**MODEL INSTITUTE OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)**

**(Permanently Affiliated to the University of Jammu, Accredited by NAAC with “A” Grade)**

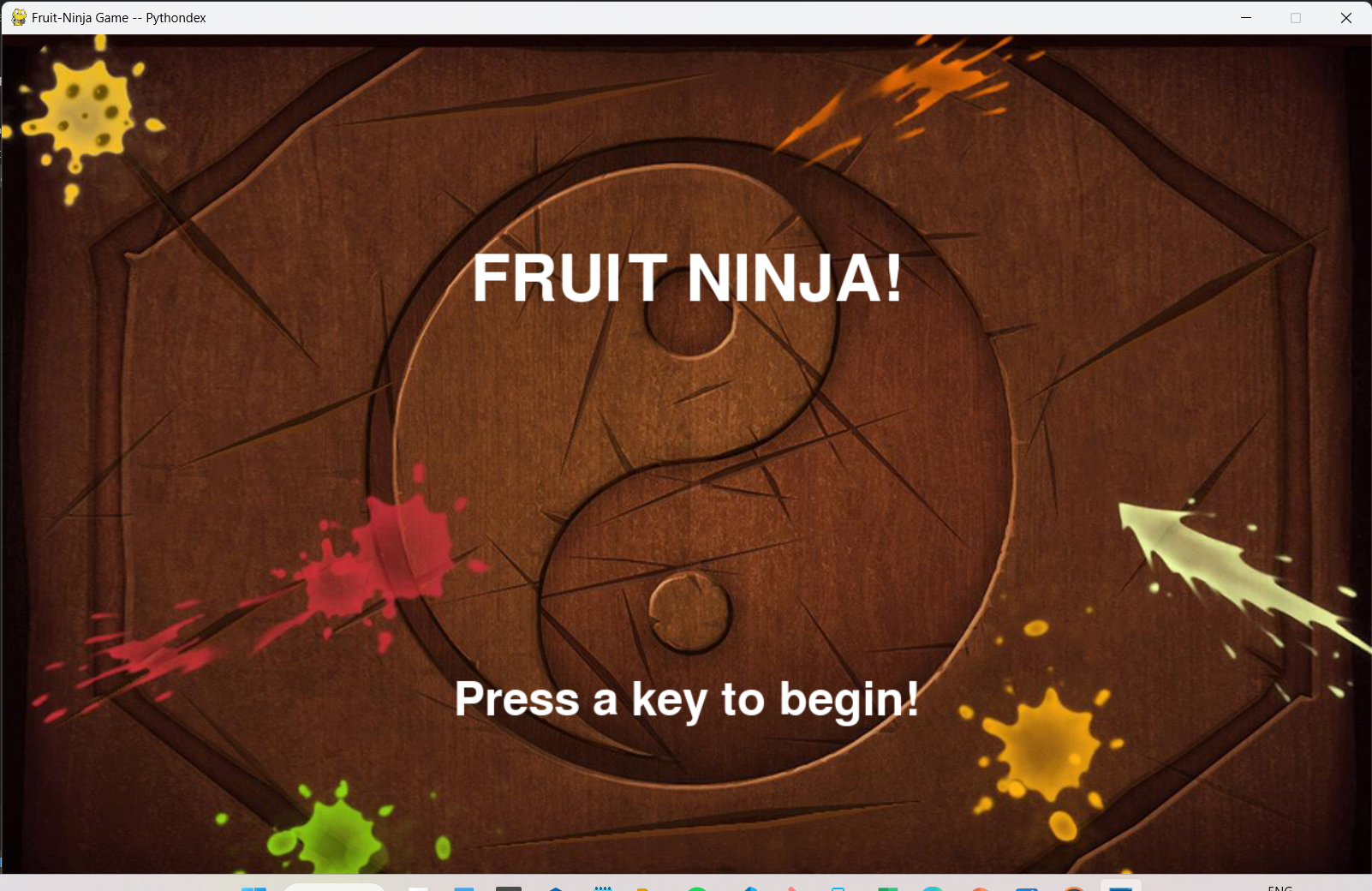
Minor project report| python

“Fruit Ninja Game”



By: Sahil Kumar, Rikit Gupta, Arya Dev Singh, Aryan Sharma

Submitted To: Mr Anil Gupta



Introduction

Fruit ninja game, also known as fruit-slicing game which is easy to play. Fruit ninja game is popular among children. The objective of this project is to build a fruit ninja game with python. This game is built with the help of “pygame” module and basic concept of python. In this game, the user has to cut the fruits by touching the mouse on fruits. There are also bombs with fruits. If the mouse touches more than three bombs then the game will be over

Project Requirement

In this python project, we require pygame, random, sys, and os module of python. Please install pygame and random:pip install pygamePip install random.

System Requirement

* Operating system: window 8 or higher
* Ram: min 2gb
* Processor: intel basic

Software used

* Vs code
* Python
* Jupyter notebook

Project File Structure

These are the steps to build fruit ninja game :

* Importing required modules
* Initialize window
* Define functions
* Game loop

Steps For Build a Game

Step 1:Importing required libraries

import pygame, sys

import os

import random

import time

Step2: Creating window display

player\_lives = 3                                                #keep track of lives

score = 0                                                       #keeps track of score

fruits = ['melon', 'orange', 'pomegranate', 'guava', 'bomb']    #entities in the game

# initialize pygame and create window

WIDTH = 1280

HEIGHT = 800

FPS = 4                                                #controls how often the gameDisplay should refresh. In our case, it will refresh every 1/12th second

pygame.init()

pygame.display.set\_caption('Fruit-Ninja Game -- Pythondex')

gameDisplay = pygame.display.set\_mode((WIDTH, HEIGHT))   #setting game display size

clock = pygame.time.Clock()

# Define colors

WHITE = (255,255,255)

BLACK = (0,0,0)

RED = (255,0,0)

GREEN = (0,255,0)

BLUE = (0,0,255)

background = pygame.image.load('background.jpg')                                #game background

font = pygame.font.Font(os.path.join(os.getcwd(), 'comic.ttf'), 42)

score\_text = font.render('Score : ' + str(score), True, (255, 255, 255)) #score display

lives\_icon = pygame.image.load('images/white\_lives.png')                    #images that shows remaining lives

* **player\_lives** will keep track of remaining lives
* **score** will keeps track of score
* **fruits** are the entities in the game
* **pygame.init()** initialize pygame
* **pygame.display.set\_caption** will set the caption of game window
* **FPS** controls how often the gameDisplay should refresh. In our case, it will refresh every 1/12th second
* **WIDTH and HEIGHT** are setting game display size by using pygame.display.set\_mode
* game background set by**pygame.image.load** which is used to set image
* **Lives-icon** stores images that show remaining lives

Step 3:Generalized structure of fruit dictionary

*def* generate\_random\_fruits(*fruit*):

    fruit\_path = "images/" + *fruit* + ".png"

    data[*fruit*] = {

        'img': pygame.image.load(fruit\_path),

        'x' : random.randint(100,500),          #where the fruit should be positioned on x-coordinate

        'y' : 800,

        'speed\_x': random.randint(-10,10),      #how fast the fruit should move in x direction. Controls the diagonal movement of fruits

        'speed\_y': random.randint(-80, -60),    #control the speed of fruits in y-directionn ( UP )

        'throw': False,                         #determines if the generated coordinate of the fruits is outside the gameDisplay or not. If outside, then it will be discarded

        't': 0,                                 #manages the

        'hit': False,

    }

    if random.random() >= 0.75:     #Return the next random floating point number in the range [0.0, 1.0) to keep the fruits inside the gameDisplay

        data[*fruit*]['throw'] = True

    else:

        data[*fruit*]['throw'] = False

# Dictionary to hold the data the random fruit generation

data = {}

for fruit in fruits:

    generate\_random\_fruits(fruit)

This function generates random fruits and generalized structure

* **‘x’** and **‘y’** store the value where the fruit should be positioned on x-coordinate and y – coordinate
* **Speed\_x** and **speed\_y** are key that store the value of how fast the fruit should move in the x and y-direction. It also controls the diagonal movement of fruits
* **throws** key used to determine that the generated coordinate of the fruits is outside the gameplay or not. If outside, then it will be discarded.
* Return the next random floating-point number in the range [0.0, 1.0) to keep the fruits inside the gameDisplay
* Data Dictionary used to hold the data of the random fruit generation

Step 4: Method to draw fonts

font\_name = pygame.font.match\_font('comic.ttf')

*def* draw\_text(*display*, *text*, *size*, *x*, *y*):

    font = pygame.font.Font(font\_name, *size*)

    text\_surface = font.render(*text*, True, WHITE)

    text\_rect = text\_surface.get\_rect()

    text\_rect.midtop = (*x*, *y*)

    gameDisplay.blit(text\_surface, text\_rect)

* **Draw\_text** function helps to draw text on the screen.
* **get\_rect()** return the Rect object.
* **X** and **y** is the dimension of x-direction and y-direction
* **blit()** draws image or writes text on the screen at a specified position

Step 5: Now for drawing lives

*def* draw\_lives(*display*, *y*, *z*, *lives*, *image*) :

    for i in range(*lives*) :

        img = pygame.image.load(*image*)

        img\_rect = img.get\_rect()       #gets the (x,y) coordinates of the cross icons (lives on the the top rightmost side)

        img\_rect.x = int(*y* + 45 \* i)    #sets the next cross icon 35pixels awt from the previous one

        img\_rect.y = *z*                  #takes care of how many pixels the cross icon should be positioned from top of the screen

*display*.blit(img, img\_rect)

* img\_rect gets the (x,y) coordinates of the cross icons (lives on the top rightmost side)
* img\_rect .x sets the next cross icon 35 pixels from the previous one
* img\_rect.y takes care of how many pixels the cross icon should be positioned from the top of the screen

Step 6: Show game over display & front display

# show game over display & front display

*def* show\_gameover\_screen():

    gameDisplay.blit(background, (0,0))

    draw\_text(gameDisplay, "FRUIT NINJA!", 90, WIDTH / 2, HEIGHT / 4)

    if not game\_over :

      draw\_text(gameDisplay,"Score : " + str(score), 50, WIDTH / 2, HEIGHT /2)

    draw\_text(gameDisplay, "Press a key to begin!", 64, WIDTH / 2, HEIGHT \* 3 /4)

    pygame.display.flip()

    waiting = True

    while waiting:

        clock.tick(FPS)

        for event in pygame.event.get():

            if event.type == pygame.QUIT:

                pygame.quit()

            if event.type == pygame.KEYUP:

                waiting = False

* **show\_gameover\_screen()** function shows the initial game screen and game over screen
* **pygame.display.flip()** will update only a part of screen but if no args will pass then it will update the entire screen
* **pygame.event.get()** will return all the event stored in the pygame event queue
* If event type is equal to quit then the pygame will quit
* **event.KEYUP** event that occurs when the key is pressed and released

step 7: Game loop

# Game Loop

first\_round = True

game\_over = True       #terminates the game While loop if more than 3-Bombs are cut

game\_running = True     #used to manage the game loop

while game\_running :

    if game\_over :

        if first\_round :

            show\_gameover\_screen()

        first\_round = False

        game\_over = False

        player\_lives = 3

        draw\_lives(gameDisplay, 690, 5, player\_lives, 'images/red\_lives.png')

        score = 0

    for event in pygame.event.get():

        # checking for closing window

        if event.type == pygame.QUIT:

            game\_running = False

    gameDisplay.blit(background, (0, 0))

    gameDisplay.blit(score\_text, (0, 0))

    draw\_lives(gameDisplay, 690, 5, player\_lives, 'images/red\_lives.png')

    for key, value in data.items():

        if value['throw']:

            value['x'] += value['speed\_x']          #moving the fruits in x-coordinates

            value['y'] += value['speed\_y']          #moving the fruits in y-coordinate

            value['speed\_y'] += (1 \* value['t'])    #increasing y-corrdinate

            value['t'] += 1                         #increasing speed\_y for next loop

            if value['y'] <= 800:

                gameDisplay.blit(value['img'], (value['x'], value['y']))    #displaying the fruit inside screen dynamically

            else:

                generate\_random\_fruits(key)

            current\_position = pygame.mouse.get\_pos()   #gets the current coordinate (x, y) in pixels of the mouse

            if not value['hit'] and current\_position[0] > value['x'] and current\_position[0] < value['x']+60 \

                    and current\_position[1] > value['y'] and current\_position[1] < value['y']+60:

                if key == 'bomb':

                    player\_lives -= 1

                    if player\_lives == 0:

                        hide\_cross\_lives(690, 15)

                    elif player\_lives == 1 :

                        hide\_cross\_lives(725, 15)

                    elif player\_lives == 2 :

                        hide\_cross\_lives(760, 15)

                    #if the user clicks bombs for three time, GAME OVER message should be displayed and the window should be reset

                    if player\_lives <= 0 :

                        show\_gameover\_screen()

                        game\_over = True

                    half\_fruit\_path = "images/explosion.png"

                else:

                    half\_fruit\_path = "images/" + "half\_" + key + ".png"

                value['img'] = pygame.image.load(half\_fruit\_path)

                value['speed\_x'] += 10

                if key != 'bomb' :

                    score += 1

                score\_text = font.render('Score : ' + str(score), True, (255, 255, 255))

                value['hit'] = True

        else:

            generate\_random\_fruits(key)

    pygame.display.update()

    clock.tick(FPS)      # keep loop running at the right speed (manages the frame/second. The loop should update afer every 1/12th pf the sec

pygame.quit()

* This is the mainloop of the game
* game\_over terminates the game while loop if more than 3-Bombs are cut
* game\_running used to manage the game loop
* If the event type is quit then the game window will be closed
* In this game loop we displaying the fruits inside the screen dynamically
* If a fruit is not cut then nothing will happen to it. if fruit cut, then a half-cut-fruit image should appear in place of that fruit
* if the user clicks bombs for three-time, a GAME OVER message should be displayed and the window should be reset
* clock.tick() will keep the loop running at the right speed (manages the frame/second). The loop should update after every 1/12th of the sec

project screenshots





Conclusion

Here we complete the project of making fruit ninja game using python its is very basic and easy to play game and in we used pygame library and functions for game .In this game when user click on bomb they lost one live there are total three lives in the game .After lost the every live the game will show game over screen.

Reference

1. <https://www.python.org/downloads/>

2. [Fruit Ninja - Slash Fruits in Air - Fruit NInja Game Review (debongo.com)](https://www.debongo.com/fruit-ninja-review/#iLightbox[gallery69]/0)

3. <https://www.geeksforgeeks.org/>