np

import random

from . import functions

class FNN():

def \_\_init\_\_(self, num\_neurons, activation=functions.sigmoid()):

self.num\_neurons=num\_neurons

self.num\_layers=len(num\_neurons)

self.acfunc=activation

self.reset()

def compute\_oup(self,inp):

tmpinp=inp

for bi,wei in zip(self.bias,self.weights):

tmpinp=self.acfunc.f(np.dot(wei,tmpinp)+bi)

return tmpinp

def reset(self):

self.bias=[np.random.randn(size,1) for size in self.num\_neurons[1:]]

self.weights=[np.random.randn(size2,size1)/np.sqrt(size1) for size1,size2 in zip(self.num\_neurons[:-1],self.num\_neurons[1:])]

def setBias(self,bias):

self.bias=bias

def setWeights(self,weights):

self.weights=weights

***ai.functions***

import numpy as np

class sigmoid():

def f(self,z):

return 1.0/(1.0+np.exp(-z))

def fp(self,z):

return self.f(z)\*(1.0-self.f(z))

class elu():

def \_\_init\_\_(self,alpha):

self.alpha=alpha

def f(self,z):

# return z if z>=0 else self.alpha\*(np.exp(z)-1)

return np.where(z>0,z,self.alpha\*(np.exp(z)-1))

def fp(self,z):

# return 1 if z>0 else self.alpha\*np.exp(z)

return np.where(z>0,1,self.alpha\*np.exp(z))

***ai.qlearner***

import numpy as np

import random

from ai.fnn import \*

from ai import functions

class QLearner():

def \_\_init\_\_(self, num\_actions, input\_size):

self.num\_actions=num\_actions

self.input\_size=input\_size

self.brain= FNN([input\_size,100,100,num\_actions],activation=functions.elu(1.0))

def updateIntent(self,state):

biggest=-1000.0

newAction=0

qvals=self.brain.compute\_oup(self.createBrainArray(state))

for action in range(self.num\_actions):

if qvals[action][0]>biggest:

biggest=qvals[action][0]

newAction=action

self.intent=newAction

def quadraticCostDerivative(self, good, almostgood):

return almostgood-good

def createBrainArray(self,state):

return np.asarray([[state[i]] for i in range(self.input\_size)])

def loadMemory(self, identifier):

b=None

w=None

try:

w=np.load(identifier+"\_w.npy",allow\_pickle=True)

b=np.load(identifier+"\_b.npy",allow\_pickle=True)

except IOError:

print("No memory/bad memory found, creating fresh ones...\n")

return

self.brain.setBias(b)

self.brain.setWeights(w)

def getIntent(self):

return self.intent

***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 = 60

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***

import pygame

from sprites.Border import \*

from sprites.Paddle import \*

from sprites.Ball import \*

from sprites.Label import \*

from sprites.Button import \*

from sprites.PauseButton import \*

from ai.qlearner import \*

import r

from r.main import \*

from r.game import \*

CB\_RETURN = 0

CB\_PAUSE = 1

CB\_ENDGAME = 2

CB\_QUIT = -1

IND\_BVELX=0

IND\_BVELY=1

IND\_BVELXP=10

IND\_BVELYP=11

IND\_BPOSX=2

IND\_BPOSY=3

IND\_BPOSXP=6

IND\_BPOSYP=7

IND\_PPOSX=4

IND\_PPOSY=5

IND\_PPOSXP=8

IND\_PPOSYP=9

I\_UP=1

I\_DOWN=0

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

class GameScreen():

def \_\_init\_\_(self, screen, screen\_dimen, bg\_color, fg\_color, score\_margin, fps):

self.screen=screen

self.screen\_dimen = screen\_dimen

self.bg\_color = bg\_color

self.fg\_color = fg\_color

self.score\_margin=score\_margin

self.fps=fps

self.border = Border()

self.p1Name="Player1"

self.p2Name="Player2"

self.p2ai=False

self.ai=QLearner(2,12)

self.ai.loadMemory("mem")

self.winnerName="Winner"

self.winnerColor=fg\_color\_default

self.color1=fg\_color\_default

self.color2=fg\_color\_default

self.game\_obj = ""

self.bounce\_acceleration = 1

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

self.font2 = pygame.font.Font("r\\font\_styles\Courier Bold Italic.ttf", r.font\_size.xl)

self.font3 = pygame.font.Font(None,r.font\_size.l)

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

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

def gameReset(self):

self.score1=0

self.score2=0

self.paddle1.rect.x = self.paddle\_margin

self.paddle1.rect.y = self.screen\_dimen[1]//2 - self.paddle\_dimen[1]//2 + self.score\_margin//2

self.paddle2.rect.x = SCREEN\_WIDTH - PADDLE\_WIDTH - PADDLE\_MARGIN

self.paddle2.rect.y = self.screen\_dimen[1]//2 - self.paddle\_dimen[1]//2 + self.score\_margin//2

self.ball.ballReset()

def collides(self):

if (self.ball.x <= self.paddle1.rect.x + self.paddle\_dimen[0] and self.ball.x >= self.paddle1.rect.x + self.paddle\_dimen[0] - self.ball.speed\*3) and (self.ball.y + self.ball\_dimen[1] >= self.paddle1.rect.y and self.ball.y <=self.paddle1.rect.y + self.paddle\_dimen[1]):

return 1

if (self.ball.x >= self.paddle2.rect.x - self.ball.ball\_dimen[0] and self.ball.x <= self.paddle2.rect.x - self.ball.ball\_dimen[0] + self.ball.speed\*3) and (self.ball.y + self.ball\_dimen[1] >= self.paddle2.rect.y and self.ball.y <=self.paddle2.rect.y + self.paddle\_dimen[1]):

return 2

return 0

def play(self):

clock = pygame.time.Clock()

self.ball.setResetMargin(self.ball\_reset\_margin)

self.ball.setBounceBias(self.bounce\_bias)

self.ball.update()

movingsprites = pygame.sprite.Group()

movingsprites.add(self.paddle1)

movingsprites.add(self.paddle2)

movingsprites.add(self.ball)

pausebtn = PauseButton(action = CB\_PAUSE)

buttons = [pausebtn]

self.countdown()

exit\_window = False

while not exit\_window:

mouse\_up=False

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

for event in pygame.event.get():

if event.type == pygame.QUIT:

exit\_window = True

return CB\_QUIT

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

mouse\_up = True

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)

self.border.rectangle(self.screen)

self.ball.update()

if self.collides() == 1:

self.collideSound.play()

diff = (self.paddle1.rect.y + self.paddle\_dimen[1]/2) - (self.ball.rect.y+self.ball\_dimen[1]/2)

self.ball.x = self.paddle\_margin+self.paddle\_dimen[0] + 2

self.ball.bounce(diff)

self.ball.speed = self.ball.speed\*self.bounce\_acceleration

if self.collides() == 2:

self.collideSound.play()

diff = (self.paddle2.rect.y + self.paddle\_dimen[1]/2) - (self.ball.rect.y+self.ball\_dimen[1]/2)

self.ball.x = self.screen\_dimen[0] - (self.paddle\_margin+self.ball\_dimen[0]+self.paddle\_dimen[0]+2)

self.ball.bounce(-diff)

self.ball.speed = self.ball.speed\*self.bounce\_acceleration

if self.ball.crossed(self.paddle\_margin//2):

self.score2+=1

if self.ball.crossed(self.screen\_dimen[0]-self.paddle\_margin//2):

self.score1+=1

keys = pygame.key.get\_pressed()

if keys[pygame.K\_w]:

self.paddle1.moveUp(self.paddle\_speed)

if keys[pygame.K\_s]:

self.paddle1.moveDown(self.paddle\_speed)

if self.p2ai:

self.ai.updateIntent(self.getGameState())

aimove=self.ai.getIntent()

if aimove==I\_UP:

self.paddle2.moveUp(self.paddle\_speed)

if aimove==I\_DOWN:

self.paddle2.moveDown(self.paddle\_speed)

else:

if keys[pygame.K\_UP]:

self.paddle2.moveUp(self.paddle\_speed)

if keys[pygame.K\_DOWN]:

self.paddle2.moveDown(self.paddle\_speed)

if keys[pygame.K\_ESCAPE]:

exit\_window = True

self.gameReset()

return CB\_RETURN

if keys[pygame.K\_p]:

return CB\_PAUSE

movingsprites.update()

pygame.draw.line(self.screen,r.colors.WHITE,[self.screen\_dimen[0]//2,self.score\_margin],[self.screen\_dimen[0]//2,self.screen\_dimen[1]],5)

pygame.draw.line(self.screen,r.colors.WHITE,[0,self.score\_margin],[self.screen\_dimen[0],self.score\_margin],5)

movingsprites.draw(self.screen)

text1 = self.font3.render(str(self.score1),1,r.colors.WHITE)

self.screen.blit(text1,(int(self.screen\_dimen[0]/4),10))

text2 = self.font3.render(str(self.score2),1,r.colors.WHITE)

self.screen.blit(text2,(3\*int(self.screen\_dimen[0]/4),10))

if self.score1 == 10 or self.score2 == 10:

self.crossedSound.play()

if self.score1==10:

self.winnerName=self.p1Name

self.winnerColor=self.color1

if self.score2==10:

self.winnerName=self.p2Name

self.winnerColor=self.color2

self.gameReset()

return CB\_ENDGAME

pygame.display.flip()

clock.tick(self.fps)

def countdown(self):

clock = pygame.time.Clock()

three = Label(self.screen, pygame.Rect(380, 240, 1000 ,1000), self.fg\_color, self.bg\_color, self.font1, text="3")

two = Label(self.screen, pygame.Rect(380, 240, 1000 ,1000), self.fg\_color, self.bg\_color, self.font1, text="2")

one = Label(self.screen, pygame.Rect(380, 240, 1000 ,1000), self.fg\_color, self.bg\_color, self.font1, text="1")

go = Label(self.screen, pygame.Rect(285, 240, 1000 ,1000), self.fg\_color, self.bg\_color, self.font1, text="GO!")

game\_objective = Label(self.screen, pygame.Rect(145, 490, 1000 ,1000), self.fg\_color, self.bg\_color, self.font2, text=self.game\_obj)

for i in range(1,self.fps\*4+1):

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

self.border.rectangle(self.screen)

game\_objective.draw()

num = 4 - i//self.fps

if num == 1:

go.draw()

elif num == 2:

one.draw()

elif num == 3:

two.draw()

elif num == 4:

three.draw()

pygame.display.flip()

clock.tick(self.fps)

def setGameObjective(self, game\_obj):

self.game\_obj=game\_obj

def setMovables(self, ball\_radius, paddle\_dimen, color1, color2):

self.paddle\_dimen=paddle\_dimen

self.color1=color1

self.color2=color2

self.ball\_dimen=(ball\_radius,ball\_radius)

self.ball=Ball(self.ball\_dimen, self.screen\_dimen, self.paddle\_dimen, self.score\_margin)

self.ball\_group=pygame.sprite.Group()

self.ball\_group.add(self.ball)

self.paddle1=Paddle(self.screen\_dimen, self.paddle\_dimen, self.score\_margin, self.color1)

self.paddle2=Paddle(self.screen\_dimen, self.paddle\_dimen, self.score\_margin, self.color2)

self.gameReset()

def setPaddleMargin(self, paddle\_margin):

self.paddle\_margin=paddle\_margin

def setPaddleSpeed(self, paddle\_speed):

self.paddle\_speed=paddle\_speed

def setBallResetMargin(self, ball\_reset\_margin):

self.ball\_reset\_margin=ball\_reset\_margin

def setBounceBias(self, bounce\_bias):

self.bounce\_bias=bounce\_bias

def setPlayer1Name(self, p1Name):

self.p1Name=p1Name

def setPlayer2Name(self, p2Name):

self.p2Name=p2Name

def setBounceAcceleration(self, bounce\_acceleration):

self.bounce\_acceleration = bounce\_acceleration

def setPlayerColors(self,color1,color2):

self.color1=color1

self.color2=color2

self.paddle1=Paddle(self.screen\_dimen, self.paddle\_dimen, self.score\_margin, self.color1)

self.paddle2=Paddle(self.screen\_dimen, self.paddle\_dimen, self.score\_margin, self.color2)

def enableAi(self):

self.p2ai=True

def disableAi(self):

self.p2ai=False

def getWinnerColor(self):

return self.winnerColor

def getWinnerName(self):

return self.winnerName

def getScores(self):

return (self.score1,self.score2)

def getGameState(self):

new\_state2=[None for i in range(12)]

new\_state2[IND\_BVELX]=self.ball.getXSpeed()/self.screen\_dimen[0]

new\_state2[IND\_BVELY]=self.ball.getYSpeed()/self.screen\_dimen[1]

new\_state2[IND\_BVELXP]=(-new\_state2[IND\_BVELX])

new\_state2[IND\_BVELYP]=(-new\_state2[IND\_BVELY])

new\_state2[IND\_BPOSX]=self.ball.x/self.screen\_dimen[0]

new\_state2[IND\_BPOSY]=self.ball.y/self.screen\_dimen[1]

new\_state2[IND\_BPOSXP]=1 - new\_state2[IND\_BPOSX]

new\_state2[IND\_BPOSYP]=1 - new\_state2[IND\_BPOSY]

new\_state2[IND\_PPOSX]=self.paddle2.rect.x/self.screen\_dimen[0]

new\_state2[IND\_PPOSY]=self.paddle2.rect.y/self.screen\_dimen[1]

new\_state2[IND\_PPOSXP]=1 - new\_state2[IND\_PPOSX]

new\_state2[IND\_PPOSYP]=1 - new\_state2[IND\_PPOSY]

return new\_state2

***screens.main\_menu***

import pygame

import pygame.freetype

import r

from sprites.Border import \*

from sprites.Label import \*

from sprites.Button import \*

CB\_1PLAYER = 1

CB\_2PLAYERS = 2

CB\_QUIT = -1

CB\_ABOUT = 7

class MainMenuScreen():

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

self.screen = screen

self.screen\_dimen = screen\_dimen

self.bg\_color = bg\_color

self.fg\_color = fg\_color

self.gamename = gamename

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

self.bgimg = bg

def show\_menu(self):

game\_name = Label(self.screen, pygame.Rect(90, 100, 1000 ,1000), self.fg\_color, self.bg\_color, self.font, text=self.gamename)

Player1\_btn = Button(

center\_position=(r.game.SCREEN\_WIDTH/4, 430),

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

bg\_rgb=r.colors.BLACK,

text\_rgb=r.colors.WHITE,

text=r.main.r\_1Player\_txt,

action=CB\_1PLAYER,

)

Players2\_btn = Button(

center\_position=(3\*(r.game.SCREEN\_WIDTH/4), 430),

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

bg\_rgb=r.colors.BLACK,

text\_rgb=r.colors.WHITE,

text=r.main.r\_2Players\_txt,

action=CB\_2PLAYERS,

)

about\_btn=Button(

center\_position=(r.game.SCREEN\_WIDTH/2, 515),

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

bg\_rgb=r.colors.BLACK,

text\_rgb=r.colors.WHITE,

text=r.main.r\_about\_button\_txt,

action=CB\_ABOUT,

)

quit\_btn = Button(

center\_position=(r.game.SCREEN\_WIDTH/2, 600),

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

bg\_rgb=r.colors.BLACK,

text\_rgb=r.colors.WHITE,

text=r.main.r\_quit\_button\_txt,

action=CB\_QUIT,

)

buttons = [Player1\_btn, Players2\_btn, about\_btn, quit\_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)

border = Border()

border.rectangle(self.screen)

game\_name.draw()

pygame.display.flip()

***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***

import pygame

import random

import r

from r.playernames import \*

from r.game import \*

from sprites.Border import \*

from sprites.Label import \*

from sprites.Button import \*

from sprites.Textbox import \*

CB\_RETURN = 101

CB\_PLAY = 104

CB\_PLAY\_AI=105

BLUE=0

PINK=1

GREEN=2

YELLOW=3

RED=4

P1=0

P2=1

COLOR\_BTN\_ACTIONS=[BLUE\*10+P1,PINK\*10+P1,GREEN\*10+P1,YELLOW\*10+P1,RED\*10+P1,BLUE\*10+P2,PINK\*10+P2,GREEN\*10+P2,YELLOW\*10+P2,RED\*10+P2]

COLOR\_LIST=[r.colors.BLUE,r.colors.PINK,r.colors.GREEN,r.colors.YELLOW,r.colors.RED]

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

class PlayerNamesScreen():

def \_\_init\_\_(self, screen, playernames, player1\_1, player2\_1, player1\_2, player2\_2, name, screen\_dimen, bg\_color, fg\_color, fontsize1 = r.font\_size.l, fontsize2 = r.font\_size.m, fontsize3 = r.font\_size.xs, bg=None):

self.screen = screen

self.screen\_dimen = screen\_dimen

self.bg\_color = bg\_color

self.fg\_color = fg\_color

self.playernames = playernames

self.player1\_1 = player1\_1

self.player2\_1 = player2\_1

self.player1\_2 = player1\_2

self.player2\_2 = player2\_2

self.name = name

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

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

self.font3 = pygame.font.Font("r\\font\_styles\Courier.ttf", fontsize3)

self.playersReset()

self.bgimg = bg

def Player1\_Name(self):

self.playersReset()

ai\_color = random.randint(5,9)

Player\_Names = Label(self.screen, pygame.Rect(165, 10, 1000 ,1000), self.fg\_color, self.bg\_color, self.font1, text=self.playernames)

Player = Label(self.screen, pygame.Rect(120, 100, 1000 ,1000), self.fg\_color, self.bg\_color, self.font2, text=self.player1\_1)

Name1 = Label(self.screen, pygame.Rect(15, 187, 1000 ,1000), self.fg\_color, self.bg\_color, self.font3, text=self.name)

Computer = Label(self.screen, pygame.Rect(535, 100, 1000 ,1000), self.fg\_color, self.bg\_color, self.font2, text=self.player2\_1)

Name2 = Label(self.screen, pygame.Rect(465, 187, 1000 ,1000), self.fg\_color, self.bg\_color, self.font3, text=self.name)

enter\_btn = Button(

center\_position = (SCREEN\_WIDTH/2, 550),

font\_size = r.font\_size.s,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = enter\_button\_txt,

action = CB\_PLAY\_AI,

)

return\_to\_mainmenu\_btn = Button(

center\_position = (r.game.SCREEN\_WIDTH/2, 620),

font\_size = r.font\_size.s,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = return\_to\_mainmenu\_button\_txt,

action=CB\_RETURN,

)

self.setColorButtons()

buttons = [self.Blue1, self.Green1, self.Yellow1, self.Pink1, self.Red1, self.Blue2, self.Green2, self.Yellow2, self.Pink2, self.Red2, enter\_btn, return\_to\_mainmenu\_btn]

self.Blue2.setHighlightable(False)

self.Green2.setHighlightable(False)

self.Yellow2.setHighlightable(False)

self.Pink2.setHighlightable(False)

self.Red2.setHighlightable(False)

Human = Textbox(180, 190, 200, 30)

AI = Textbox(SCREEN\_WIDTH/2+180, 190, 200, 30, text="AI")

while True:

mouse\_up = False

events = pygame.event.get()

for event in events:

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

mouse\_up = True

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

self.handleColorClick(COLOR\_BTN\_ACTIONS[ai\_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:

if button\_action==CB\_PLAY\_AI:

self.p1name=Human.getText()

self.p2name=AI.getText()

for space in range(0,14):

if self.p1name == space \* " ":

self.p1name = "Player1"

if self.p2name == space \* " ":

self.p2name = "Player2"

if button\_action in COLOR\_BTN\_ACTIONS:

self.handleColorClick(button\_action)

else:

return button\_action

pygame.draw.line(self.screen,r.colors.WHITE,[r.game.SCREEN\_WIDTH/2,95],[r.game.SCREEN\_WIDTH/2,505],5)

button.draw(self.screen)

Human.handle\_event(events)

Human.draw(self.screen)

AI.ai\_draw(self.screen)

border = Border()

border.rectangle(self.screen)

Player\_Names.draw()

Player.draw()

Name1.draw()

Computer.draw()

Name2.draw()

pygame.display.flip()

def Players2\_Names(self):

self.playersReset()

Player\_Names = Label(self.screen, pygame.Rect(165, 10, 1000 ,1000), self.fg\_color, self.bg\_color, self.font1, text=self.playernames)

Player1 = Label(self.screen, pygame.Rect(80, 100, 1000 ,1000), self.fg\_color, self.bg\_color, self.font2, text=self.player1\_2)

Name1 = Label(self.screen, pygame.Rect(15, 187, 1000 ,1000), self.fg\_color, self.bg\_color, self.font3, text=self.name)

Player2 = Label(self.screen, pygame.Rect(535, 100, 1000 ,1000), self.fg\_color, self.bg\_color, self.font2, text=self.player2\_2)

Name2 = Label(self.screen, pygame.Rect(465, 187, 1000 ,1000), self.fg\_color, self.bg\_color, self.font3, text=self.name)

enter\_btn = Button(

center\_position = (SCREEN\_WIDTH/2, 550),

font\_size = r.font\_size.s,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = enter\_button\_txt,

action = CB\_PLAY,

)

return\_to\_mainmenu\_btn = Button(

center\_position = (r.game.SCREEN\_WIDTH/2, 620),

font\_size = r.font\_size.s,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.WHITE,

text = return\_to\_mainmenu\_button\_txt,

action=CB\_RETURN,

)

self.setColorButtons()

buttons = [self.Blue1, self.Green1, self.Yellow1, self.Pink1, self.Red1, self.Blue2, self.Green2, self.Yellow2, self.Pink2, self.Red2, enter\_btn, return\_to\_mainmenu\_btn]

P1 = Textbox(180, 190, 200, 30)

P2 = Textbox(SCREEN\_WIDTH/2+180, 190, 200, 30)

textboxes = [P1, P2]

while True:

mouse\_up = False

events = pygame.event.get()

for event in events:

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 textbox in textboxes:

textbox.handle\_event(events)

textbox.draw(self.screen)

for button in buttons:

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

if button\_action is not None:

if button\_action==CB\_PLAY:

self.p1name=P1.getText()

self.p2name=P2.getText()

for space in range(0,14):

if self.p1name == space \* " ":

self.p1name = "Player1"

if self.p2name == space \* " ":

self.p2name = "Player2"

if button\_action in COLOR\_BTN\_ACTIONS:

self.handleColorClick(button\_action)

else:

return button\_action

pygame.draw.line(self.screen,r.colors.WHITE,[r.game.SCREEN\_WIDTH/2,95],[r.game.SCREEN\_WIDTH/2,505],5)

button.draw(self.screen)

border = Border()

border.rectangle(self.screen)

Player\_Names.draw()

Player1.draw()

Name1.draw()

Player2.draw()

Name2.draw()

pygame.display.flip()

def handleColorClick(self, clicked):

i=P1

for p in self.colorBtnList:

if p[clicked//10].staysHighlighted() and i!=clicked%10:

return

i+=1

for btn in self.colorBtnList[clicked%10]:

btn.stayHighlighted(False)

self.colorBtnList[clicked%10][clicked//10].stayHighlighted(True)

if clicked%10==P1:

self.color1=COLOR\_LIST[clicked//10]

elif clicked%10==P2:

self.color2=COLOR\_LIST[clicked//10]

def getPlayer1Name(self):

return self.p1name

def getColor1(self):

return self.color1

def getPlayer2Name(self):

return self.p2name

def getColor2(self):

return self.color2

def setColorButtons(self):

self.Blue1 = Button(

center\_position = (SCREEN\_WIDTH/4, 270),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.BLUE,

text = color\_blue\_label\_txt,

action = BLUE\*10+P1,

)

self.Pink1 = Button(

center\_position = (SCREEN\_WIDTH/4, 320),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.PINK,

text = color\_pink\_label\_txt,

action = PINK\*10+P1,

)

self.Green1 = Button(

center\_position = (SCREEN\_WIDTH/4, 370),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.GREEN,

text = color\_green\_label\_txt,

action = GREEN\*10+P1,

)

self.Yellow1 = Button(

center\_position = (SCREEN\_WIDTH/4, 420),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.YELLOW,

text = color\_yellow\_label\_txt,

action = YELLOW\*10+P1,

)

self.Red1 = Button(

center\_position = (SCREEN\_WIDTH/4, 470),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.RED,

text = color\_red\_label\_txt,

action = RED\*10+P1,

)

self.Blue2 = Button(

center\_position = (3\*(SCREEN\_WIDTH/4), 270),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.BLUE,

text = color\_blue\_label\_txt,

action = BLUE\*10+P2,

)

self.Pink2 = Button(

center\_position = (3\*(SCREEN\_WIDTH/4), 320),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.PINK,

text = color\_pink\_label\_txt,

action = PINK\*10+P2,

)

self.Green2 = Button(

center\_position = (3\*(SCREEN\_WIDTH/4), 370),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.GREEN,

text = color\_green\_label\_txt,

action = GREEN\*10+P2,

)

self.Yellow2 = Button(

center\_position = (3\*(SCREEN\_WIDTH/4), 420),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.YELLOW,

text = color\_yellow\_label\_txt,

action = YELLOW\*10+P2,

)

self.Red2 = Button(

center\_position = (3\*(SCREEN\_WIDTH/4), 470),

font\_size = r.font\_size.xs,

bg\_rgb = r.colors.BLACK,

text\_rgb = r.colors.RED,

text = color\_red\_label\_txt,

action = RED\*10+P2,

)

self.colorBtnList=[[self.Blue1,self.Pink1,self.Green1,self.Yellow1,self.Red1], [self.Blue2,self.Pink2,self.Green2,self.Yellow2,self.Red2]]

def playersReset(self):

self.p1name="Player1"

self.p2name="Player2"

self.color1=\_color\_default

self.color2=\_color\_default

***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

def getYSpeed(self):

return -math.sin(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





























***Limitations***

The limitations of the projects are :

1. No option to change the Theme of the project
2. No option to turn off the sounds
3. The project screen cannot be resized
4. No leader-board maintained
5. Joysticks not supported
6. No settings screen present
7. No other game modes like Infinite Play, Timed Play, etc.

***Bibliography***

1. <https://www.pygame.org/docs/>
2. <https://docs.python.org/3/>
3. <https://www.geeksforgeeks.org/>
4. <https://pythonprogramming.net/pygame-python-3-part-1-intro/>
5. NCERT Class-XI Computer Science Textbook
6. NCERT Class-XII Computer Science Textbook