INDEX:

1. CERTIFICATE

2. ACKNOWLEDGEMENT

3. INTRODUCTION

4. SYSTEM REQUIREMENTS

5. IMPORTANT DATA STRUCTURES INVOLVED

6. DATA FILES USED IN PROGRAM

7. IMPORTANT MODULES AND THEIR FUNCTIONS

8. IMPORTANT FUNCTIONS

9. INPUT OUTPUT SCREENS

10. SOURCE CODE

11. LIMITATIONS/ FUTURE ENHANCEMENTS

12. BIBLOGRAPHY

**Certificate**

**Certified to be the bona fide work done by**

**RAHUL JOON**

**Of class XII**

**In COMPUTER SCIENCE during the**

**Year 2020-2021**

**Date: / /2020**

**Submitted for the CENTRAL BOARD OF SECONDARY**

**EDUCATION**

**Examination held in the COMPUTER LAB at NIRMAL BHARTIA**

**SCHOOL**

**Examiner:**

**Date:**

**Acknowledgement**

**I am deeply indebted to Mrs Punam ma’am, my Computer Science teacher, for helping me during the development of my project by giving valuable guidance throughout the process and giving me tips to improve my project.**

**I am also indebted to my parent for providing me a better pc so that I could work faster and better on the project and also for their honest inputs throughout the process.**

**I would also like to thank Mrs Charu ma’am and Mrs Amita ma’am, the Principal and head of senior school respectively for providing me with the necessary infrastructure to learn.**

**I would also like to thank the creators of youtube videos on tkinter and pygame for helping me learn them both respectively.**

**Introduction**

ABOUT THE PROJECT

The project is a two player game in which the player controls a tank and destroys the other player in a randomly generated maze. The name of the game is tank busters and is played on a single computer. Once the bullet of either of the tanks hits the other, the game is over and the other player wins.

The final scores of both the players are stored in a data file which can be accessed in the game by going into the settings. The scores are added if the player chooses to play again and only the final score is stored in the file. The banners and the tank itself has been created by us in paint 3d.

The game starts with a video for showing the name of the company which made the game and opens onto the main menu. The menu consists of the play, settings and exit buttons. The settings menu consists of play and music off and displaying of scores options. The start button opens a window to choose color and name of both players and then the game begins with both tanks with player name on top and the color chosen

The source of our idea is that we both are gaming enthusiasts and were interested in developing a game which we used to play with our friends.

THE DEVELOPMENT CYCLE:

**System Requirements**

* Python version 3.8 installed
* Modules required – Tkinter, PIL, Imageio, Pathlib, Pygame, os and math
* 64 bit arm architecture CPU
* Windows 7 and above
* 4 GB ram recommended

**Important Data Structures**

**Involved:**

**Class : Player**

|  |  |
| --- | --- |
| **Variable name** | **Purposes** |
| X | X axis of tank |
| Y | Y axis of tank |
| Width | Width of tank |
| Height | Height of tank |
| Color | Color of tank |
| Tank | Properties of tank |
| Walls | Properties of walls |

**Methods**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method name** | **Inputs** | **Purpose** | **Returns** |
| \_\_init\_\_ | X,y,width,height,color | Definition of tank | None |
| movementcheck() | Tank , walls | Checks collision with walls of tank | ‘n’ |
| Animate | Win | Creates tank | Blits tank image on screen |

**Class : Gun**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method name** | **Inputs** | **Purpose** | **Returns** |
| \_\_init\_\_ | X,y, radius , degree | Defination of gun | none |
| bounce | Bullet, walls | Defination of bullets | ‘n’ |
| animate | Win | Creates bullet | Bullet blit on screen |

|  |  |
| --- | --- |
| **Variable name** | **Purposes** |
| **Variable name** | **Purpose** |
| X | Position in x axis of bullet |
| Y | Position on y axis of bullet |
| Radius | Radius of bullet |
| Degree | Angle at which bullet moves |
| Bullet | Properties of bullet |
| Walls | Properties of wall |
| Win | Game window |

**Class : Obstructions**

|  |  |
| --- | --- |
| **Variable name** | **Purpose** |
| Win | Game window |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method name** | **Inputs** | **Purpose** | **Returns** |
| --init-- | none | Walls defination | none |
| animate | win | Creates walls | Blits walls on the screen |

**Data files used in program**

* **Data -** Stores the data of all the games played and their scores.

**Important Modules And Their**

**Functions Used:**

|  |  |  |
| --- | --- | --- |
| **Module name** | **Library function** | **Purpose** |
| Tkinter | Label() | Creates a widget |
| root | Creates main window |
| Toplevel() | Creates another window |
| Iconify() | Makes the window a icon on the taskbar |
| Button() | A button which when pressed executes a function |
| Grid() | Positions item with respect to other items on window |
| Configure() | Change the configuration of the button or label |
| Destroy() | Removes the widget |
| Entry() | An entry widget to enter text |
| PIL | ImageTk | A module which contains image modification functions |
| Image | Helps in using images in python window |
| Imageio | get\_reader | Enables converting images into video in python |
| pathlib | Path.join() | joins all material locations to code |
| pygame | init | Initializes pygame |
| display | Creates window on which game runs |
| fill | Fills page with specified color |
| Key.get\_pressed | Checks which key has been entered by the user |
| animate | Creates the object onto the display |
| quit | Exits pygame window |
| math | Round() | Mathematical calculations rounding to nearest whole number |
| os | Exit() | Exits the program. |

**Important Functions:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Inputs** | **Purpose** | **Returns** |
| Startup() | none | Video for the beginning | Displays a mp4 video on the screen |
| Menu\_page() | none | Startup menu for the game | Displays screen with start,settings and exit buttons. |
| Start() | none | Function on pressing start button and all other functions are defined inside thereafter. | Display screen with option to choose color of tank |
| Settings() | none | Function on pressing settings button. | Displays screen with option to see score and mute music |
| Exit1() | none | Function on pressing exit button | Exits program |

**Input Output Screens**

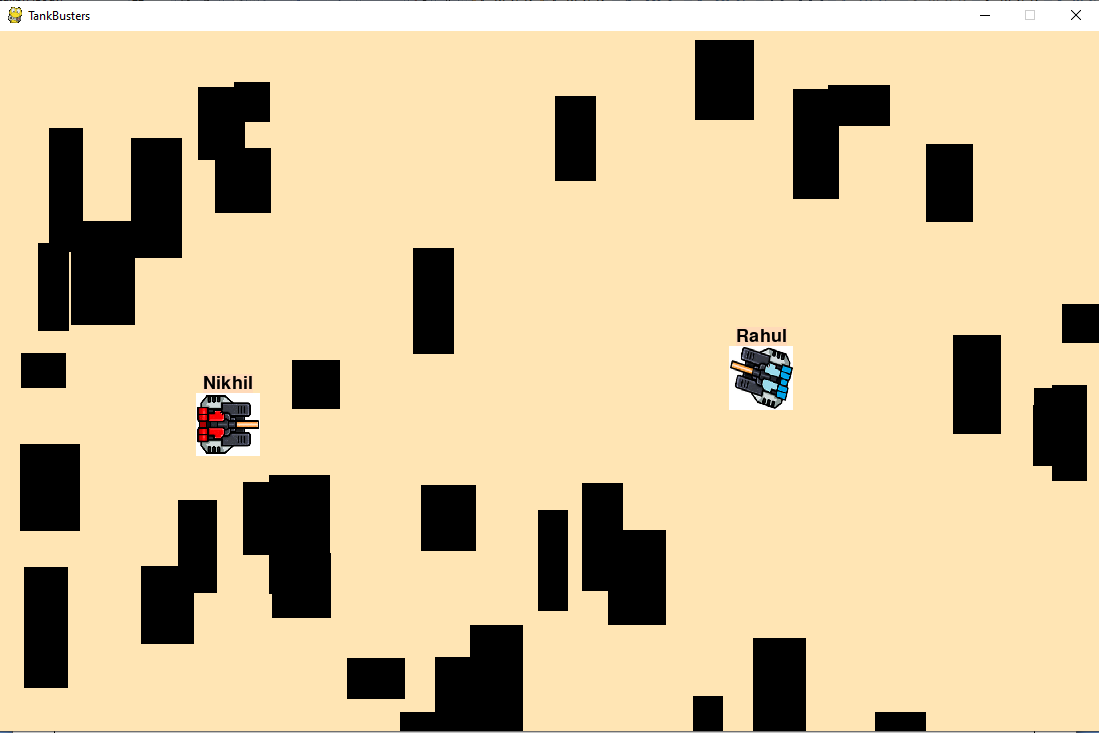
**Main Menu**

****

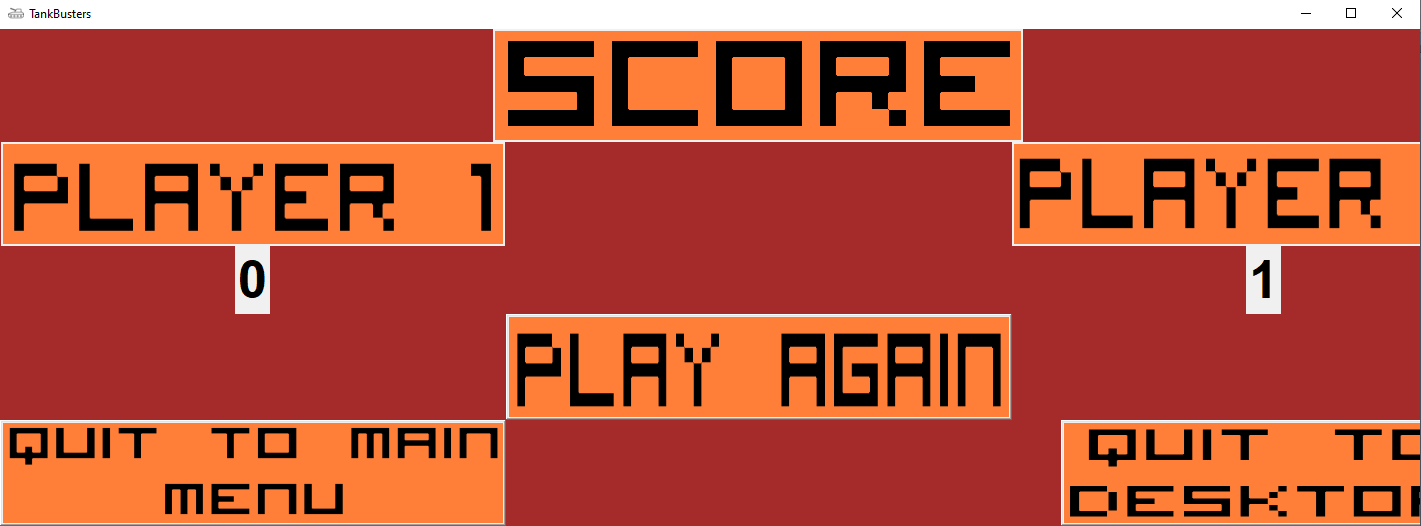
**Settings**

****

**Gameplay**

****

**Final Window**

****

**Source Code**

**Classes –**

import socket #importing all the necessary modules.

import pygame as p

import random as r

import math as m

import os

class player(): #Creating the class for the player itslef.

def \_\_init\_\_(self, x , y, width, height, color): #initializing the class

self.x = x

self.y = y

self.height = height

self.width = width

self.vel = 7

self.deg = 0

self.dim = p.Rect(self.x, self.y, self.width, self.height)

self.color=color

def movementcheck(self, tank, walls): #defining the function for the checking the movement of the player

self.tank = tank

self.walls = walls

for wall in self.walls:

if self.tank.dim.colliderect(wall.dim):

return 'n'

def animate(self, win): #defining a function for animating the player on the window provided

win.blit(p.image.load(os.path.join('Resources/Tanks Animations', f't{self.color}{self.deg}.png')).convert() , (self.x, self.y) )

class gun(): #Creating the class for the bullets objects

def \_\_init\_\_(self, x, y, radius, deg): #initializing the class

self.x = x

self.y = y

self.radius = radius

self.deg = deg

self.vel = 10

def bounce(self, bullet, walls): #defining the function for the bullet to bounce of obstacles.

self.bullet = bullet

self.walls = walls

for wall in walls:

p1 = (wall.dim[0], wall.dim[1])

p2 = (wall.dim[0] + wall.dim[2], wall.dim[1])

p3 = (wall.dim[0] + wall.dim[2], wall.dim[1] + wall.dim[3])

p4 = (wall.dim[0], wall.dim[1] + wall.dim[3])

lines = [(p1, p2, wall.dim[2]), (p3, p4, wall.dim[2]), (p1, p4, wall.dim[3]), (p2, p3, wall.dim[3])]

for line in lines:

AB = [line[1][0] - line[0][0], line[1][1] - line[0][1]]

BE = [self.bullet.x - line[1][0], self.bullet.y - line[1][1]]

AE = [self.bullet.x - line[0][0], self.bullet.y - line[0][1]]

AB\_BE = AB[0] \* BE[0] + AB[1] \* BE[1]

AB\_AE = AB[0] \* AE[0] + AB[1] \* AE[1]

distance = 0

if AB\_BE > 0:

y = self.bullet.y - line[1][1]

x = self.bullet.x - line[1][0]

distance = m.sqrt(x\*x + y\*y)

elif AB\_AE < 0:

y = self.bullet.y - line[0][1]

x = self.bullet.x - line[0][0]

distance = m.sqrt(x\*x + y\*y)

else:

x1 = AB[0]

y1 = AB[1]

x2 = AE[0]

y2 = AE[1]

mod = m.sqrt(x1 \* x1 + y1 \* y1);

distance = abs(x1 \* y2 - y1 \* x2) / mod

if distance <= self.radius:

if line[0][0] == line[1][0]:

self.bullet.deg = - self.bullet.deg

else :

self.bullet.deg = 180 - self.bullet.deg

return 'done'

def animate(self, win): #defining the function for animating the bullets on the window.

p.draw.circle(win, (0 ,0, 0), (self.x, self.y), self.radius)

class obstructions(): #Creating a class for the obstacles to be produced randomly every match.

def \_\_init\_\_(self): #initializing the class

self.x = r.randint(0, 1436)

self.y = r.randint(0, 700)

self.width = r.randint(30, 64)

self.height = r.randint(30, 128)

self.dim = (self.x, self.y, self.width, self.height)

def animate(self, win): #defining the function for animating the obstacles.

p.draw.rect(win, (0, 0, 0), self.dim)

**Main code –**

from tkinter import \*

import tkinter.font as font

from PIL import ImageTk,Image

import imageio

from pathlib import Path

import pygame as p

import math as m

import os

from Classes import player, gun, obstructions

global player1\_score

global player2\_score

# Stores the score of both players

player1\_score=0

player2\_score=0

p.mixer.init()

#loads in sound and plays it

p.mixer.music.load(os.path.join('Resources/Sounds', "battle.mp3"))

p.mixer.music.play(loops=2)

p.mixer.music.set\_volume(0.05)

#creates window for user interface

root=Tk()

root.title("TankBusters")

root.iconbitmap(os.path.join('Resources/Banners', 'tank.ico'))

root.geometry("1280x720")

root.configure(bg="brown")

# The video which plays during the beginning

def startup():

# Creating another window on which the video will play while the main menu loads

top=Toplevel()

top.title("TankBusters")

top.iconbitmap(os.path.join('Resources/Banners', 'tank.ico'))

my\_label=Label(top)

my\_label.grid(row=0,column=0)

video\_name= os.path.join('Resources/Vids Animations', 'starting.mp4')

video=imageio.get\_reader(video\_name)

#Determines the frames per second the video will run on

delay = int(1000 / video.get\_meta\_data()['fps'])

def stream(label):

try:

image=video.get\_next\_data()

except:

video.close()

return

root.iconify()

label.after(delay,lambda:stream(label))

frame\_image=ImageTk.PhotoImage(Image.fromarray(image))

label.config(image=frame\_image)

label.image = frame\_image

my\_label.after(delay,lambda:stream(my\_label))

root.deiconify()

my\_label.after(25000,lambda:top.destroy())

root.deiconify()

#the main menu consisting of start settings and exit

def menu\_page():

#Making all images global so they appear on screen while calling them outside function

global img

global img2

global img3

global img4

# button code for play button

img=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', 'start .png')))

global Play\_button

Play\_button=Button(image=img,command=start,padx=5)

Play\_button.configure(highlightthickness=0)

Play\_button.grid(row=0,column=6)

# code for main menu image

img2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "tanki.png")))

global menulbl

menulbl=Label(image=img2)

menulbl.grid(row=0,column=0,rowspan=5,columnspan=1)

# button code for settings button

img3=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "settings.png")))

global settings\_button

settings\_button=Button(image=img3,command=settings,padx=1)

settings\_button.configure(highlightthickness=0)

settings\_button.grid(row=1,column=6)

# button code for exit button

img4=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "exit.png")))

global exit\_button

exit\_button=Button(image=img4,command=exit1,padx=5)

exit\_button.configure(highlightthickness=0)

exit\_button.grid(row=2,column=6)

# The function of the start button

def start():

# Destroy all images and buttons on the previous page

menulbl.destroy()

Play\_button.destroy()

settings\_button.destroy()

exit\_button.destroy()

# Game to start in this function

def END\_GAME():

# Creating a window for endscreen

top2=Tk()

top2.title("TankBusters")

top2.iconbitmap(os.path.join('Resources/Banners', 'tank.ico'))

global imgend

imgend=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "end.png")))

labelre=Label(top2,image=imgend)

labelre.grid(row=0,column=0)

top2.mainloop()

def MAIN\_GAME(player1\_score,player2\_score): #function for the Main loop of the game

def END\_GAME(player1\_score,player2\_score): # function for displaying the scores

if tank1y==False:

player2\_score+=1

if tank2y==False:

player1\_score+=1

def menu(): #function for producing the menu.

top2.destroy()

root=Tk()

root.title("TankBusters")

root.iconbitmap('tank.ico')

root.geometry("1280x720")

root.configure(bg="brown")

menu\_page()

menu

def quitz(): #function for qutting the program

def data(): #fucntion for storing the data in database file.

writer=open("data.txt","a+")

line=player\_name1+" = "+str(player1\_score)+" "+player\_name2+" = "+str(player2\_score)+"\n"

writer.write(line)

writer.close()

data()

top2.destroy()

def again():

top2.destroy()

MAIN\_GAME(player1\_score,player2\_score)

top2=Tk()

top2.title("TankBusters")

top2.iconbitmap('tank.ico')

top2.configure(bg="brown")

global imgend

imgend=ImageTk.PhotoImage(Image.open("score.png"))

labelre=Label(top2,image=imgend)

labelre.grid(row=0,column=0,rowspan=2,columnspan=6)

myFont = font.Font(family='Helvetica', size=40, weight='bold')

img1score=ImageTk.PhotoImage(Image.open("player 1 score.png"))

score1=Label(image=img1score)

score1.grid(row=3,column=1)

score11=Label(text=player1\_score)

score11.grid(row=4,column=1)

score11['font'] = myFont

img2score=ImageTk.PhotoImage(Image.open("player 2 score.png"))

score2=Label(image=img2score)

score2.grid(row=3,column=3)

score22=Label(text=player2\_score)

score22['font'] = myFont

score22.grid(row=4,column=3)

imgagain=ImageTk.PhotoImage(Image.open("play again.png"))

play\_again=Button(top2,image=imgagain,command=again)

play\_again.grid(row=5,column=2)

imgquitm=ImageTk.PhotoImage(Image.open("quit menu.png"))

Quit\_menu=Button(top2,image=imgquitm,command=menu)

Quit\_menu.grid(row=6,column=1)

imgquitd=ImageTk.PhotoImage(Image.open("quit desktop.png"))

play\_again=Button(top2,image=imgquitd,command=quitz)

play\_again.grid(row=6,column=3)

top2.mainloop()

# Game starts

p.init() #initializing the pygame func

win = p.display.set\_mode((1100, 700)) #setting up the base window

p.display.set\_caption("TankBusters") #setting the caption

win.fill((255,255,255)) #filling the window with color

run = True #game running constant

tank1 = player(200, 200, 64, 64, tank\_color1) #initializing the player 1

tank1y=True #player 1 alive constant

tank2 = player(800, 400, 64, 64, tank\_color2) #initializing the player 2

tank2y=True #player 2 alive constant

bullets1 = [] #list of bullets for player 1

bullets2 = [] #list of bullets for player 2

clock = p.time.Clock() #setting up clock for frames setting

wallt = True #constant for walls(obstacles)

walls = [] #list of obstacles

wall = obstructions() #assigning the obstructions to wall

font2=p.font.Font('freesansbold.ttf', 18)

text1 = font2.render(player\_name1, True, (0,0,0), (255,218,185))

textRect1 = text1.get\_rect()

text2 = font2.render(player\_name2, True, (0,0,0), (255,218,185))

textRect2 = text2.get\_rect()

while run: #running the main while loop

textRect1.center = (tank1.x+32,tank1.y-10)

textRect2.center = (tank2.x+32,tank2.y-10)

p.time.delay(35)

clock.tick(120) #setting up the frame limit

for b1 in bullets1: #function for checking the movement of bullets

x = b1.bounce(b1, walls) #bouncing off the bullets that collide

if b1.y < 700 and b1.y > 0 and b1.x > 0 and b1.x < 1000: #removing bullets that go out window

b1.y -= round(m.cos(((m.pi)/180) \* b1.deg ) \* b1.vel)

b1.x += round(m.sin(((m.pi)/180) \* b1.deg) \* b1.vel)

else :

bullets1.pop(bullets1.index(b1))

if round(((tank2.y - b1.y)\*\*2 + (tank2.x - b1.x)\*\*2)\*\*(1/2)) <= tank2.width:

tank2y=False

for b2 in bullets2: #bullets operation for player 2 bullets.

x = b2.bounce(b2, walls)

if b2.y < 700 and b2.y > 0 and b2.x > 0 and b2.x < 1000:

b2.y -= round(m.cos(((m.pi)/180) \* b2.deg ) \* b2.vel)

b2.x += round(m.sin(((m.pi)/180) \* b2.deg) \* b2.vel)

else :

bullets2.pop(bullets2.index(b2))

if round(((tank1.y - b2.y)\*\*2 + (tank1.x - b2.x)\*\*2)\*\*(1/2)) <= tank1.width:

tank1y=False

if wallt: #creating all the obstacles

for i in range(50):

a = obstructions()

walls.append(a)

if tank1.dim.colliderect(a.dim): #removing walls that are already colliding with tanks

walls.remove(a)

if tank2.dim.colliderect(a.dim):

walls.remove(a)

wallt = False

for event in p.event.get(): #loop to quit pygame when pressed the cross sign

if event.type == p.QUIT:

run = False

keys = p.key.get\_pressed() #list for all the keys that are pressed at the moment

if keys[p.K\_UP]: #if upper key is pressed than tank movement is processed forward.

tempx1 = tank1.x

tempy1 = tank1.y

tank1.y -= round(m.cos(((m.pi)/180) \* tank1.deg ) \* tank1.vel)

tank1.dim[1] = tank1.y

if tank1.movementcheck(tank1, walls) == 'n':

tank1.y = tempy1

tank1.x += round(m.sin(((m.pi)/180) \* tank1.deg) \* tank1.vel)

tank1.dim[0] = tank1.x

if tank1.movementcheck(tank1, walls) == 'n':

tank1.x = tempx1

if keys[p.K\_DOWN]: #if down key is pressed then tank movement is pressed backward.

tempx1 = tank1.x

tempy1 = tank1.y

tank1.y += round(m.cos(((m.pi)/180) \* tank1.deg ) \* tank1.vel)

tank1.dim[1] = tank1.y

if tank1.movementcheck(tank1, walls) == 'n':

tank1.y = tempy1

tank1.x -= round(m.sin(((m.pi)/180) \* tank1.deg) \* tank1.vel)

tank1.dim[0] = tank1.x

if tank1.movementcheck(tank1, walls) == 'n':

tank1.x = tempx1

if keys[p.K\_RIGHT]: #left and right key acts as joystick to rotate the player on its axis.

tank1.deg += 15

if keys[p.K\_LEFT]:

tank1.deg -= 15

if keys[p.K\_SPACE]: #pressing space adds a bullet in the tank;s bullets list.

bullets1.append(gun(tank1.x + 32, tank1.y + 32, 5, tank1.deg))

if keys[p.K\_w]: #same operations are performed for tank 2 as well just with different set of keys.

tempx2 = tank2.x

tempy2 = tank2.y

tank2.y -= round(m.cos(((m.pi)/180) \* tank2.deg ) \* tank2.vel)

tank2.dim[1] = tank2.y

if tank2.movementcheck(tank2, walls) == 'n':

tank2.y = tempy2

tank2.x += round(m.sin(((m.pi)/180) \* tank2.deg) \* tank2.vel)

tank2.dim[0] = tank2.x

if tank2.movementcheck(tank2, walls) == 'n':

tank2.x = tempx2

if keys[p.K\_s]:

tempx2 = tank2.x

tempy2= tank2.y

tank2.y += round(m.cos(((m.pi)/180) \* tank2.deg ) \* tank2.vel)

tank2.dim[1] = tank2.y

if tank2.movementcheck(tank2, walls) == 'n':

tank2.y = tempy2

tank2.x -= round(m.sin(((m.pi)/180) \* tank2.deg) \* tank2.vel)

tank2.dim[0] = tank2.x

if tank2.movementcheck(tank2, walls) == 'n':

tank2.x = tempx2

if keys[p.K\_d]:

tank2.deg += 15

if keys[p.K\_a]:

tank2.deg -= 15

if keys[p.K\_f]:

bullets2.append(gun(tank2.x + 32, tank2.y + 32, 5, tank2.deg))

if tank1.deg == 360: #this is logic to prevent the degrees going down from 0 or above than 360.

tank1.deg = 0

elif tank1.deg < 0:

tank1.deg = 345

if tank2.deg == 360:

tank2.deg = 0

elif tank2.deg < 0:

tank2.deg = 345

win.fill((255, 229, 180)) #filling the window with colors

for wal in walls: #animating the walls in the game.

wal.animate(win)

for bullet1 in bullets1: #animating all the bullets in the game.

bullet1.animate(win)

for bullet2 in bullets2:

bullet2.animate(win)

tank1.animate(win) #animating the playing tanks in the game.

tank2.animate(win)

win.blit(text1, textRect1) #showing the name of the tanks on top of them

win.blit(text2, textRect2)

p.display.update() #updating the window.

#ending the game if any bullet hits the tank.

if tank1y==False :

p.draw.rect(win, (0, 155, 0), (tank1.x, tank1.y, tank1.width, tank1.height))

run = False

if tank2y==False :

p.draw.rect(win, (0, 155, 0), (tank2.x, tank2.y, tank2.width, tank2.height))

run=False

p.display.flip() #updating the game again.

p.quit() #quitting the pygame window

END\_GAME(player1\_score,player2\_score)

def color2():

#destroy all items on previous page

tanklabel1.destroy()

tanker\_name1.destroy()

submit1.destroy()

start.destroy()

# option to choose color of tank for player 2

if tank\_color1=="red":

showcolor1.destroy()

if tank\_color1=="blue":

showcolor2.destroy()

if tank\_color1=="green":

showcolor3.destroy()

if tank\_color1=="white":

showcolor4.destroy()

if tank\_color1=="yellow":

showcolor5.destroy()

# option to choose name for player 2

def name2():

choice2.destroy()

blue2.destroy()

red2.destroy()

green2.destroy()

white2.destroy()

yellow2.destroy()

start1.destroy()

colorsee2.destroy()

# storing name in a variable

def namesub2():

global player\_name2

player\_name2=tanker\_name2.get()

# Starts the game by calling main game function

def GAME():

root.destroy()

MAIN\_GAME(0,0)

# image code for name banner

global imgt2

imgt2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "name2.png")))

global tanklabel1

tanklabel11=Label(image=imgt2)

tanklabel11.grid(row=0,column=0,rowspan=2,columnspan=9,padx=200)

tanklabel11.configure(highlightthickness=0)

# code for entry of name

global tanker\_name2

tanker\_name2=Entry(root,width=60)

tanker\_name2.grid(row=4,column=4,pady=100)

# button to store entered name in variable

global imgs2

imgs2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "submit.png")))

global submit1

submit2=Button(root,image=imgs2,command=namesub2)

submit2.grid(row=6,column=4)

# button to start the game

global imgst2

imgst2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "start game.png")))

global start

start2=Button(root,image=imgst2,command=GAME)

start2.grid(row=6,column=7)

# color for second tank

global color2

color2=""

def colorz2(color2):

global tank\_color2

tank\_color2=color2

def color22():

# Below the code selects what the user inputted and displays the tank in respective color

if tank\_color2=="r":

global imgr2

imgr2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "red.png")))

global showcolor11

showcolor11=Label(image=imgr2)

showcolor11.grid(row=5,column=2)

if tank\_color2=="b":

global imgb2

imgb2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "blue.png")))

global showcolor22

showcolor22=Label(image=imgb2)

showcolor22.grid(row=5,column=2)

if tank\_color2=="g":

global imgg2

imgg2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "green.png")))

global showcolor33

showcolor33=Label(image=imgg2)

showcolor33.grid(row=5,column=2)

if tank\_color2=="w":

global imgw2

imgw2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "white.png")))

global showcolor44

showcolor44=Label(image=imgw2)

showcolor44.grid(row=5,column=2)

if tank\_color2=="y":

global imgy2

imgy2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "yellow.png")))

global showcolor55

showcolor55=Label(image=imgy2)

showcolor55.grid(row=5,column=2)

# image code for player 2 choose color

global imgx2

imgx2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "player2.png")))

choice2=Label(image=imgx2)

choice2.grid(row=0,column=0,rowspan=2,columnspan=9,padx=200)

# button code for color blue

blue2=Button(root,bg="blue",width=20,height=10,command=lambda:colorz2("b"))

blue2.grid(row=3,column=1,pady=5)

# button code for color red

red2=Button(root,bg="red",width=20,height=10,command=lambda:colorz2("r"))

red2.grid(row=3,column=4,pady=5)

#button code for color green

green2=Button(root,bg="green",width=20,height=10,command=lambda:colorz2("g"))

green2.grid(row=3,column=7,pady=5)

# button code for color white

white2=Button(root,bg="white",width=20,height=10,command=lambda:colorz2("w"))

white2.grid(row=4,column=2,pady=30)

# button code for color yellow

yellow2=Button(root,bg="yellow",width=20,height=10,command=lambda:colorz2("y"))

yellow2.grid(row=4,column=5,pady=30)

#button code for seeing selected color

global imgchosen2

imgchosen2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "see color chosen.png")))

colorsee2=Button(root,image=imgchosen,command=color22)

colorsee2.grid(row=5,column=1)

# button code for going to name selection window

global imgst2

imgst2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "next.png")))

start1=Button(root,image=imgst2,command=name2)

start1.grid(row=5,column=7)

# stores name of player 1

def namesub1():

global player\_name1

player\_name1=tanker\_name1.get()

# choosing name of player 1

def name():

choice.destroy()

blue.destroy()

red.destroy()

green.destroy()

white.destroy()

yellow.destroy()

next1.destroy()

colorsee.destroy()

# image code for name selection of player 1

global imgt

imgt=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "name1.png")))

global tanklabel1

tanklabel1=Label(image=imgt)

tanklabel1.grid(row=0,column=0,rowspan=2,columnspan=9,padx=200)

tanklabel1.configure(highlightthickness=0)

# Entry widget for entering name

global tanker\_name1

tanker\_name1=Entry(root,width=60)

tanker\_name1.grid(row=4,column=4,pady=100)

# button for submiting name into variable

global imgs

imgs=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "submit.png")))

global submit1

submit1=Button(root,image=imgs,command=namesub1)

submit1.grid(row=6,column=4)

# button to go to selection color window

global imgst

imgst=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "next.png")))

global start

start=Button(root,image=imgst,command=color2)

start.grid(row=6,column=7)

global tank\_color1

# tank color selection for tank 1

tank\_color1=str()

color1=""

def colorz(color1):

global tank\_color1

tank\_color1=color1

root.configure(bg="brown")

# Text for choice of color

global imgx

imgx=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "player1.png")))

choice=Label(image=imgx)

choice.grid(row=0,column=0,rowspan=2,columnspan=9,padx=200)

# Option buttons for all the colors

blue=Button(root,bg="blue",width=20,height=10,command=lambda:colorz("b"))

blue.grid(row=3,column=1,pady=5)

red=Button(root,bg="red",width=20,height=10,command=lambda:colorz("r"))

red.grid(row=3,column=4,pady=5)

green=Button(root,bg="green",width=20,height=10,command=lambda:colorz("g"))

green.grid(row=3,column=7,pady=5)

white=Button(root,bg="white",width=20,height=10,command=lambda:colorz("w"))

white.grid(row=4,column=2,pady=30)

yellow=Button(root,bg="yellow",width=20,height=10,command=lambda:colorz("y"))

yellow.grid(row=4,column=5,pady=30)

root.update()

# functions for assigning value to color variable

def color():

if tank\_color1=="r":

global imgr

imgr=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "red.png")))

global showcolor1

showcolor1=Label(image=imgr)

showcolor1.grid(row=5,column=2)

if tank\_color1=="b":

global imgb

imgb=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "blue.png")))

global showcolor2

showcolor2=Label(image=imgb)

showcolor2.grid(row=5,column=2)

if tank\_color1=="g":

global imgg

imgg=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "green.png")))

global showcolor3

showcolor3=Label(image=imgg)

showcolor3.grid(row=5,column=2)

if tank\_color1=="w":

global imgw

imgw=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "white.png")))

global showcolor4

showcolor4=Label(image=imgw)

showcolor4.grid(row=5,column=2)

if tank\_color1=="y":

global imgy

imgy=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "yellow.png")))

global showcolor5

showcolor5=Label(image=imgy)

showcolor5.grid(row=5,column=2)

# button code for going to next window

global imgz

imgz=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "Next.png")))

next1=Button(root,image=imgz,command=name)

next1.configure(highlightthickness=0)

next1.grid(row=5,column=7)

# button code for seeing color of tank

global imgchosen

imgchosen=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "see color chosen.png")))

colorsee=Button(root,image=imgchosen,command=color)

colorsee.grid(row=5,column=1)

root.configure(bg="brown")

root.update()

# The function of the settings button

def settings():

# removing all the items from the previous window

Play\_button.destroy()

settings\_button.destroy()

exit\_button.destroy()

var1=StringVar()

var2=IntVar()

# image code for settings

global imgA

imgA=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "settings.png")))

setting=Label(image=imgA)

setting.grid(row=0,column=6)

# For viewing all the scores

def opendata():

def opend():

try:

reader=open("data.txt","r")

except:

pass

scores=reader.read()

my\_text.insert(END,scores)

top2=Toplevel()

top2.title("Tank Busters Score")

my\_text=Text(top2,width=50,height=10,font=("Helvetica",16))

opend()

my\_text.pack(pady=20)

# function to go back to menu page

def back():

menulbl.destroy()

musicon.destroy()

musicoff.destroy()

setting.destroy()

back\_button.destroy()

scores\_button.destroy()

menu\_page()

def play():

# play music with help of pygame inbuilt commands

p.mixer.music.load(os.path.join('Resources/Sounds', "battle.mp3"))

p.mixer.music.play(loops=2)

p.mixer.music.set\_volume(0.05)

def stop():

p.mixer.music.stop()

# button code for playing the music

global imgm1

imgm1=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "music on.png")))

musicon=Button(root,image=imgm1,command=play)

musicon.grid(row=1,column=6)

# button code for switching off the music

global imgm2

imgm2=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "music off.png")))

musicoff=Button(root,image=imgm2,command=stop)

musicoff.grid(row=2,column=6)

root.update()

# button code for opening data file to see score

global imgscore

imgscore=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "scores.png")))

scores\_button=Button(root,image=imgscore,command=opendata)

scores\_button.configure(highlightthickness=0)

scores\_button.grid(row=3,column=6)

# button code for going cabk to main menu

global imgB

imgB=ImageTk.PhotoImage(Image.open(os.path.join('Resources/Banners', "back.png")))

back\_button=Button(root,image=imgB,command=back)

back\_button.configure(highlightthickness=0)

back\_button.grid(row=4,column=6)

# The function of the exit button

def exit1():

root.destroy()

# All executions

menu\_page()

root.iconify()

startup()

root.deiconify()

root.mainloop()

**Limitations / Future Enhancements**

The game still has some limitations like-

* The hitbox of the tanks is not perfect
* The code is not efficiently written

**Bibliography**

[www.freecodecamp.org](http://www.freecodecamp.org)

[www.geeksforgeeks.org](http://www.geeksforgeeks.org)

[www.runestone.academy](http://www.runestone.academy)

[www.stackoverflow.com](http://www.stackoverflow.com)

\*\*\*\*\*\*\*\*\*\*\*\*End of Report\*\*\*\*\*\*\*\*\*\*