Computer Graphics And Multimedia Lab Project

CS 1633 | 1 Credit

Submitted to Manipal University, Jaipur
Towards the partial fulfilment for the Award of the Degree of
BACHELORS OF TECHNOLOGY
In Computers Science and Engineering
2021-2022

Akshay Jain
189301060

Shubham Maheshwari
189301129

Shreya Saxena
189301022

Under the guidance of

Mrs. Bali Devi

Department of Computer Science and Engineering
School of Computing and Information Technology
Manipal University Jaipur
Jaipur, Rajasthan

MANIPAL UNIVERSITY JAIPUR, RAJASTHAN - 303007

DEPARTMENT OF COMPUTER SCENCE AND ENGINEERING

CERTIFICATE

Certified that the mini project work entitled **“Pygame - PLANTS V/S ZOMBIES”** is a bonafide work carried out by **AKSHAY JAIN (189301060)**, **SHUBHAM MAHESHWARI (189301129) and SHREYA SAXENA (189301022),** in partial fulfillment for the award of Degree of **Bachelor of Engineering** in **Computer Science Engineering** of the **Manipal University Jaipur, Rajasthan** during the year **2020-2021** in the **Computer Graphics and Multimedia Lab.** The lab project report has been approved as it satisfies the academic requirements in respect of the mini project work prescribed for the course of Bachelor of Engineering Degree.

Signature of the Guide	Signature of the HOD
Mrs. Balie Devi	**Prof. Sandeep Joshi**
(Asst. Prof, Dept of CSE)	(HOD, Dept. of CSE)
MUJ	MUJ

ACKNOWLEDGEMENT

We would like to express our immense gratitude to all the people who have helped us and supported us in our zeal for the completion of this mini project. The success of this mini project would not have been certain if it weren't for their help, whose constant guidance and support encourages us in our stride towards a better future.

We would like to thank **Prof. Sandeep Joshi, H.O.D.** of Computer Science and Engineering Department, Manipal University Jaipur, Rajasthan, for supporting us and helping us out in every way. To MUJ for having given us this wonderful opportunity, to explore for ourselves the various possibilities in the field of Computer Graphics, and showcase our dedication and talent.

We express our sincere gratitude to our internal guide **Mrs. Bali Devi,** Asst. Professor, Computer Science Department, Manipal University Jaipur. We are thankful to her for clearing all our doubts and clearing all our queries and helping us in our understanding of the subject and thus, helping us in the success of our mini project.

We would also like to thank the entire faculty of the **Computer Science & Engineering Department** for their co-operation and suggestions without which the project wouldn't have been complete.

And also, not to forget mentioning our friends who have helped us in uplifting our zeal and giving a hand in the completion of our project.

We are truly grateful to the above-mentioned people who have contributed in the completion of this project.

- **Akshay Jain (189301060)**
- **Shubham Maheshwari (189301129)**
- **Shreya Saxena (189301022)**

ABSTRACT

The game developed using Pygame module of Python is an alteration of classical "Plants v/s Zombies" game by PopCap Games. It is a hit action-strategy adventure where you meet, greet, and defeat legions of hilarious zombies from the dawn of time, to the end of days. Dominate the lawn with your pal, the classical Peashooter, supercharge them with Plant Food, and devise the ultimate plan to protect your house from being intruded.

The game proceeds in forms of levels, with each level offering a new challenge in terms of number of zombies and the variety. The zombies appear in incremental swarms and their number increasing with each level. The plant gets duly rewarded for jabbing down each zombie in terms of movement speed and scores.

INDEX

List of Figures

CHAPTER 1

INTRODUCTION

The ancient Chinese proverb, “a picture is worth ten thousand words” became a cliché in our society. Graphics provides one of the most natural way of communication with the computer, since our highly developed 2D and 3D pattern recognition ability allows us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics thus permits extensive, high bandwidth user computer interaction. This significantly enhances our ability to understand data, to perceive trends and to visualize real and imaginary objects, indeed to create a “virtual world” that we can explore from arbitrary points and views. It makes communication more efficient; graphics makes possible higher quality and more precise results or products, greater productivity, and lower analysis and design cost.

Pygame is a Free and Open Source python programming language library for making multimedia applications like games built on top of the excellent SDL library. Like SDL, pygame is highly portable and runs on nearly every platform and operating system. Millions of people have downloaded Pygame itself, which is a whole lot of bits flying across the innerwebs.

CHAPTER 2

LITERATURE SURVEY

2.1 PRIMITIVE GRAPHICS SYSTEM

Computer graphics started with pen plotter model. We had Cathode Ray Tube Display showing the graphics. Each line drawn was a result of intense calculation which was a huge overhead a few years back.

2.2 GRAPHICS IN C/C++ - Graphics.h

The main use of <graphics.h> library is to include and facilitate graphical operations in your C/C++ program. C graphics using graphics.h functions or WinBGM (Windows 7) can be used to draw different shapes, display text in different fonts, change colors and many more. Using functions of graphics.h in Turbo c compiler you can make graphics programs, animations, projects and games. You can draw circles, lines, rectangles, bars and many other geometrical figures. You can change their colors using the available functions and fill them. Following is a list of functions of graphics.h header file. Every function is discussed with the arguments it needs, its description, possible errors while using that function and a sample c graphics program with its output.

It initializes the graphics system by loading the past graphics driver then changing the system into graphics mode. It also resets or initializes all graphics settings like color, palette, current position etc., to their default values. Below is the description of input parameters of initgraph function.

graphicsDriver : It is a pointer to an integer specifying the graphics driver to be used. It tells the compiler that what graphics driver to use or to automatically detect the drive. In all our programs we will use DETECT macro of graphics.h library that instruct compiler for auto detection of graphics driver.

graphicsMode : It is a pointer to an integer that specifies the graphics mode to be used. If *gdriver is set to DETECT, then initgraph sets *gmode to the highest resolution available for the detected driver.

driverDirectoryPath : It specifies the directory path where graphics driver files (BGI files) are located. If directory path is not provided, then it will search for driver files in current working directory directory. In all our sample graphics programs, you have to change path of BGI directory accordingly where you Turbo C++ compiler is installed.

2.3 EXISTING SYSTEM

The existing graphics systems were the graphics header in C/C++. These graphics system is not system independent. Moreover, the underlying hardware knowledge is important for proper working of the code. Moreover, only 2D graphics were supported. Complex graphics concepts like camera position, shading, 3D graphics, material properties were absent.

2.4 PROPOSED SYSTEM

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. Pygame was originally written by Pete Shinners to replace PySDL after its development stalled.

Pygame uses the Simple DirectMedia Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS

Minimum hardware requirement specifications are:

- **Processor**: 1.5GHz or faster
- **RAM**: 4 GB or more
- **HDD**: 5 GB free disk space
- **Keyboard**: US English QWERTY keyboard
- **Mouse**: Normal working mouse
- **Monitor**: 800 x 600 or better
- **Architecture**: x86 64-bit CPU (Intel / AMD architecture)

3.2 SOFTWARE REQUIREMENTS

Minimum software specifications are:

- **OS**: Ubuntu 16.06 (Linux 4.6) or MacOS/OSX 10.7 or Windows 7 or better
- Latest NVidia and/or Intel Drivers for GPU
- Tools, IDE, Compilers:
    - Python >= 3.5
    - SDL >= 1.2.15
    - NumPy >= 1.6.2 (optional)
    - Visual Studio Code 2020

CHAPTER 4

DESIGN

4.1 MOTIVATION

The Pygame “Plants v/s Zombies” is a classic recreation of the beloved PopCap game, which was launched in 5th May, 2009. However, since the gameplay turned out to boring and banal, it demanded some tweaks and twists so that players can feel the ecstasy and relive the days of immersive gameplay.

4.2 STATEMENT OF PROBLEM

The game needs to be recreated in Python using the module named Pygame with a view of enhancing the gameplay. The rudimentary requirements for the game need to be satisfied before working on other features. Plant should be user controllable with freedom to move in any direction. Random appearance of sun charges up the plant and fuel its speed. Also, user score should be improved with each knockdown.

4.3 OBJECTIVE OF THE PROBLEM

In this program, the main objective to recreate the classic “Plant’s v/s Zombies” game with some additional features and demonstrate the principle of computer graphics such as scaling, transition, rotation, luminous intensity and some other concepts.

The game starts with a screen which wary user of things to avoid. The Game gets over under 2 circumstances:

1. If 3 zombies touch and cross the left side of the lawn
2. If the plant gets in contact with the zombie in any possible manner.

Our beloved “Peashooter” not only shoots peas but the amount of peas can be controlled upon user discretion using a “SPACEBAR”. Also, the plant is free to move in any possible direction using the “ARROW KEYS”.

2 types of Zombies: - “Conehead” and “Regular” are incorporated in the game. The “Regular” one is faster in speed and offers 1 point, when knocked down whereas, the “Conehead” one is tougher to spot and killing it rewards user with 2 points.

The random appearance of the rotating “Sun”, keeps the user engaged and galvanize them to move. Upon touching the Sun, plants get rewarded with +5 score.

The game is time adaptive, with difficulty level increasing with time. With each level, the number of zombies and the speed increases.

The screen displays the current score and the no. of zombies trespassed the lawn so that user can keep a check on how the/she is performing.

CHAPTER 5

5. IMPLEMENTATION

```
###   PyGame with a SnowPea shoot bullet for defensing the ombie army coming
###   Some images are used in this program; those can be found in the GitHub repo
###   To run the game, instructions are given in the README.md file.

import Pygame, random, sys, time
from pygame.locals import *

#set up some variables
WINDOWWIDTH = 1024
WINDOWHEIGHT = 600
FPS = 60

MAXGOTTENPASS = 3
ZOMBIESIZE = 110 #includes newKindZombies
ADDNEWZOMBIERATE = 60
ADDNEWKINDZOMBIE = ADDNEWZOMBIERATE

NORMALZOMBIESPEED = 2
NEWKINDZOMBIESPEED = NORMALZOMBIESPEED / 2

PLAYERMOVERATE = 15
BULLETSPEED = 10
ADDNEWBULLETRATE = 15
MAXBULLETSPEED = 20
MINNEWBULLETRATE = 8

TEXTCOLOR = (255, 255, 255)
RED = (255, 0, 0)

angle = 0

def terminate():
    pygame.quit()
    sys.exit()
```

```python
def waitForPlayerToPressKey():
    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            if event.type == KEYDOWN:
                if event.key == K_ESCAPE: # pressing escape quits
                    terminate()
                if event.key == K_RETURN:
                    return

def playerHasHitZombie(playerRect, zombies):
    for z in zombies:
        if playerRect.colliderect(z['rect']):
            return True
    return False

def bulletHasHitZombie(bullets, z):
    for b in bullets:
        if b['rect'].colliderect(z['rect']):
            bullets.remove(b)
            return True
    return False

def bulletHasHitCrawler(bullets, c):
    for b in bullets:
        if b['rect'].colliderect(c['rect']):
            bullets.remove(b)
            return True
    return False

def drawText(text, font, surface, x, y):
    textobj = font.render(text, 1, TEXTCOLOR)
    textrect = textobj.get_rect()
    textrect.topleft = (x, y)
    surface.blit(textobj, textrect)
```

```
def rotate(surface, rect, angle):
    rotated_surface = pygame.transform.rotozoom(surface,angle,1)
    rotated_rect = rotated_surface.get_rect(center = rect.center)
    return rotated_surface,rotated_rect

def plant_touches_sun(playerRect, sunRect):
    if playerRect.colliderect(sunRect):
        return True
    else :
        return False

# set up pygame, the window, and the mouse cursor
pygame.init()
mainClock = pygame.time.Clock()
windowSurface = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))#, pygame.FULL
SCREEN)
pygame.display.set_caption('Zombie Defence')
pygame.mouse.set_visible(False)

# set up fonts
font = pygame.font.SysFont(None, 48)

# set up sounds
gameOverSound = pygame.mixer.Sound('gameover.wav')
pygame.mixer.music.load('grasswalk.mp3')

# set up images
playerImage = pygame.image.load('SnowPea.gif')
playerRect = playerImage.get_rect()

bulletImage = pygame.image.load('SnowPeashooterBullet.gif')
bulletRect = bulletImage.get_rect()

zombieImage = pygame.image.load('Normal.gif')

newKindZombieImage = pygame.image.load('ConeheadZombieAttack.png')

#sunImage = pygame.image.load("Sun.png")
```

```
#sunImage = pygame.transform.scale(sunImage,(75,75))
#sunRect = sunImage.get_rect(center=(600,300))


backgroundImage = pygame.image.load('background_2.jpg')
rescaledBackground = pygame.transform.scale(backgroundImage, (WINDOWWIDTH + 100, W
INDOWHEIGHT))


# show the "Start" screen
windowSurface.blit(rescaledBackground, (0, 0))
windowSurface.blit(playerImage, (WINDOWWIDTH / 2, WINDOWHEIGHT - 70))
drawText('Plants Vs Zombies', font, windowSurface, (WINDOWWIDTH / 4) + 80, (WINDOW
HEIGHT / 4))
drawText('Press Enter to start', font, windowSurface, (WINDOWWIDTH / 4) + 80, (WIN
DOWHEIGHT / 3) + 50)
drawText("Don't let 3 zombies touch the left side!", font, windowSurface, (WINDOWW
IDTH / 4) - 60, (WINDOWHEIGHT / 3) + 120)
drawText("Don't hit the zombie!", font, windowSurface, (WINDOWWIDTH / 4) + 80, (WI
NDOWHEIGHT / 3) + 160)
pygame.display.update()
waitForPlayerToPressKey()

x_coordinate = 0

while True:
    # set up the start of the game

    zombies = []
    newKindZombies = []
    bullets = []

    zombiesGottenPast = 0
    score = 0
    Level = 1
    Level_inc=True
    Level_change=20

    playerRect.topleft = (50, WINDOWHEIGHT /2)
```

```python
moveLeft = moveRight = False
moveUp=moveDown = False
shoot = False

zombieAddCounter = 0
newKindZombieAddCounter = 0
bulletAddCounter = 40

newSunTime=600
newSunRange=[600,1200]
sunTime=0
sunHorizontalRange=[60,WINDOWWIDTH-80]
sunVerticalRange=[60,WINDOWHEIGHT-40]
sunDrop=0
sunDropSpeed=5
sunAlpha=30
sun=0
sunappear=0
sunLimit=600
ADDNEWBULLETRATE=15
BULLETSPEED=10

pygame.mixer.music.play(-1, 0.0)



while True: # the game loop runs while the game part is playing
    for event in pygame.event.get():
        if event.type == QUIT:
            terminate()

        if event.type == KEYDOWN:
            if event.key == K_UP or event.key == ord('w'):
                moveDown = False
                moveUp = True
            if event.key == K_DOWN or event.key == ord('s'):
                moveUp = False
                moveDown = True
            if event.key == K_RIGHT or event.key == ord('d'):
                    moveRight = True
                    moveLeft = False
                if event.key == K_LEFT or event.key == ord('a'):
                    moveLeft = True
                    moveRight = False

                if event.key == K_SPACE:
                    shoot = True

            if event.type == KEYUP:
                if event.key == K_ESCAPE:
                        terminate()

                if event.key == K_UP or event.key == ord('w'):
                    moveUp = False
                if event.key == K_DOWN or event.key == ord('s'):
                    moveDown = False
                if event.key == K_RIGHT or event.key == ord('d'):
                    moveRight = False
                if event.key == K_LEFT or event.key == ord('a'):
                    moveLeft = False

                if event.key == K_SPACE:
                    shoot = False

        # Add new zombies at the top of the screen, if needed.
        zombieAddCounter += 1
        if zombieAddCounter == ADDNEWKINDZOMBIE:
            zombieAddCounter = 0
            zombieSize = ZOMBIESIZE
            newZombie = {'rect': pygame.Rect(WINDOWWIDTH, random.randint(10,WINDOW
HEIGHT-zombieSize-10), zombieSize, zombieSize),
                        'surface':pygame.transform.scale(zombieImage, (zombieSize,
 zombieSize)),
                        }

            zombies.append(newZombie)
        # Add new newKindZombies at the top of the screen, if needed.
        newKindZombieAddCounter += 1
        if newKindZombieAddCounter == ADDNEWZOMBIERATE:
            newKindZombieAddCounter = 0
            newKindZombiesize = ZOMBIESIZE
            newCrawler = {'rect': pygame.Rect(WINDOWWIDTH, random.randint(10,WINDO
WHEIGHT-newKindZombiesize-10), newKindZombiesize, newKindZombiesize),
                        'surface':pygame.transform.scale(newKindZombieImage, (70,
140)),
                        }
            newKindZombies.append(newCrawler)

        # add new bullet
        bulletAddCounter += 1
        if bulletAddCounter >= ADDNEWBULLETRATE and shoot == True:
            bulletAddCounter = 0
            newBullet = {'rect':pygame.Rect(playerRect.centerx+10, playerRect.cent
ery-25, bulletRect.width, bulletRect.height),
                         'surface':pygame.transform.scale(bulletImage, (bulletRect
.width, bulletRect.height)),
                        }
            bullets.append(newBullet)

        # add new sun
        sunTime+=1
        if sunTime == newSunTime:
            if sun!=0:
                sunTime=0
            else:
                sunHorizontal=random.randint(sunHorizontalRange[0],sunHorizontalRa
nge[1])
                sunVertical=random.randint(sunVerticalRange[0],sunVerticalRange[1]
)
                sunImage = pygame.image.load("Sun.png")
                sunImage = pygame.transform.scale(sunImage,(75,75))
                sunRect = sunImage.get_rect(center=(sunHorizontal,sunDrop))
                sun=1
                newSunTime=random.randint(newSunRange[0],newSunRange[1])
                sunAlphaSpeed=(255-30)//(sunVertical//sunDropSpeed)
                #print(sunRect)

        # Move the player around.
        if moveUp and playerRect.top > 30:
            playerRect.move_ip(0,-1 * PLAYERMOVERATE)
        if moveDown and playerRect.bottom < WINDOWHEIGHT-10:
            playerRect.move_ip(0,PLAYERMOVERATE)
        if moveRight and playerRect.right < WINDOWWIDTH - 10:
            playerRect.move_ip(PLAYERMOVERATE,0)
        if moveLeft and playerRect.left > 30:
            playerRect.move_ip(-1*PLAYERMOVERATE,0)

        # Move the zombies down.
        for z in zombies:
            z['rect'].move_ip(-1*NORMALZOMBIESPEED, 0)

        # Move the newKindZombies down.
        for c in newKindZombies:
            c['rect'].move_ip(-1*NEWKINDZOMBIESPEED,0)

        # move the bullet
        for b in bullets:
            b['rect'].move_ip(1 * BULLETSPEED, 0)

        # Delete zombies that have fallen past the bottom.
        for z in zombies[:]:
            if z['rect'].left < 0:
                zombies.remove(z)
                zombiesGottenPast += 1

        # Delete newKindZombies that have fallen past the bottom.
        for c in newKindZombies[:]:
            if c['rect'].left <0:
                newKindZombies.remove(c)
                zombiesGottenPast += 1

        for b in bullets[:]:
```

```python
        if b['rect'].right>WINDOWWIDTH:
            bullets.remove(b)

    # check if the bullet has hit the zombie
    for z in zombies:
        if bulletHasHitZombie(bullets, z):
            score += 1
            zombies.remove(z)

    for c in newKindZombies:
        if bulletHasHitCrawler(bullets, c):
            score += 1
            newKindZombies.remove(c)

    # check for collission of plant and sun
    if sun==2 and plant_touches_sun(playerRect,sunRect):
        if ADDNEWBULLETRATE>MINNEWBULLETRATE:
            ADDNEWBULLETRATE-=2
        if BULLETSPEED<MAXBULLETSPEED:
            BULLETSPEED+=4
        sun=0
        sunTime=0
        sunAlpha=30
        score+=10
        sunDrop=0
        sunappear=0
        #print('sun')

    if score!=0 and score%Level_change==0 and Level_inc:
        Level+=1
        Level_inc=False
        if ADDNEWZOMBIERATE>15:
            ADDNEWZOMBIERATE-=2
        if ADDNEWKINDZOMBIE>15:
            ADDNEWKINDZOMBIE-=2
    elif score%Level_change!=0 and not Level_inc:
        Level_inc=True
        rel_x = x_coordinate % rescaledBackground.get_rect().width
        # Draw the game world on the window.
        windowSurface.blit(rescaledBackground, (rel_x - rescaledBackground.get_rec
t().width, 0))
        if (rel_x < rescaledBackground.get_rect().width) :
             windowSurface.blit(rescaledBackground, (rel_x, 0))
        x_coordinate = x_coordinate - 1

        # Draw the player's rectangle, rails
        windowSurface.blit(playerImage, playerRect)

        # drop sun
        if sun==1:
            if sunDrop<sunVertical:
                if sunDrop+sunDropSpeed<sunVertical:
                    sunDrop+=sunDropSpeed
                else:
                    sunDrop=sunVertical
            else:
                sun=2
            if sunAlpha+sunAlphaSpeed<255:
                sunAlpha+=sunAlphaSpeed
            else:
                sunAlpha=255
            sunImage.set_alpha(sunAlpha)
            sunRect=sunImage.get_rect(center=(sunHorizontal,sunDrop))
            windowSurface.blit(sunImage,sunRect)

        # draw sun
        if sun==2:
            sunappear+=1
            if sunappear>=sunLimit:
                sun=0
                sunTime=0
                sunAlpha=30
                sunDrop=0
                sunappear=0
            angle = angle + 1
            sunImage.set_alpha(255)
            sun_rotated, sun_rotated_rect = rotate(sunImage,sunRect,angle)
            windowSurface.blit(sun_rotated,sun_rotated_rect)




        # Draw each baddie
        for z in zombies:
            windowSurface.blit(z['surface'], z['rect'])

        for c in newKindZombies:
            windowSurface.blit(c['surface'], c['rect'])

        # draw each bullet
        for b in bullets:
            windowSurface.blit(b['surface'], b['rect'])


        # Draw the score and how many zombies got past
        drawText('zombies gotten past: %s' % (zombiesGottenPast), font, windowSurf
ace, 10, 20)
        drawText('score: %s' % (score), font, windowSurface, 10, 50)
        drawText('Level: %s' % (Level),font,windowSurface,800,20)

        # update the display
        pygame.display.update()

        # Check if any of the zombies has hit the player.
        if playerHasHitZombie(playerRect, zombies):
            break
        if playerHasHitZombie(playerRect, newKindZombies):
            break

        # check if score is over MAXGOTTENPASS which means game over
        if zombiesGottenPast >= MAXGOTTENPASS:
            break
        mainClock.tick(FPS)


    # Stop the game and show the "Game Over" screen.
    pygame.mixer.music.stop()
    gameOverSound.play()
    time.sleep(1)
    if zombiesGottenPast >= MAXGOTTENPASS:
        windowSurface.blit(rescaledBackground, (0, 0))
        windowSurface.blit(playerImage, (WINDOWWIDTH / 2, WINDOWHEIGHT - 70))
        drawText('score: %s' % (score), font, windowSurface, 10, 30)
        drawText('Level: %s' % (Level), font, windowSurface, 800, 30)
        drawText('GAME OVER', font, windowSurface, (WINDOWWIDTH / 3) + 40, (WINDOWHEIGHT / 3))
        drawText('YOUR COUNTRY HAS BEEN DESTROIED', font, windowSurface, (WINDOWWIDTH / 4)- 80, (WINDOWHEIGHT / 3) + 100)
        drawText('Press enter to play again or escape to exit', font, windowSurface, (WINDOWWIDTH / 4) - 80, (WINDOWHEIGHT / 3) + 150)
        pygame.display.update()
        waitForPlayerToPressKey()
    if playerHasHitZombie(playerRect, zombies):
        windowSurface.blit(rescaledBackground, (0, 0))
        windowSurface.blit(playerImage, (WINDOWWIDTH / 2, WINDOWHEIGHT - 70))
        drawText('score: %s' % (score), font, windowSurface, 10, 30)
        drawText('Level: %s' % (Level), font, windowSurface, 800, 30)
        drawText('GAME OVER', font, windowSurface, (WINDOWWIDTH / 3)+40, (WINDOWHEIGHT / 3))
        drawText('YOU HAVE BEEN KISSED BY THE ZOMMBIE', font, windowSurface, (WINDOWWIDTH / 4) - 110, (WINDOWHEIGHT / 3) +100)
        drawText('Press enter to play again or escape to exit', font, windowSurface, (WINDOWWIDTH / 4) - 90, (WINDOWHEIGHT / 3) + 150)
        pygame.display.update()
        waitForPlayerToPressKey()
    if playerHasHitZombie(playerRect, newKindZombies):
        windowSurface.blit(rescaledBackground, (0, 0))
        windowSurface.blit(playerImage, (WINDOWWIDTH / 2, WINDOWHEIGHT - 70))
        drawText('score: %s' % (score), font, windowSurface, 10, 30)
        drawText('Level: %s' % (Level), font, windowSurface, 800, 30)
        drawText('GAME OVER', font, windowSurface, (WINDOWWIDTH / 3)+40, (WINDOWHEIGHT / 3))
        drawText('YOU HAVE BEEN KISSED BY THE ZOMMBIE', font, windowSurface, (WINDOWWIDTH / 4) - 110, (WINDOWHEIGHT / 3) +100)
        drawText('Press enter to play again or escape to exit', font, windowSurface, (WINDOWWIDTH / 4) - 90, (WINDOWHEIGHT / 3) + 150)
        pygame.display.update()
        waitForPlayerToPressKey()
    gameOverSound.stop()
```

CHAPTER 6

IMPORTANT FUNCTIONS

6.1 Libraries used:

Pygame: To include computer graphics and sound libraries.

Random: To generate random numbers.

Sys: To include system specific parameters and functions.

6.2 PyGame functions:

init( )	**:**	Initializes all imported Pygame modules
time.clock( )	**:**	Create an object to help track time
display.set_mode()	**:**	Initialize a window or screen for display
display.set_caption()	**:**	Sets the current window caption
mouse.set_visible( )	**:**	Hide or show the mouse cursor
font.SysFont( )	**:**	Create a font object from the system fonts
mixer.Sound( )	**:**	Create a new sound object from a from a file
mixer.music.load( )	**:**	Load a music file for playback
image.load( )	**:**	Load an image from a file
get_rect( )	**:**	Get the rectangular area of the surface
blit( )	**:**	Draw one image onto another
display.update( )	**:**	Update portions of the screen for software displays

move_ip( )	Moves the rectangle, in place
colliderect( )	Test if two rectangles overlap
transform.scale( )	Resize to new resolution
transform.rotozoom( )	Filtered scale and rotation
tick( )	Update the clock
quit( )	Uninitialize all Pygame modules

6.3 User Defined Functions:

terminate( )	This function is used to draw the outlines of some important parts of the scenery. The Lines seen in the output window at first are drawn through this function.
waitForPlayerToPressKey( ):	This function is used along the movesun( ) function in order give the perfect viewing for the viewer. If this part is removed from the program then there will be case of formation of number of suns as long as we use move sun.
playerHasHitZombie( ):	This function is used to show the movement of the sun at day time. It ends when the sun has reached the other end of the window while using the keyboard in order to move the sun.
bulletHasHitZombie( ):	This function is used to see the tree when the output screen is opened. It only consists of the outline of the tree.
bulletHasHitCrawler( ):	This function is used so that we can see the tree in the day time mode.

drawText( ) **:** This function is used to depict the night time of the program.

rotate( ) **:** This function is used to depict the day time of the program.

plant_touches_sun( ) : In order to show the different phases of the moon. This function can be accessed through keyboard keys.

6.4 Processing:

All the movements, like the zombies or plant or bullets are moved by changing the location of their rectangle in every frame using move_ip(). Zombie’s speed is fixed. Bullet’s speed is also fixed and increased whenever the plant touches the sun. Positions of these objects are changed by the factor of their speed in every loop. All the changes in positions of all elements are updated on the screen by display.update().

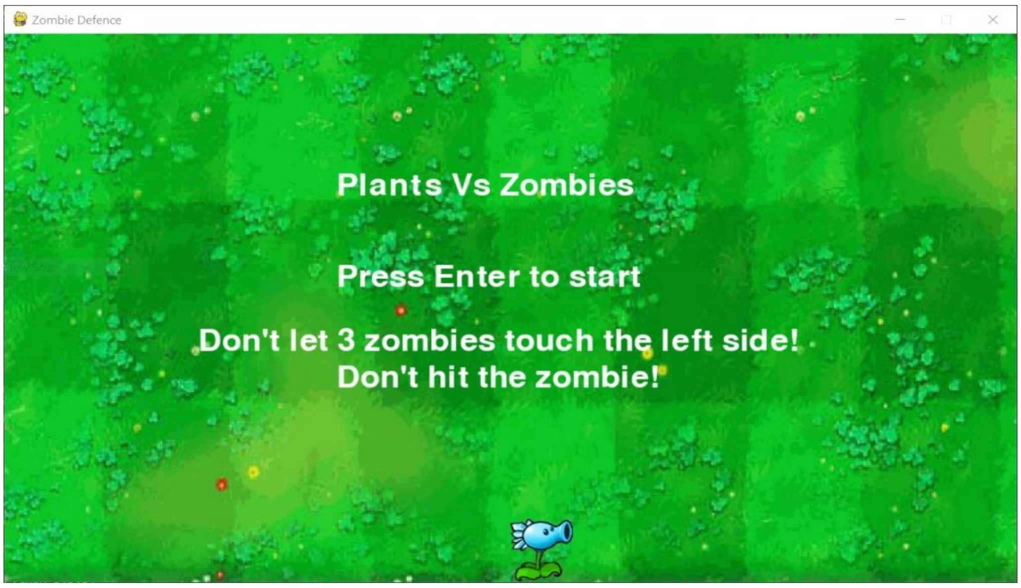
After changing coordinates or positions of all elements in the screen, before displaying them, if there is any collision it is checked and processed accordingly. If there is a collision in bullet and zombie, both the pieces disappear. If there is a collision between plant and zombie, Game Over. If there a collision between plant and sun, bullet’s speed and frequency increases.

When the plant touches a zombie or 3 zombies touches the left border of the screen, the Game is over, and the player receives a menu whether to start the game again or quit by pressing esc.

CHAPTER 7

7. SCREENSHOTS

7.1 WELCOME SCREEN



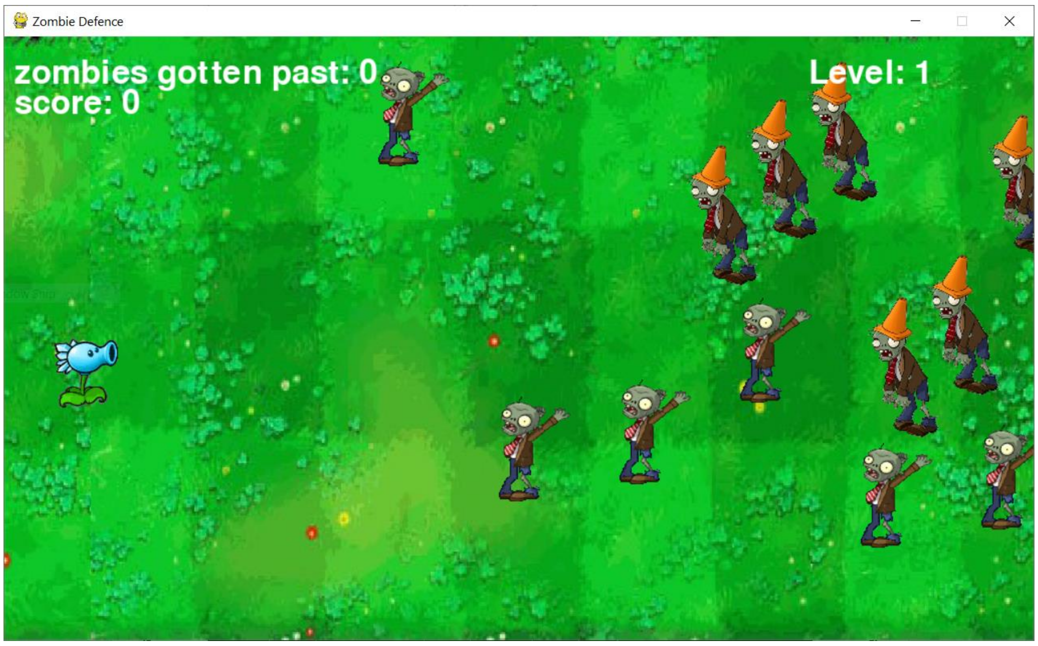
Plants Vs Zombies

Press Enter to start

Don't let 3 zombies touch the left side!
Don't hit the zombie!

Figure 4.1 Initial Output Window

7.2 GAME INITIALIZATION AFTER PRESSING ENTER



zombies gotten past: 0
score: 0

Level: 1

Figure 4.2 Initial Output Window after pressing Enter

7.3 USER GAMEPLAY

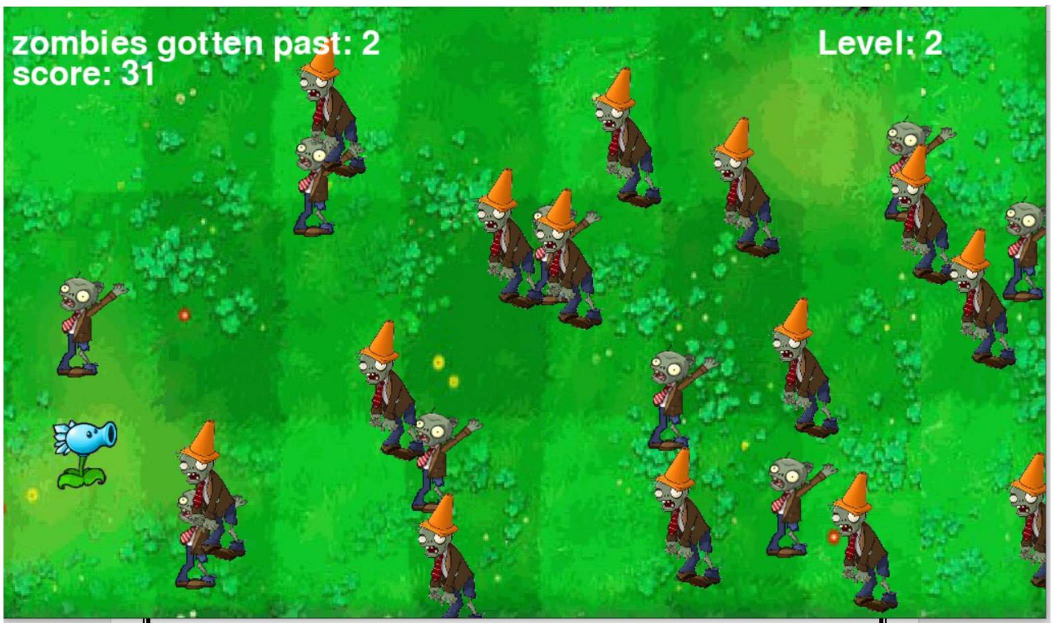


Figure 4.3 Snap of in-between Gameplay

7.4 RANDOM APPEARANCE OF ROTATING SUN

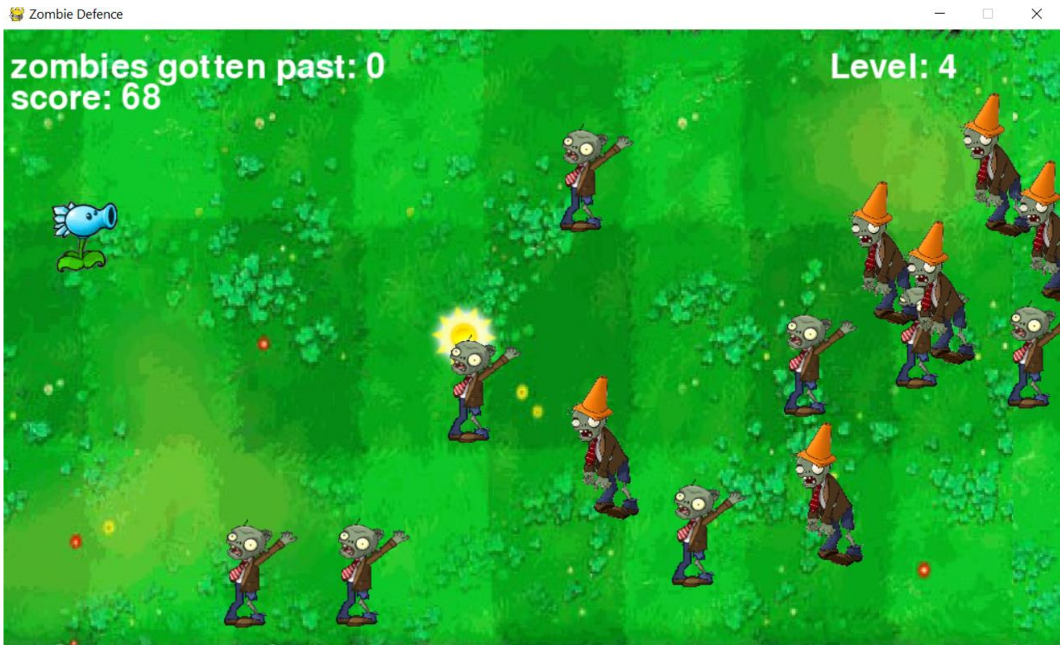
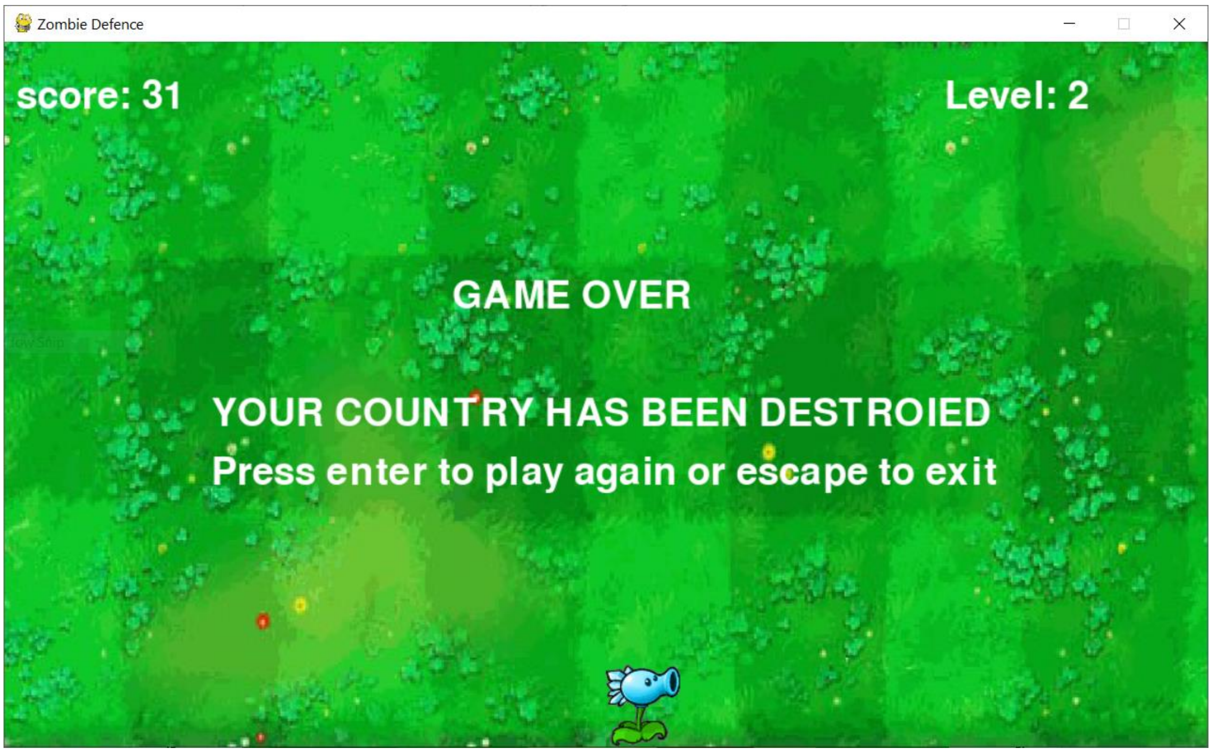


Figure 4.4 Rotating SUN appearing randomly

7.5 GAME OVER UPON 3 ZOMBIES CROSSING THE LAWN



GAME OVER

YOUR COUNTRY HAS BEEN DESTROIED

Press enter to play again or escape to exit

Figure 4.5 Game over (3 zombies crossed the lawn)

7.6 GAME OVER UPON TOUCHING THE ZOMBIE

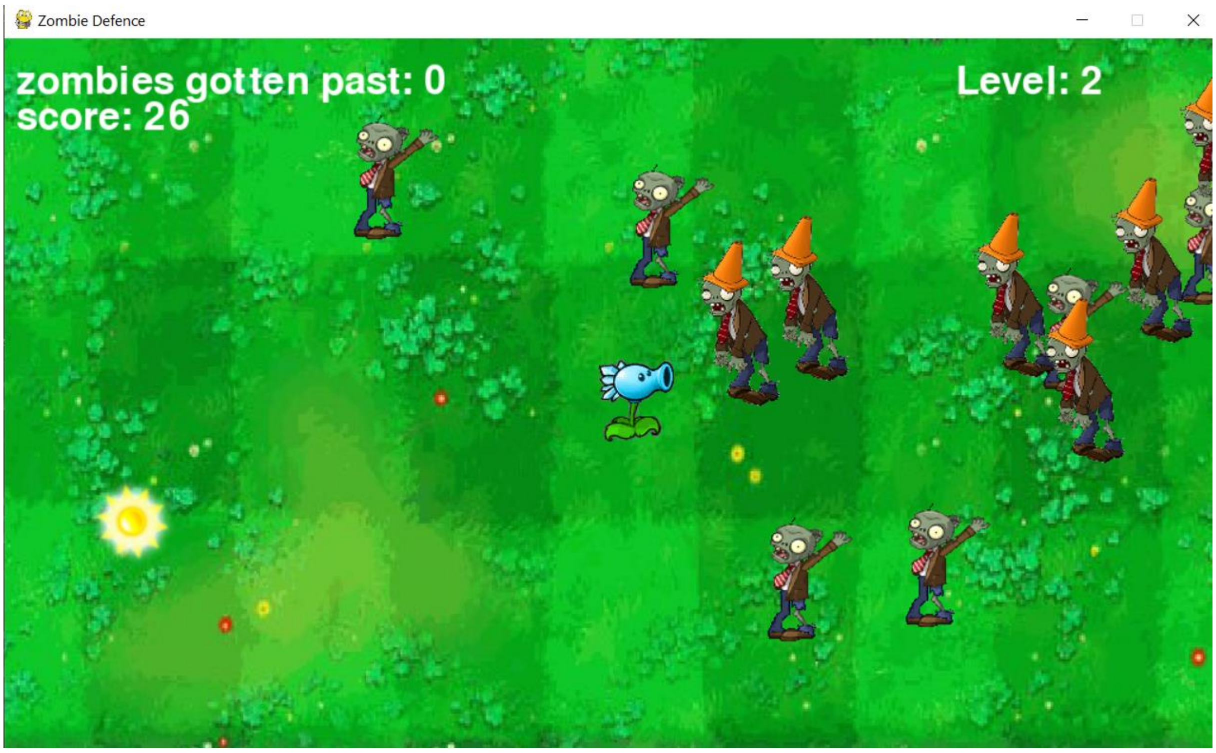
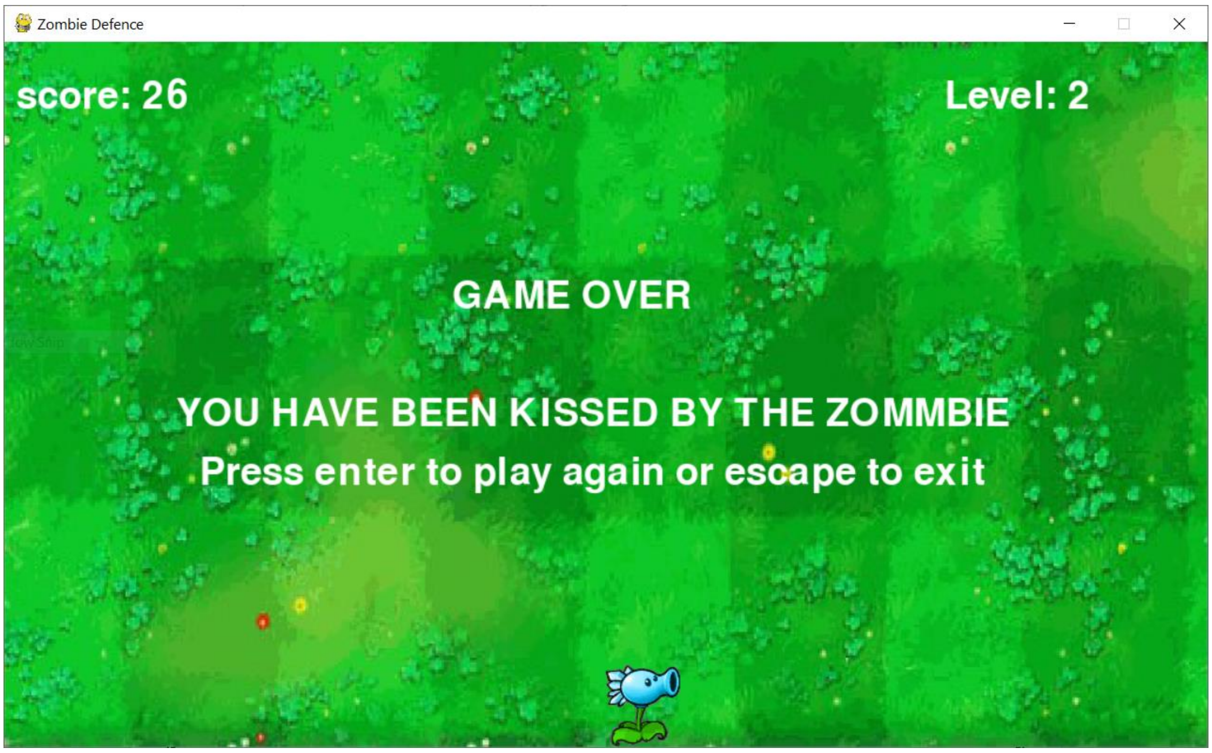


Figure 4.6 Game over (Touching Zombie)


Zombie Defence

score: 26

Level: 2

GAME OVER

YOU HAVE BEEN KISSED BY THE ZOMMBIE

Press enter to play again or escape to exit

Figure 4.7 Game over (Asking user to play again by pressing "ENTER" key)

CHAPTER 8

CONCLUSION

Through this project, we applied a pragmatic methodology to comprehend the essential concepts involved in computer graphics. We tried to understand how various concepts such as scaling, rotation, translation and reflection are used in real life scenarios.

Consequently, we build a project using Pygame which polished out programming skills and helped in learning more about Python as a multidisciplinary programming language.

CHAPTER 9

FUTURE ENHANCEMENTS

The project designed can be implemented in the future with more effects in lighting as well as functions. The Scene can be viewed in a 3D display. Further we can implement more user interaction as well as more functions which works according to System timing. The project can be implemented with the actual timing of the surrounding so that the landscape can be changed according to time, with difference in the game environment and associated viewing angles.

With an inclusion of variety of zombies, each with a peculiar characteristic can added in the game to make it more challenging for the user. Plants can be upgraded in terms of the type of peas it can shoot and can be upgraded accordingly.

Thus, in future this project has more scope in order to fulfill the viewing needs of the user and can offer more than what is considered an immersive gameplay experience.

BIBLIOGRAPHY

1. Books:

[1] Making Games with Python & Pygame by Albert Sweigart

2. Websites :

[1] https://www.pygame.org/wiki/GettingStarted

[2] https://pypi.org/project/pygame/

[3] https://www.javatpoint.com/pygame

[4] https://www.digitalocean.com/community/tutorials/how-to-install-pygame-and-create-a-template-for-developing-games-in-python-3

[5] https://www.geeksforgeeks.org/introduction-to-pygame/

[6] https://medium.com/iothincvit/pygame-for-beginners-234da7d3c56f

[7] http://programarcadegames.com/index.php?lang=en&chapter=libraries

[8] https://inventwithpython.com/pygame/

[9] https://realpython.com/pygame-a-primer/

3. Project GitHub Link:

https://github.com/akj2018/CGM-Project