**Amity International School**

Noida



**Computer Science Project**

**‘ATRA a game’**

## Name : Parth Bhargava

## Class: XII-D

## Roll no. :

INDEX

1. Certificate
2. Acknowledgment
3. Introduction
4. Modules used
5. Source Code
6. Project Output
7. User Manual
8. Bibliography

CERTIFICATE

This is to certify that Parth Bhargava of class XII-D, Amity International School, Nodia, roll number- \_\_\_\_\_\_ has successfully completed this project in computer practical for the AISSCE as prescribed by CBSE in the academic year 2023-24.

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

External Internal

Examiner Examiner

INTRODUCTION

According to our project we were required to write a python script that would help the world while using File Handling. We have created such a thing that would make many people feel like artists . Initially inspired by LED Art, we had made it so you can’t see the drawing as you make it(how? check in our project output page). The more important question is why? . It is said without saying that nowadays everyone is stressed about something or another . we have lost that feeling of trying something and unexpected results are shown. When we draw without the expectation of its becoming or without knowing what we doing we free ourself of the fear of failure and have fun.

This game in its current form does not fully live up to these expectations and it still shows a glimpse of what it can become. Currently the program , game only gives us the option to draw or view our artwork. I am planning to add other features that support this core gameplay of drawing e.i. more different color spectrum where color of pointer changes (there is already one color spectrum which is turn red to blue as show in our project output),the ability to play online with friends and finally other minigames where u have to draw a object from a word or u compare each others artwork so see who is the best artist.

MODULES USED

* Pygame: This is a foreign function library for python. It provides a GUI

,way to observer key changes and mouse movement ,internal clock ,system of color like black =(0,0,0),a way to add circle , rectangle and text objection. There are more functionality of the module but these are main functionality used in the project.

* Sys: This is a another external module Used. It provide way to close a python file efficiently and safely. This used for the sole reason to close a file.
* Pickle: This module is mainly used to manage binary files. In this project  
  It is use to store and extract from binary files.
* time: This module has various uses. Here it use to examine which code performs better. As it was experimentally found that some piece of code perform very poorly in some computers.

SOURCE CODE

The ‘C.s\_Project\_parth’ folder is composed of 6 python files and 8 other files:

* System.py –structures the python files in a program loop
* Parth\_projectCs.py- creates the main GUI of the program
* Image\_recorder.py- records the mouse movent(brush)
* Imgae\_drawer.py- displays the mouse movent(brush)
* artwork\_display.py-display the stored artworks by user
* artwork\_storer.py- saves data file of save1.bat to user entry name of data files
* other files

4 text files mainly used to transfer data to from python files

14 binary files

1. one of them is save1.bat which transfers data to image\_drawer.py from image\_recorder.py . it also moves data to artwork\_storer.py .
2. another one of them is fileBat.bat which saves data as the program is running. There a text file named fileText.txt which  
   which does the same job. which file to use is determine by which way is faster.
3. other binary files are saved data files which produce by

artwork\_storer.py and can be read by artwork\_display.py .

source code of files:

system.py

'''

system

'''

import pygame

import pickle

import sys

import random

import time

#colors

white = (255, 255, 255)

pink = (254, 131, 199)

shodow\_blue = ( 69, 87, 217)

light\_blue = (147, 252, 252)

light\_pink = (202, 10, 244)

black = ( 0, 0, 0)

red = (255, 0, 0)

gold = (248, 172, 0)

with open('game\_size.txt','r') as f:

game\_size = f.read().split(',')

x = int(game\_size[0])

y = int(game\_size[1])

pygame.init()

full\_screen=(x,y)

screen=pygame.display.set\_mode(full\_screen)

#conditions

main\_program\_condition=True

pressed\_draw\_button\_condition=False

pressed\_artwork\_button\_condition=False

pressed\_save\_button\_condition=False

sysquit=False

def neaar\_button\_red(lst):

for info\_about\_a\_button in lst:

(mx,my) = pygame.mouse.get\_pos()

location\_of\_button=1

color\_of\_button =0

x\_axis=0

y\_axis=1

location\_button=info\_about\_a\_button[location\_of\_button]

x\_limit=location\_button[x\_axis] < mx+25 and location\_button[x\_axis] > mx-25

y\_limit=location\_button[y\_axis] < my+25 and location\_button[y\_axis] > my-25

if x\_limit and y\_limit :

info\_about\_a\_button[color\_of\_button]=red

else:

info\_about\_a\_button[color\_of\_button]=black

def content\_data(file):

with open(file,'rb') as f:

L=pickle.load(f)

return L

def name\_OF\_selectedfiles():

with open('game\_file\_location.txt','r') as f:

name\_of\_all\_drawing\_N=f.readlines()

name\_of\_all\_drawing=[]

for i in name\_of\_all\_drawing\_N:

drawing=i.rstrip('\n')

name\_of\_all\_drawing.append(drawing)

return name\_of\_all\_drawing

def animation\_of(file\_name,position) :

name\_of\_all\_drawing = name\_OF\_selectedfiles()

if file\_name == 'random':

file\_number=random.randint(0,len(name\_of\_all\_drawing)-1)

else:

file\_number=name\_of\_all\_drawing.index(file\_name+".bat")

sum\_sub='+'

program=True

interval=0

color\_sep=0

center\_point=(350,350)

new\_center\_point = position

x\_addition=new\_center\_point[0]-center\_point[0]

y\_addition=new\_center\_point[1]-center\_point[1]

data\_file\_of\_recent\_painting = content\_data(name\_of\_all\_drawing[file\_number])

data\_file\_of\_recent\_painting.pop(0)

final\_stop=len(data\_file\_of\_recent\_painting)-1

screen\_animation\_data=[]

while program:

circle\_cordinateX,Y = data\_file\_of\_recent\_painting[interval][0]

circle\_cordinateX += x\_addition

Y += y\_addition

circle\_cordinate = circle\_cordinateX,Y

size\_of\_circle = data\_file\_of\_recent\_painting[interval][1]

color=(255-color\_sep,0,color\_sep)

screen\_animation\_data.append((color,circle\_cordinate,size\_of\_circle))

if color\_sep == 255:

sum\_sub='-'

elif color\_sep == 0:

sum\_sub='+'

if sum\_sub == '-':

color\_sep-=1

else:

color\_sep+=1

print(color\_sep)

interval+=1

if interval == final\_stop - 1 :

return screen\_animation\_data

def readpyfile(file):

with open(file,'r') as f:

l=f.readlines()

start=0

end=len(l)

for i in range(len(l)):

if l[i]=='#start\n':

print('how')

start=i

elif l[i]=='#end\n':

end=i

print(start,end)

f.seek(0)

lines\_of\_code=f.readlines()[start:end]

string\_form='\n'.join(lines\_of\_code)

print(string\_form)

return string\_form

while main\_program\_condition:

#open main menu

exec(readpyfile('parth\_projectCs.py'))

if pressed\_draw\_button\_condition:

exec(readpyfile('image\_recorder.py'))

exec(readpyfile('image\_drawer.py'))

if pressed\_save\_button\_condition:

exec(readpyfile('artwork\_storer.py'))

if pressed\_artwork\_button\_condition:

exec(readpyfile('artwork\_display.py'))

pressed\_draw\_button\_condition,pressed\_artwork\_button\_condition=False,False

pressed\_save\_button\_condition=False

pygame.quit()

sys.exit()

parth\_projectCs.py

'''

GUI

'''

import pygame

import random

import pickle

#color

shodow\_blue= (69, 87, 217)

light\_pink= (202, 10, 244)

black=(0,0,0)

red=(255,0,0)

with open('game\_size.txt','r') as f:

game\_size = f.read().split(',')

x = int(game\_size[0])

y = int(game\_size[1])

pygame.init()

full\_screen=(x,y)

screen=pygame.display.set\_mode(full\_screen)

#conditions

pressed\_draw\_button\_condition,pressed\_artwork\_button\_condition=False,False

def neaar\_button\_red(lst):

for info\_about\_a\_button in lst:

(mx,my) = pygame.mouse.get\_pos()

location\_of\_button=1

color\_of\_button =0

x\_axis=0

y\_axis=1

location\_button=info\_about\_a\_button[location\_of\_button]

x\_limit=location\_button[x\_axis] < mx+25 and location\_button[x\_axis] > mx-25

y\_limit=location\_button[y\_axis] < my+25 and location\_button[y\_axis] > my-25

if x\_limit and y\_limit :

info\_about\_a\_button[color\_of\_button]=red

else:

info\_about\_a\_button[color\_of\_button]=black

def content\_data(file):

with open(file,'rb') as f:

L=pickle.load(f)

return L

def name\_OF\_selectedfiles():

with open('game\_file\_location.txt','r') as f:

name\_of\_all\_drawing\_N=f.readlines()

name\_of\_all\_drawing=[]

for i in name\_of\_all\_drawing\_N:

drawing=i.rstrip('\n')

name\_of\_all\_drawing.append(drawing)

return name\_of\_all\_drawing

def animation\_of(file\_name,position) :

name\_of\_all\_drawing = name\_OF\_selectedfiles()

if file\_name == 'random':

file\_number=random.randint(0,len(name\_of\_all\_drawing)-1)

else:

file\_number=name\_of\_all\_drawing.index(file\_name+".bat")

sum\_sub='+'

program=True

interval=0

color\_sep=0

center\_point=(350,350)

new\_center\_point = position

x\_addition=new\_center\_point[0]-center\_point[0]

y\_addition=new\_center\_point[1]-center\_point[1]

data\_file\_of\_recent\_painting = content\_data(name\_of\_all\_drawing[file\_number])

data\_file\_of\_recent\_painting.pop(0)

final\_stop=len(data\_file\_of\_recent\_painting)-1

screen\_animation\_data=[]

while program:

circle\_cordinateX,Y = data\_file\_of\_recent\_painting[interval][0]

circle\_cordinateX += x\_addition

Y += y\_addition

circle\_cordinate = circle\_cordinateX,Y

size\_of\_circle = data\_file\_of\_recent\_painting[interval][1]

color=(255-color\_sep,0,color\_sep)

screen\_animation\_data.append((color,circle\_cordinate,size\_of\_circle))

if color\_sep == 255:

sum\_sub='-'

elif color\_sep == 0:

sum\_sub='+'

if sum\_sub == '-':

color\_sep-=1

else:

color\_sep+=1

print(color\_sep)

interval+=1

if interval == final\_stop - 1 :

return screen\_animation\_data

#start

clock = pygame.time.Clock()

starting\_time = pygame.time.get\_ticks()

current\_slide=True

head\_font = pygame.font.Font('freesansbold.ttf', 128 )

options\_font = pygame.font.Font('freesansbold.ttf', 48 )

head\_text = head\_font .render('ATRA' , True, light\_pink )

option1 = options\_font.render('Draw' , True, black )

option2 = options\_font.render('Arworks', True, black )

option3 = options\_font.render('QUIT' , True, black )

textRect0 = head\_text.get\_rect()

textRect1 = option1 .get\_rect()

textRect2 = option2 .get\_rect()

textRect3 = option3 .get\_rect()

CENTER\_POSITION=x//2

textRect0.center = (CENTER\_POSITION, 100 )

textRect1.center = (CENTER\_POSITION, 300 )

textRect2.center = (CENTER\_POSITION, 400 )

textRect3.center = (CENTER\_POSITION, 500 )

color\_op1,color\_op2,color\_op3=black,black,black

color\_toggle=[

[color\_op1,(CENTER\_POSITION, 300 )],

[color\_op2,(CENTER\_POSITION, 400 )],

[color\_op3,(CENTER\_POSITION, 500 )],

]

animation\_Main\_GUI\_1 = animation\_of('Tree',(x//2,y//2+50))

animation\_Main\_GUI\_2 = animation\_of('snowball1',(0,y//2+50))

animation\_Main\_GUI\_3 = animation\_of('snowball2',(x,y//2+50))

interval = 0

time\_span = 25

final\_stop\_1 = len(animation\_Main\_GUI\_1) - 1

final\_stop\_2 = len(animation\_Main\_GUI\_2) - 1

final\_stop\_3 = len(animation\_Main\_GUI\_3) - 1

starting\_time = pygame.time.get\_ticks()

screen.fill(shodow\_blue)

while current\_slide:

current\_time = pygame.time.get\_ticks()

for event in pygame.event.get():

if event.type == pygame.QUIT:

current\_slide=False

elif event.type == pygame.KEYDOWN:

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

current\_slide=False

main\_program\_condition=False

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

if color\_toggle[0][0] == red:

pressed\_draw\_button\_condition=True

current\_slide=False

elif color\_toggle[1][0] == red:

pressed\_artwork\_button\_condition=True

current\_slide=False

elif color\_toggle[2][0] == red:

main\_program\_condition=False

current\_slide=False

neaar\_button\_red(color\_toggle)

option1 = options\_font.render('Draw' , True, color\_toggle[0][0])

option2 = options\_font.render('Arworks', True, color\_toggle[1][0])

option3 = options\_font.render('QUIT' , True, color\_toggle[2][0])

if current\_time-starting\_time >= time\_span :

if interval < final\_stop\_1:

screen, color, circle\_cordinate, size\_of\_circle = (screen,) + animation\_Main\_GUI\_1[interval]

pygame.draw.circle(screen, color, circle\_cordinate, size\_of\_circle)

if interval < final\_stop\_2:

screen, color, circle\_cordinate, size\_of\_circle = (screen,) + animation\_Main\_GUI\_2[interval]

pygame.draw.circle(screen, color, circle\_cordinate, size\_of\_circle)

if interval < final\_stop\_3:

screen, color, circle\_cordinate, size\_of\_circle = (screen,) + animation\_Main\_GUI\_3[interval]

pygame.draw.circle(screen, color, circle\_cordinate, size\_of\_circle)

interval+=1

starting\_time=current\_time

screen.blit(head\_text, textRect0 )

screen.blit(option1 , textRect1 )

screen.blit(option2 , textRect2 )

screen.blit(option3 , textRect3 )

pygame.display.flip()

#end

pygame.quit()

image\_recorder.py

import pygame

import pickle

import sys

import time

#color

light\_blue = (147, 252, 252)

light\_pink = (202, 10, 244)

black = ( 0, 0, 0)

white = (255, 255, 255)

with open('game\_size.txt','r') as f:

game\_size = f.read().split(',')

x = int(game\_size[0])

y = int(game\_size[1])

pygame.init()

full\_screen=(x,y)

screen=pygame.display.set\_mode(full\_screen)

#start

def add\_in\_file(item,mode):

if mode == 'binary':

with open('fileBat.bat','ab+') as f:

pickle.dump(item,f)

elif mode == 'text':

with open('fileText.txt','a') as f:

f.write('{}\n'.format(item))

def test\_for\_speed\_best\_mode(item):

t0 = time.time()

for i in range(100):

add\_in\_file(item,'binary')

give\_arry('binary')

t1 = time.time()

time\_binary=t1-t0

t0 = time.time()

for i in range(100):

add\_in\_file(item,'text')

give\_arry('text')

t1 = time.time()

time\_text=t1-t0

if time\_binary > time\_text:

return 'binary'

return 'text'

def give\_arry(mode):

"""

the code below

manipulates the data recived by the file.bat such that all mouse position are unique

and then store it temporary in save1.dat

"""

if mode == 'binary':

l=[(700,700)] #the l[0] acts as a canvas size

with open('fileBat.bat','rb+') as f:

try:

while 1:

mos\_pos=pickle.load(f)

change\_in\_mouse\_pos= mos\_pos == l[-1]

if change\_in\_mouse\_pos:

continue

l.append(mos\_pos)

except Exception:

pass

elif mode == 'text':

l=[(700,700)] #the l[0] acts as a canvas size

with open('fileText.txt','r+') as f:

L=f.read().split('\n')

interval=0

try:

while 1:

mos\_pos=eval(L[interval])

print(interval)

change\_in\_mouse\_pos= mos\_pos == l[-1]

interval+=1

if change\_in\_mouse\_pos:

continue

l.append(mos\_pos)

except Exception:

pass

return l

def color\_change(mouse\_postion,color\_sep,sum\_sub):

print(mouse\_postion,color\_sep,sum\_sub)

if len(mouse\_postion) == 1:

pass

elif mouse\_postion[-2]!=mouse\_postion[-1] :

print(1)

exec('color\_sep{}=1'.format(sum\_sub))

print(color\_sep)

return color\_sep

else:

mouse\_postion.clear()

print(color\_sep)

return color\_sep

def color\_rotion(color\_sep):

if color\_sep == 255:

sum\_sub='-'

elif color\_sep == 0:

sum\_sub='+'

return sum\_sub

with open('fileBat.bat','wb') as f:

pass

with open('fileText.txt','w') as f:

pass

mode = test\_for\_speed\_best\_mode(((0,0),0))

time.sleep(5)

with open('fileBat.bat','wb') as f:

pass

with open('fileText.txt','w') as f:

pass

color\_sep=0

clock = pygame.time.Clock()

starting\_time = pygame.time.get\_ticks()

background\_colour = light\_blue

white\_canvas=(600,600)

size\_of\_circle=5

program=True

toggle\_command={

'pause':False

}

rec\_size = pygame.Rect(100,100,600,600)

head\_font = pygame.font.Font('freesansbold.ttf', 48)

pause\_sign = head\_font.render('Pause', True, (0,0,0))

rec\_pause\_sign = pause\_sign.get\_rect()

rec\_pause\_sign.center = (x-200,100)

color\_display\_cords = (x-400,y-500)

mouse\_postion=[]

while program:

current\_time = pygame.time.get\_ticks()

color\_of\_display=(255-color\_sep,0,color\_sep)

screen.fill(background\_colour)

pygame.draw.rect(screen, white, rec\_size, )

pygame.draw.rect(screen, black, rec\_size, 2 )

for event in pygame.event.get():

if event.type == pygame.QUIT:

program=False

elif event.type == pygame.KEYDOWN:

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

program=False

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

if toggle\_command['pause'] == False:

toggle\_command['pause'] = True

else:

toggle\_command['pause'] = False

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

size\_of\_circle+=10

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

if size\_of\_circle <=10:

continue

size\_of\_circle-=10

sum\_sub=color\_rotion(color\_sep) #each time it hits 255 or 0 the color\_sep

#decreases negitively or increases positively

if toggle\_command['pause']:

# code when program is pause

screen.blit(pause\_sign, rec\_pause\_sign)

continue

mx,my = pygame.mouse.get\_pos()

mouse\_not\_in\_canvas= mx > 700 or mx < 100 or my > 700 or my < 100

if mouse\_not\_in\_canvas:

continue

add\_in\_file(((mx,my),size\_of\_circle),mode)

mouse\_postion.append((mx,my))

color\_sep=color\_change(mouse\_postion,color\_sep,sum\_sub) # the function should give +1 increase color\_step

# if mouse\_pos is changed

circle\_cordinate=(mx,my)

pygame.draw.circle(screen, color\_of\_display, color\_display\_cords, 50, )

pygame.draw.circle(screen, color\_of\_display, circle\_cordinate , size\_of\_circle, )

pygame.draw.circle(screen, black , circle\_cordinate , size\_of\_circle,6)

pygame.display.flip()

l=give\_arry(mode)

with open('save1.bat','wb') as f:

pickle.dump(l,f)

#end

pygame.quit()

image\_drawer.py

import pygame

import pickle

import time

import sys

def neaar\_button\_red(lst):

(mx,my) = pygame.mouse.get\_pos()

location\_of\_button = 1

color\_of\_button = 0

x\_cordinate = 0

y\_cordinate = 1

for info\_about\_a\_button in lst:

location\_button = info\_about\_a\_button [ location\_of\_button ]

x\_limit=location\_button[x\_cordinate] < mx+25 and location\_button[x\_cordinate] > mx-25

y\_limit=location\_button[y\_cordinate] < my+25 and location\_button[y\_cordinate] > my-25

if x\_limit and y\_limit :

info\_about\_a\_button[color\_of\_button]=red

else:

info\_about\_a\_button[color\_of\_button]=black

def content\_data(file):

with open(file,'rb') as f:

L=pickle.load(f)

return L

#color

gold = ( 248, 172, 0)

red = ( 255, 0, 0)

light\_blue= ( 147, 252, 252)

light\_pink= ( 202, 10, 244)

black = ( 0, 0, 0)

white = ( 255, 255, 255)

with open('game\_size.txt','r') as f:

game\_size = f.read().split(',')

x = int(game\_size[0])

y = int(game\_size[1])

pygame.init()

full\_screen=(x,y)

screen=pygame.display.set\_mode(full\_screen)

#start

data\_file\_of\_recent\_painting = content\_data('save1.bat')

data\_file\_of\_recent\_painting.pop(0) # removes the intial size of canvas

pressed\_save\_button\_condition=False

sum\_sub='+'

background\_colour = light\_blue

size\_of\_circle=10

program=True

final\_stop=len(data\_file\_of\_recent\_painting)-1

interval=0

#with time color of drawer changes

color\_sep=0 #color=(255-color\_sep,0,color\_sep)

clock = pygame.time.Clock()

starting\_time = pygame.time.get\_ticks()

toggle\_command={

'speed':'slow'

}

white\_canvas\_cordinates=(100,100)

white\_canvas\_size=(600,600)

x\_axis=0

y\_axis=1

canvas=pygame.Rect(white\_canvas\_cordinates[x\_axis],

white\_canvas\_cordinates[y\_axis],

white\_canvas\_size[x\_axis],

white\_canvas\_size[y\_axis]

)

overline=pygame.Rect(white\_canvas\_cordinates[x\_axis]-50,

white\_canvas\_cordinates[y\_axis]-50,

white\_canvas\_size[x\_axis]+100,

white\_canvas\_size[y\_axis]+100 # size increase by 100 as

# both ends of rectangle inc 50

)

distances\_from\_y\_axis\_start\_correctionprosses=white\_canvas\_cordinates[x\_axis]+white\_canvas\_size[x\_axis]

screen\_size = screen.get\_size()

correction=pygame.Rect(700,

100,

screen\_size[x\_axis],

screen\_size[y\_axis]

)

head\_font = pygame.font.Font('freesansbold.ttf', 48)

canvas\_filled\_sign = head\_font.render('done', True, black)

save\_button = head\_font.render('save', True, black)

speed\_display = head\_font.render('<>', True, black)

rec\_canvas\_filled\_sign = canvas\_filled\_sign.get\_rect()

rec\_save\_button = save\_button.get\_rect()

rec\_speed\_display = speed\_display.get\_rect()

screen\_size = screen.get\_size()

mean\_bw\_distance\_after\_canvas = (screen\_size[x\_axis]-white\_canvas\_size[x\_axis]-100)//2

distance\_before\_canvas=100

com\_dis = white\_canvas\_size[x\_axis]+distance\_before\_canvas

text\_position\_from\_left = mean\_bw\_distance\_after\_canvas+com\_dis

text\_position\_from\_top={'first':100,'second':200,'third':300,'fourth':400}

rec\_canvas\_filled\_sign.center = (text\_position\_from\_left,

text\_position\_from\_top['first']

)

rec\_save\_button.center = (text\_position\_from\_left,

text\_position\_from\_top['second']

)

rec\_speed\_display.center = (text\_position\_from\_left,

text\_position\_from\_top['fourth']

)

repaint\_background = head\_font.render(' ', True, black,background\_colour)

screen.fill(background\_colour)

pygame.draw.rect(screen,(255,255,255),canvas)

buttons = {

'save\_button\_color\_call':[ black,

(text\_position\_from\_left, text\_position\_from\_top['second'])

]

}

while program:

speed\_display = head\_font.render('<{}>'.format(toggle\_command['speed']), True, black)

current\_time = pygame.time.get\_ticks()

for event in pygame.event.get():

if event.type == pygame.QUIT:

program=False

elif event.type == pygame.KEYDOWN:

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

program=False

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

if toggle\_command['speed'] == 'slow':

toggle\_command['speed'] = 'normal'

elif toggle\_command['speed'] == 'normal':

toggle\_command['speed'] = 'fast'

elif toggle\_command['speed'] == 'fast':

toggle\_command['speed'] = 'slow'

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

if buttons['save\_button\_color\_call'][0] == red:

pressed\_save\_button\_condition = True

program=False

if interval == final\_stop:

save\_button = head\_font.render('save', True, buttons['save\_button\_color\_call'][0])

neaar\_button\_red(list(buttons.values()))

screen.blit( save\_button , rec\_save\_button )

screen.blit( canvas\_filled\_sign , rec\_canvas\_filled\_sign )

screen.blit( repaint\_background , rec\_speed\_display )

screen.blit( speed\_display , rec\_speed\_display )

pygame.display.flip()

continue

print(data\_file\_of\_recent\_painting)

print(interval)

print(final\_stop)

circle\_cordinate = data\_file\_of\_recent\_painting[interval][0]

size\_of\_circle = data\_file\_of\_recent\_painting[interval][1]

color=(255-color\_sep,0,color\_sep)

#make it a def function input toggle\_command['speed'] returns timespan

if toggle\_command['speed']=='slow':

time\_span=25

elif toggle\_command['speed']=='normal':

time\_span=15

elif toggle\_command['speed']=='fast':

time\_span=2

if current\_time-starting\_time >= time\_span:

if color\_sep == 255:

sum\_sub='-'

elif color\_sep == 0:

sum\_sub='+'

exec('color\_sep{}=1'.format(sum\_sub))

interval+=1

starting\_time=current\_time

pygame.draw.circle(screen, color, circle\_cordinate, size\_of\_circle)

pygame.draw. rect(screen, background\_colour,correction, )

pygame.draw. rect(screen, gold ,overline , 50)

pygame.draw. rect(screen, black ,canvas , 2)

screen.blit( repaint\_background, rec\_speed\_display)

screen.blit( speed\_display , rec\_speed\_display)

pygame.display.flip()

#end

pygame.quit()

artwork\_storer.py

import pygame

import pickle

import sys

def content\_data(file):

with open(file,'rb') as f:

L=pickle.load(f)

return L

#color

white = (255, 255, 255)

red = (238, 38, 21)

black = ( 0, 0, 0)

pink = (254, 131, 199)

shodow\_blue = ( 69, 87, 217)

with open('game\_size.txt','r') as f:

game\_size = f.read().split(',')

x = int(game\_size[0])

y = int(game\_size[1])

pygame.init()

full\_screen=(x,y)

screen=pygame.display.set\_mode(full\_screen)

#start

def save(data,file\_name):

with open('{}.bat'.format(file\_name),'wb') as f:

pickle.dump(data,f)

with open("file\_location.txt",'a') as f:

f.write('{}.bat\n'.format(file\_name))

background = shodow\_blue

program = True

base\_font = pygame.font.Font(None, 32)

big\_base\_font = pygame.font.Font(None, 48)

user\_text = ''

input\_rect = pygame.Rect(200, 200, 140, 32)

x, y = screen.get\_size()

input\_rect.center=(x//2,y//2)

color = white

name\_to\_long\_error = False

active = False

head\_font = pygame.font.Font('freesansbold.ttf', 200)

save\_title = head\_font.render('save', True, red)

error\_message\_name\_to\_long = big\_base\_font.render('name entered is to long \n maximum name can 10 charecters long', True, red)

rec\_save\_title = save\_title.get\_rect()

rec\_error\_message\_name\_to\_long = error\_message\_name\_to\_long.get\_rect()

rec\_save\_title.center = (x//2,200)

rec\_error\_message\_name\_to\_long.center = (x//2,y//2+30)

screen.fill(background)

while program:

for event in pygame.event.get():

if event.type == pygame.QUIT:

program=False

elif event.type == pygame.KEYDOWN:

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

user\_text = user\_text[:-1]

name\_to\_long\_error = False

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

if len(user\_text) <=10:

program=False

else:

name\_to\_long\_error = True

else:

user\_text += event.unicode

screen.fill(background)

pygame.draw.rect(screen, color, input\_rect)

text\_surface = base\_font.render(user\_text, True, pink)

screen.blit(save\_title, rec\_save\_title)

screen.blit(text\_surface, (input\_rect.x+5, input\_rect.y+5))

input\_rect.w = max(100, text\_surface.get\_width()+10)

if name\_to\_long\_error:

screen.blit(error\_message\_name\_to\_long, rec\_error\_message\_name\_to\_long)

pygame.display.flip()

L=content\_data('save1.bat')

save(L,user\_text)

#end

pygame.quit()

artwork\_display.py

'''

artwork looker

'''

import pygame

import pickle

def neaar\_button\_red(lst):

for info\_about\_a\_button in lst:

(mx,my) = pygame.mouse.get\_pos()

location\_of\_button=1

color\_of\_button =0

x\_axis=0

y\_axis=1

location\_button=info\_about\_a\_button[location\_of\_button]

x\_limit=location\_button[x\_axis] < mx+25 and location\_button[x\_axis] > mx-25

y\_limit=location\_button[y\_axis] < my+25 and location\_button[y\_axis] > my-25

if x\_limit and y\_limit :

info\_about\_a\_button[color\_of\_button]=red

else:

info\_about\_a\_button[color\_of\_button]=black

def content\_data(file):

with open(file,'rb') as f:

L=pickle.load(f)

return L

gold=(248, 172, 0)

red=(255,0,0)

light\_blue= (147, 252, 252)

light\_pink=(202, 10, 244)

black=(0,0,0)

white=(255,255,255)

with open('game\_size.txt','r') as f:

game\_size = f.read().split(',')

x = int(game\_size[0])

y = int(game\_size[1])

pygame.init()

full\_screen=(x,y)

screen=pygame.display.set\_mode(full\_screen)

#start

def name\_OF\_files():

with open('file\_location.txt','r') as f:

name\_of\_all\_drawing\_N=f.readlines()

name\_of\_all\_drawing=[]

for i in name\_of\_all\_drawing\_N:

drawing=i.rstrip('\n')

name\_of\_all\_drawing.append(drawing)

return name\_of\_all\_drawing

name\_of\_all\_drawing = name\_OF\_files()

back=False

front=False

sum\_sub='+'

background\_colour = light\_blue

size\_of\_circle=10

program=True

interval=0

color\_sep=0

clock = pygame.time.Clock()

starting\_time = pygame.time.get\_ticks()

toggle\_command={

'speed':'slow'

}

white\_canvas\_cordinates=(100,100)

white\_canvas\_size=(600,600)

x\_axis=0

y\_axis=1

canvas=pygame.Rect(white\_canvas\_cordinates[x\_axis],

white\_canvas\_cordinates[y\_axis],

white\_canvas\_size[x\_axis],

white\_canvas\_size[y\_axis]

)

overline=pygame.Rect(white\_canvas\_cordinates[x\_axis]-50,

white\_canvas\_cordinates[y\_axis]-50,

white\_canvas\_size[x\_axis]+100,

white\_canvas\_size[y\_axis]+100 # size increase by 100 as

# both ends of rectangle inc 50

)

distances\_from\_y\_axis\_start\_correctionprosses=white\_canvas\_cordinates[x\_axis]+white\_canvas\_size[x\_axis]

screen\_size = screen.get\_size()

correction=pygame.Rect(700,

100,

screen\_size[x\_axis],

screen\_size[y\_axis]

)

head\_font = pygame.font.Font('freesansbold.ttf', 48)

right\_direction = head\_font.render('press n --->', True, black)

left\_direction = head\_font.render('press b <---', True, black)

speed\_display = head\_font.render('<>', True, black)

painting\_name\_display = head\_font.render('<>', True, black)

rec\_right\_direction = right\_direction.get\_rect()

rec\_left\_direction = left\_direction.get\_rect()

rec\_speed\_display = speed\_display.get\_rect()

rec\_painting\_name\_display = painting\_name\_display.get\_rect()

screen\_size = screen.get\_size()

mean\_bw\_distance\_after\_canvas = (screen\_size[x\_axis]-white\_canvas\_size[x\_axis]-100)//2

distance\_before\_canvas=100

com\_dis = white\_canvas\_size[x\_axis]+distance\_before\_canvas

text\_position\_from\_left = mean\_bw\_distance\_after\_canvas+com\_dis

text\_position\_from\_top={'first':100,'second':200,'third':300,'fourth':400}

rec\_right\_direction.center = (text\_position\_from\_left,

text\_position\_from\_top['first']

)

rec\_left\_direction.center = (text\_position\_from\_left,

text\_position\_from\_top['second']

)

rec\_painting\_name\_display.center = (text\_position\_from\_left,

text\_position\_from\_top['third']

)

rec\_speed\_display.center = (text\_position\_from\_left,

text\_position\_from\_top['fourth']

)

screen.fill(background\_colour)

pygame.draw.rect(screen,white,canvas)

file\_number=0

data\_file\_of\_recent\_painting = content\_data(name\_of\_all\_drawing[file\_number])

data\_file\_of\_recent\_painting.pop(0)

final\_stop=len(data\_file\_of\_recent\_painting)-1

last\_indix\_file=len(name\_of\_all\_drawing)-1

while program:

painting\_name\_display = head\_font.render('<{}>'.format(name\_of\_all\_drawing[file\_number].rstrip('.bat')), True, black)

speed\_display = head\_font.render('<{}>'.format(toggle\_command['speed']), True, black)

if back or front:

screen.fill(background\_colour)

pygame.draw.rect(screen,white,canvas)

if back and file\_number>0 :

file\_number-=1

if front and file\_number < last\_indix\_file:

file\_number+=1

interval=0

color\_sep=0

data\_file\_of\_recent\_painting = content\_data(name\_of\_all\_drawing[file\_number])

data\_file\_of\_recent\_painting.pop(0)

final\_stop=len(data\_file\_of\_recent\_painting)-1

back=False

front=False

if toggle\_command['speed']=='slow':

time\_span=25

elif toggle\_command['speed']=='normal':

time\_span=15

elif toggle\_command['speed']=='fast':

time\_span=2

current\_time = pygame.time.get\_ticks()

circle\_cordinate = data\_file\_of\_recent\_painting[interval][0]

size\_of\_circle = data\_file\_of\_recent\_painting[interval][1]

color=(255-color\_sep,0,color\_sep)

for event in pygame.event.get():

if event.type == pygame.QUIT:

program=False

elif event.type == pygame.KEYDOWN:

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

program=False

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

if toggle\_command['speed'] == 'slow':

toggle\_command['speed'] = 'normal'

elif toggle\_command['speed'] == 'normal':

toggle\_command['speed'] = 'fast'

elif toggle\_command['speed'] == 'fast':

toggle\_command['speed'] = 'slow'

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

back=True

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

front=True

pygame.draw.rect(screen,background\_colour,correction)

screen.blit(left\_direction, rec\_left\_direction)

screen.blit(right\_direction, rec\_right\_direction)

screen.blit(painting\_name\_display, rec\_painting\_name\_display)

pygame.draw.circle(screen,color,circle\_cordinate,size\_of\_circle)

pygame.draw.rect(screen,gold,overline,50)

pygame.draw.rect(screen,(0,0,0),canvas,2)

screen.blit(speed\_display, rec\_speed\_display)

pygame.display.flip()

if current\_time-starting\_time >= time\_span and interval < final\_stop - 1 :

if color\_sep == 255:

sum\_sub='-'

elif color\_sep == 0:

sum\_sub='+'

exec('color\_sep{}=1'.format(sum\_sub))

interval+=1

starting\_time=current\_time

#end

pygame.quit()

PROJECT OUTPUT

we were expected to make a program that would help users in some way. My program delivers a fun little game\challenge to draw without exactly knowing what you are drawing. By the source code you will understand the game is divide by 6 python file. While one python file “system.py” gives the overall structure of the program(when will each piece of code will be executed).

here is the main menu.

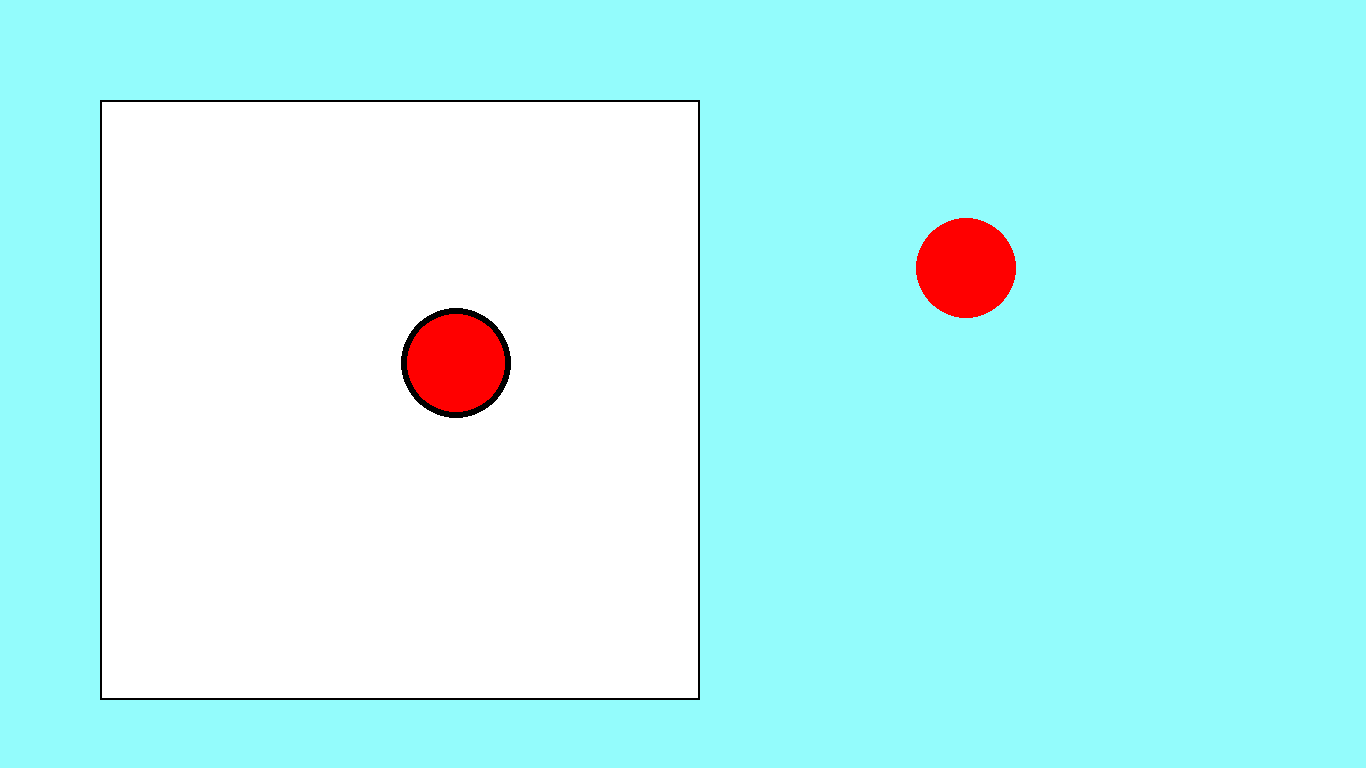


the game menu shows a short animation of tree being made . which shows the users the stuff they can make in the game.

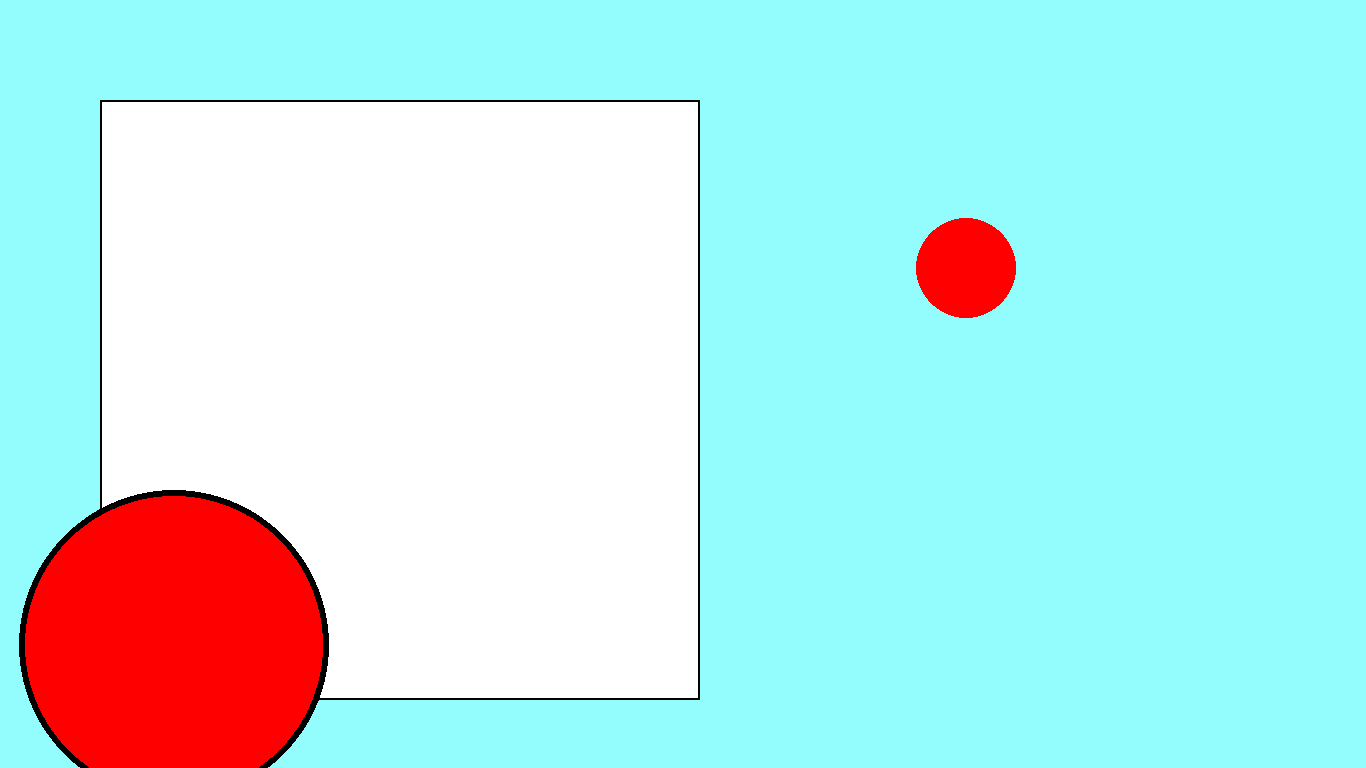
end of animation:



After clicking the game button the player can draw what they wish to draw.



the player may increase or decrease the size of the pointer which impacts the final output of the drawing.



final output:



The final output plays the animation of the thing the player draws from start to finish.

One may save their artwork by clicking the save button.



if the name is to long it gives an too long error



In the end it takes player to the main menu . Here we have named the file ‘noerror’.

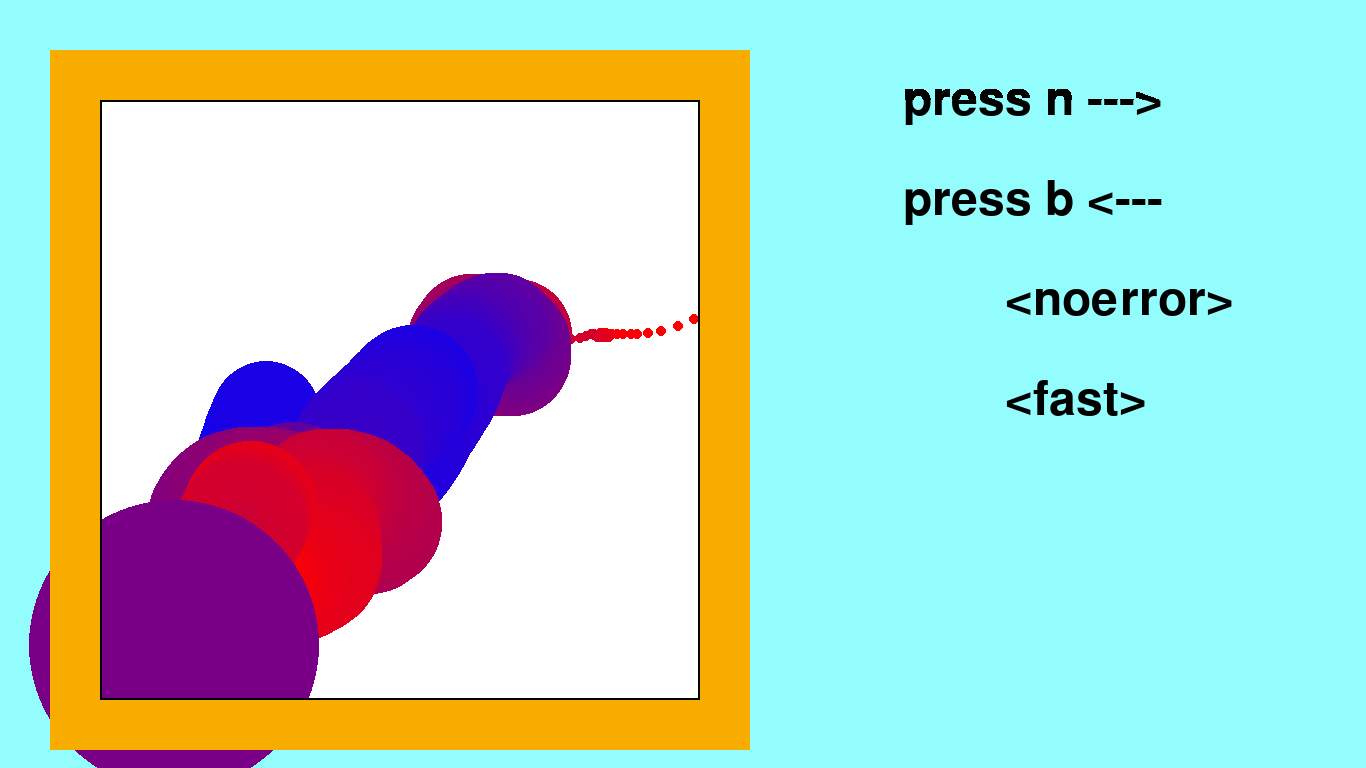


The second button artworks leads to various artworks which were saved by player.





we can navigate by pressing N and B keys.



look above thats our saved artwork ‘noerror’.

Q key leads us to the main menu



then by entering the QUIT button we exit the program

BIBLIOGRAPHY

1. <https://stackoverflow.com/>
2. <https://www.pygame.org/>
3. <https://www.geeksforgeeks.org/>
4. <https://realpython.com/pygame-a-primer/>