

|  |
| --- |
| Flappy bird IN python |
|  |
| January 8  Institute of Engineering and Management, Kolkata  Authored by:  1. Rohan Das (A\_14\_IT\_12019002004003) 2. Aritri Chowdhury (B\_97\_IT\_12019002004106) 3. Hritika Dey (B\_93\_IT\_12019002004101) 4. Ishak Debnath (B\_46\_IT\_12019002004046) |



# Abstract

# For this project, we chose to create the FLAPPY BIRD GAME which we used to play in old Android mobiles.

# The purpose of the project was to gather knowledge about different modules in Python Language, their different functions and how to use them.

# Using standard functions we were able to develop and implement most of the features, though it is not exactly perfect. We’ll discuss the drawbacks of the application in this report itself.

# Introduction

# Properly structured, group projects can reinforce skills that are relevant to both group and individual work, including the ability to:

# 1. Break complex tasks into parts and steps.

# 2. Plan and manage time.

# 3. Refine understanding through discussion and explanation.

# This report aims to provide a detailed look at the resulting application.

# 2.1) Background

# At the beginning of the project we had no prior experience of game development in Python Language. So a huge amount of time was invested investigating, understanding and testing smaller bits of functionality, as well as looking at the alternating implementations to figure out what we thought worked best.

# 2.2) Project Brief

# The main goal of this project was to develop a simple game to play and strengthen our Python Language skills.

# Application Details

# Different Python Libraries: First we’ll discuss about various python Libraries used in this Project.

# 

# As we can see that, we have imported random to generate random numbers for the pipe. Then we’ve imported the Python sys module which provides us functions and variables which are used to manipulate different parts of the Python Runtime Environment. Next, we’ve imported main module to for creating games in Python, which is pygame; pygame is a cross-platform set of Python modules designed for writing video games. The module named pygame.locals contains various constants used by pygame. Its contents are automatically placed in the pygame module namespace. However, an application can use pygame.locals to include only the pygame constants with a from pygame.locals import \*.

# Now, we’ll discuss about the Global Variables.

# 

# The first global variable is FPS which stands for Frames Per Second. We set it to 40.

# Next we set the SCREENWIDTH and SCREENHEIGHT variable to 289 and 511 respectively. These represents the width and height of the screen in terms of pixels.

# In pygame, everything is viewed on a single user-created [display](https://www.pygame.org/docs/ref/display.html), which can be a window or a full screen. The display is created using [.set\_mode()](https://www.pygame.org/docs/ref/display.html#pygame.display.set_mode), which returns a Surface representing the visible part of the window. We store the returned surface in a variable called SCREEN.

# After that, we set the y coordinate of the ground at SCREENHEIGHT \* 0.8. We’ve came across the value 0.8 by tuning.

# Then, we’re creating two empty dictionaries named GAME\_SPRITES and GAME\_SOUNDS, we’ll add elements to them later.

# Then we’re giving the file path of the images of the player, background and pipe to the Global Variables PLAYER, BACKGROUND and PIPE.

# Now, we’ll define the main function of the game.

# 

# pygame.init() initializes all imported pygame modules.

# pygame.time.Clock() create an object to help track time.

# pygame.display.set\_caption() sets the current window caption.

# Now, we’re adding the first key-value pair to the dictionary. Here the key is ‘numbers’ and the value is a tuple of size 10. Each element of the tuple is the images of the numbers 0-9.

# Let’s take an example.

# pygame.image.load('C:/Users/rohan/Desktop/Python Game/My Game/Images/Images/0.png').convert\_alpha()

# In this line of code, pygame.image.load load an image from a file. In the argument of load function we’re passing the full path of the image to be loaded. convert\_alpha() are both used to convert surfaces to the same pixel format as used by the screen.

# Similarly, we’re adding some other key-value pair with keys ‘message’, ‘base’, ‘pipe’ to the GAME\_SPRTIES dictionary.

# pygame.transform.rotate(pygame.image.load(PIPE).convert\_alpha(),180)

# This function rotates the image of the PIPE by 180o

# Next, we’re adding different game sounds to the GAME\_SOUNDS dictionary.

# pygame.mixer.Sound() function is used for that. In the argument of this function we pass the full path of the audio file.

# At last, we’re calling the openGame() and mainGame() function. We’ll discuss about the function one by one.

# Now, we’ll discuss about the function openGame()

# 

# Initially, we’re setting the x coordinate of the player to SCREENWIDTH/5 and y coordinate of the player to exactly to the mid of the screen. We’re getting the height of the player by the .get\_height() function.

# y\_player=int((SCREENHEIGHT-GAME\_SPRITES['player'].get\_height())/2)

# Then we use this code to calculate the y coordinate variable of the player.

# Similarly, we calculate the x and y coordinate variable of the image of the message to be blit on the screen.

# Then, we declare the x coordinate of the base to 0.

# Then, we’re running a while loop. Inside the while loop we’re running for loop.

# for event in pygame.event.get():

# pygame.event.get() get events from the queue.

# If the type of the event is QUIT type the the game is closed. Else if the type of the key is KEYDOWN the function execution is stopped, in that case we go the the mainGame() function. Else BACKGROUND, PIPE, MESSAGE and PLAYER is blitted on the screen. FPSCLOCK.tick(FPS) is just a measure of time in pygame module. It means that for every second at most FPS no of frames should pass.

# Now before defining the mainGame() function we’re defining other two important functions.

# Let’s discuss about getRandomPipe() function which generates random pipes.

# 

# This function generates two pipes, one upper pipe and the other one lower pipe in same coordinate. It returns a list which consists of two dictionaries which contains the x and y coordinate of both the pipes.

# Now let’s discuss about the isCollide() function.

# 

# This function is used when to stop the game. If the bird hits the any of the pipes or hits the ground, the game is stopped.

# Now, we’ll discuss broadly about the mainGame() function.

# 

# First we’re initializing the score to 0. Then we’re assigning the x and y coordinate of the player some value.

# Then we’re generating two random pipes and storing their coordiantes in the list upperPipes and lowerPipes.

# We’re assigning pipeVelX, playerVelY, playerMaxVelY, playerMinVelY, playerAccY, playerFlapAccv and playerFlapped some value. We came across the required value by tuning.

# 

# Here is the main logic function which is used to drive the game smoothly. The for loop is more or less similar to the above mentioned for loop block. Here, the only change is in the playerFlapped variables value; it indicates that the bird is flapped. And whenever the bird is flapped the ‘wing’ sound is played.

# Then, we do a crash test. If the result of the crash test is true the game ends.

# Now, we’re implementing the logic of incrementing the score of the player. The visual representation of the logic is, if the bird pass successfully within the upperPipe and lowerPipe without hitting any of the pipe, the score gets incremented by 1. And the ‘point’ sound is played.

# if 0<upperPipes[0]['x']<5:

# newpipe = getRandomPipe()

# upperPipes.append(newpipe[0])

# lowerPipes.append(newpipe[1])

# This if block generates new random pipes and append that to upperPipes and lowerPipes lists.

# if upperPipes[0]['x'] < -GAME\_SPRITES['pipe'][0].get\_width():

# upperPipes.pop(0)

# lowerPipes.pop(0)

# If the pipes x coordinate is less than 0, it is popped from the upeerPipes and lowerPipes list.

# Then we blit the upper and lower pipes on the screen.

# 

# The last few lines of the functions are used to blit the base, player and score in the screen.

# Difficulties Faced in Making the Project:

# The main problem of making the game is that, we’ve to tune a lot to get the perfect position of the GAME\_SPRITES. The Python Programming Language makes the simple game development process very easy. The pygame package makes it very much easy.

# Conclusion

# It was an exciting project to work on and there is a lot I learnt from it, above and beyond its original scope. I was able to study and train myself on development in an environment that was almost completely foreign to me, which I believe I have been reasonably successful with.

# Acknowledgement:

We would like to express our special thanks of gratitude to my teacher Moumita Ma'am as well as our principal Swagatam Sir who gave us the golden opportunity to do this wonderful project on the topic, which also helped us in doing a lot of research and we came to know about so many new things we are really thankful to them.