**OPEN ENDED PROJECT**

**Flapping Bird Game**

**1. Overview**

Project Name: Hacking Bird  
Programming Language: Python  
Libraries Used: Pygame, Random, Sys

Objective: This initiative presents an adapted iteration of the traditional Flappy Bird game, in which players guide a bird through a series of randomly generated pipes to accumulate points. The game incorporates unique assets and sound effects to enrich the overall gaming experience.

**2. Implementation**

* import random # For generating random numbers
* import sys # We will use sys.exit to exit the program
* import pygame
* from pygame.locals import \* # Basic pygame imports
* # Global variables for the game
* FPS = 32
* SCREENWIDTH = 230
* SCREENHEIGHT = 510
* SCREEN = pygame.display.set\_mode((SCREENWIDTH, SCREENHEIGHT))
* GROUNDY = SCREENHEIGHT \* 0.8
* GAME\_IMAGES = {}
* GAME\_AUDIO = {}
* # Asset paths
* PLAYER = 'Gallery/images/bird.png'
* BACKGROUND = 'Gallery/images/background.jpg'
* pipe = 'Gallery/images/pipe.jpg'
* def welcomeScreen():
* """
* Shows welcome images on the screen
* """
* playerx = int(SCREENWIDTH/50)
* playery = int((SCREENHEIGHT - GAME\_IMAGES['player'].get\_height())/20)
* startx = int((SCREENWIDTH - GAME\_IMAGES['start'].get\_width())/2)
* starty = int(SCREENHEIGHT\*0.13)
* basex = 0
* while True:
* for event in pygame.event.get():
* # if user clicks on cross button, close the game
* if event.type == QUIT or (event.type==KEYDOWN and event.key == K\_ESCAPE):
* pygame.quit()
* sys.exit()
* # If the user presses space or up key, start the game for them
* elif event.type==KEYDOWN and (event.key==K\_SPACE or event.key == K\_UP):
* return
* else:
* SCREEN.blit(GAME\_IMAGES['background'], (0, 0))
* SCREEN.blit(GAME\_IMAGES['player'], (playerx, playery))
* SCREEN.blit(GAME\_IMAGES['start'], (startx,starty ))
* SCREEN.blit(GAME\_IMAGES['base'], (basex, GROUNDY))
* pygame.display.update()
* FPSCLOCK.tick(FPS)
* def mainGame():
* score = 0
* playerx = int(SCREENWIDTH/5)
* playery = int(SCREENWIDTH/2)
* basex = 0
* # Create 2 pipes for blitting on the screen
* newpipe1 = getRandompipe()
* newpipe2 = getRandompipe()
* # my List of upper pipes
* upperpipes = [
* {'x': SCREENWIDTH+200, 'y':newpipe1[0]['y']},
* {'x': SCREENWIDTH+200+(SCREENWIDTH/2), 'y':newpipe2[0]['y']},
* ]
* # my List of lower pipes
* lowerpipes = [
* {'x': SCREENWIDTH+200, 'y':newpipe1[1]['y']},
* {'x': SCREENWIDTH+200+(SCREENWIDTH/2), 'y':newpipe2[1]['y']},
* ]
* pipeVelX = -4
* playerVelY = -9
* playerMaxVelY = 10
* playerMinVelY = -8
* playerAccY = 1
* playerFlapAccv = -8 # velocity while flapping
* playerFlapped = False # It is true only when the bird is flapping
* while True:
* for event in pygame.event.get():
* if event.type == QUIT or (event.type == KEYDOWN and event.key == K\_ESCAPE):
* pygame.quit()
* sys.exit()
* if event.type == KEYDOWN and (event.key == K\_SPACE or event.key == K\_UP):
* if playery > 0:
* playerVelY = playerFlapAccv
* playerFlapped = True
* GAME\_AUDIO['wing'].play()
* crashTest = isCollide(playerx, playery, upperpipes, lowerpipes) # This function will return true if the player is crashed
* if crashTest:
* return
* #check for score
* playerMidPos = playerx + GAME\_IMAGES['player'].get\_width()/2
* for pipe in upperpipes:
* pipeMidPos = pipe['x'] + GAME\_IMAGES['pipe'][0].get\_width()/2
* if pipeMidPos<= playerMidPos < pipeMidPos +4:
* score +=1
* print(f"Your score is {score}")
* GAME\_AUDIO['point'].play()
* if playerVelY <playerMaxVelY and not playerFlapped:
* playerVelY += playerAccY
* if playerFlapped:
* playerFlapped = False
* playerHeight = GAME\_IMAGES['player'].get\_height()
* playery = playery + min(playerVelY, GROUNDY - playery - playerHeight)
* # move pipes to the left
* for upperpipe , lowerpipe in zip(upperpipes, lowerpipes):
* upperpipe['x'] += pipeVelX
* lowerpipe['x'] += pipeVelX
* # Add a new pipe when the first is about to cross the leftmost part of the screen
* if 0<upperpipes[0]['x']<5:
* newpipe = getRandompipe()
* upperpipes.append(newpipe[0])
* lowerpipes.append(newpipe[1])
* # if the pipe is out of the screen, remove it
* if upperpipes[0]['x'] < -GAME\_IMAGES['pipe'][0].get\_width():
* upperpipes.pop(0)
* lowerpipes.pop(0)
* # Lets blit our sprites now
* SCREEN.blit(GAME\_IMAGES['background'], (0, 0))
* for upperpipe, lowerpipe in zip(upperpipes, lowerpipes):
* SCREEN.blit(GAME\_IMAGES['pipe'][0], (upperpipe['x'], upperpipe['y']))
* SCREEN.blit(GAME\_IMAGES['pipe'][1], (lowerpipe['x'], lowerpipe['y']))
* SCREEN.blit(GAME\_IMAGES['base'], (basex, GROUNDY))
* SCREEN.blit(GAME\_IMAGES['player'], (playerx, playery))
* myDigits = [int(x) for x in list(str(score))]
* width = 0
* for digit in myDigits:
* width += GAME\_IMAGES['numbers'][digit].get\_width()
* Xoffset = (SCREENWIDTH - width)/2
* for digit in myDigits:
* SCREEN.blit(GAME\_IMAGES['numbers'][digit], (Xoffset, SCREENHEIGHT\*0.12))
* Xoffset += GAME\_IMAGES['numbers'][digit].get\_width()
* pygame.display.update()
* FPSCLOCK.tick(FPS)
* def isCollide(playerx, playery, upperpipes, lowerpipes):
* if playery> GROUNDY - 25 or playery<0:
* GAME\_AUDIO['hit'].play()
* return True
* for pipe in upperpipes:
* pipeHeight = GAME\_IMAGES['pipe'][0].get\_height()
* if(playery < pipeHeight + pipe['y'] and abs(playerx - pipe['x']) < GAME\_IMAGES['pipe'][0].get\_width()):
* GAME\_AUDIO['hit'].play()
* return True
* for pipe in lowerpipes:
* if (playery + GAME\_IMAGES['player'].get\_height() > pipe['y']) and abs(playerx - pipe['x']) < GAME\_IMAGES['pipe'][0].get\_width():
* GAME\_AUDIO['hit'].play()
* return True
* return False
* def getRandompipe():
* """
* Generate positions of two pipes(one bottom straight and one top rotated ) for blitting on the screen
* """
* pipeHeight = GAME\_IMAGES['pipe'][0].get\_height()
* offset = SCREENHEIGHT/3
* y2 = offset + random.randrange(0, int(SCREENHEIGHT - GAME\_IMAGES['base'].get\_height() - 1.2 \*offset))
* pipeX = SCREENWIDTH + 10
* y1 = pipeHeight - y2 + offset
* pipe = [
* {'x': pipeX, 'y': -y1}, #upper pipe
* {'x': pipeX, 'y': y2} #lower pipe
* ]
* return pipe
* if \_\_name\_\_ == "\_\_main\_\_":
* pygame.init()
* FPSCLOCK = pygame.time.Clock()
* pygame.display.set\_caption('Hacking Bird')
* # Load images
* GAME\_IMAGES['numbers'] = (
* pygame.image.load('Gallery/images/0.png').convert\_alpha(),
* pygame.image.load('Gallery/images/1.png').convert\_alpha(),
* pygame.image.load('Gallery/images/2.png').convert\_alpha(),
* pygame.image.load('Gallery/images/3.png').convert\_alpha(),
* pygame.image.load('Gallery/images/4.png').convert\_alpha(),
* pygame.image.load('Gallery/images/5.png').convert\_alpha(),
* pygame.image.load('Gallery/images/6.png').convert\_alpha(),
* pygame.image.load('Gallery/images/7.png').convert\_alpha(),
* pygame.image.load('Gallery/images/8.png').convert\_alpha(),
* pygame.image.load('Gallery/images/9.png').convert\_alpha(),
* )
* GAME\_IMAGES['start'] = pygame.image.load('Gallery/images/front.png').convert\_alpha()
* GAME\_IMAGES['base'] = pygame.image.load('Gallery/images/base.jpg').convert\_alpha()
* GAME\_IMAGES['pipe'] = (
* pygame.transform.rotate(pygame.image.load(pipe).convert\_alpha(), 180),
* pygame.image.load(pipe).convert\_alpha()
* )
* GAME\_IMAGES['background'] = pygame.image.load(BACKGROUND).convert()
* GAME\_IMAGES['player'] = pygame.image.load(PLAYER).convert\_alpha()
* # Resize images
* PLAYER\_SIZE = (34, 24) # Example size for the bird
* BACKGROUND\_SIZE = (SCREENWIDTH, SCREENHEIGHT) # Background should cover the whole screen
* PIPE\_SIZE = (52, 320) # Example size for the pipe (adjust as needed)
* BASE\_SIZE = (SCREENWIDTH, 112) # Example size for the base
* START\_SIZE = (SCREENWIDTH, SCREENHEIGHT) # Example size for the start screen image (adjust as needed)
* # Resize loaded images
* GAME\_IMAGES['player'] = pygame.transform.scale(GAME\_IMAGES['player'], PLAYER\_SIZE)
* GAME\_IMAGES['background'] = pygame.transform.scale(GAME\_IMAGES['background'], BACKGROUND\_SIZE)
* GAME\_IMAGES['base'] = pygame.transform.scale(GAME\_IMAGES['base'], BASE\_SIZE)
* GAME\_IMAGES['start'] = pygame.transform.scale(GAME\_IMAGES['start'], START\_SIZE)
* # Resize pipes
* GAME\_IMAGES['pipe'] = (
* pygame.transform.scale(GAME\_IMAGES['pipe'][0], PIPE\_SIZE),
* pygame.transform.scale(GAME\_IMAGES['pipe'][1], PIPE\_SIZE)
* )
* # Load sounds
* GAME\_AUDIO['die'] = pygame.mixer.Sound('Gallery/Audio/die.mp3')
* GAME\_AUDIO['hit'] = pygame.mixer.Sound('Gallery/Audio/hit.mp3')
* GAME\_AUDIO['point'] = pygame.mixer.Sound('Gallery/Audio/point.mp3')
* GAME\_AUDIO['cyber'] = pygame.mixer.Sound('Gallery/Audio/cyber.mp3')
* GAME\_AUDIO['wing'] = pygame.mixer.Sound('Gallery/Audio/wing.mp3')
* while True:
* welcomeScreen()
* mainGame()

**3. Explanation**

The game mechanics are simple:

* The bird moves down due to gravity unless the player flaps, which gives an upward boost.
* Pipes are continuously generated and move leftward.
* If the bird collides with an obstacle or the ground, the game resets.
* The objective is to survive as long as possible and achieve the highest score.

Key Functions

* welcomeScreen(): Displays the start screen and waits for player input.
* mainGame(): Runs the game loop, updates positions, checks collisions, and keeps score.
* isCollide(): Checks if the bird has hit any obstacles.
* getRandompipe(): Generates random pipe heights.

**5. Output**

|  |  |
| --- | --- |
|  |  |

**Conclusion**

The "Hacking Bird" game effectively emulates the gameplay mechanics of Flappy Bird, while adding a unique flair through custom graphics and audio. It is an excellent educational project for those interested in game development utilizing Python and Pygame.