PLAYER 1: KARIM BOUAOUINA

PLAYER 2: MERIAM MGHAIETH

SEMESTRIAL PROJECT

HOCKEY-T 64

PYTHON VIDEOGAME

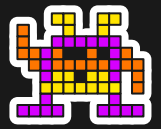

TABLE OF CONTENTS

Summary of how we plan on introducing to you this project:

01

INTRODUCTION

- Pygame is a popular Python module used for developing 2D video games and multimedia applications. It provides a variety of functions and tools for game development, including graphics and sound libraries, event handling, and user input management, without having to worry about low-level programming details.

Pong is a classic arcade game that was released in 1972 by Atari. It is a two-player game in which each player controls a paddle that can move up and down on their side of the screen. That inspired us to create our game, Hockey-T 64, an arcade hockey table, just like in real life, and whoever scores more than the other in under 30 seconds wins.

HOCKEY-T 64

02

GAME DESIGN

GAME FEATURES:

- Two-player gameplay using the same keyboard
- Simple controls that are easy to learn, making it accessible to players of all ages and skill levels.
- Responsive physics engine and accurate, allowing the ball to bounce realistically off the paddles and walls.
- Music in the background
- Scorekeeping allowing players to compete against each other
- Main menu that allow players to access important information and settings quickly and easily.

MAIN MENU:

When you start ‘Hockey-T 64’, you will be presented with the main menu, it gives you several options to choose from, three buttons in total:

PLAY

Default option and it will start a new game of Hockey-T 64 when you select it.

MENU

It will display a submenu that shows the button mapping for the game. This submenu can be useful if you need a quick reference for the keyboard controls used to play the game.

EXIT

It will allow you to exit the game and return to the desktop.


0 - 0
22s
INGAME SCREENSHOT:

03

CODE EXPLANATION

Libraries and Modules:

- import pygame: This library is used to create games and multimedia applications in Python.
- from pygame import mixer: This library is used to handle sound effects and music in Pygame.
- from pygame.locals import *: This line imports all of the constants and functions defined in the pygame.locals module.
- import os: This library provides a way to interact with the operating system in Python.
- import time: This library provides a way to measure time in Python.

```
import pygame
from pygame.locals import *
from pygame import mixer
import os
import time
```

Implemented Functions:

- main_menu():
  This function has a unique tricky feature of manipulating interfaces and images as an interactive working main menu using a list called
- “backgrounds”, allowing the user to scroll through the main menus different choices and to even pause or resume the background music

```
def main_menu():

    backgrounds = ["mmopt1.jpg", "mmopt2.jpg", "mmopt3.jpg"]
    global current_background
    current_background = 0
    global background
    background = pygame.image.load(backgrounds[current_background])
    window.blit(background, (0, 0))
    pygame.display.update()

    while True:
        ev = pygame.event.poll()
        if ev.type == pygame.KEYDOWN:
            if ev.key == pygame.K_RIGHT:
                current_background = (current_background + 1) % len(backgrounds)
                background = pygame.image.load(backgrounds[current_background])
                window.blit(background, (0, 0))
                pygame.display.update()
            elif ev.key == pygame.K_KP_PLUS:
                mixer.music.unpause()
            elif ev.key == pygame.K_KP_MINUS:
                mixer.music.pause()
            elif ev.key == pygame.K_ESCAPE:
                if current_background == 1:
                    current_background = 0
                    background = pygame.image.load(backgrounds[current_background])
            elif ev.key == pygame.K_RETURN:
                break
        if ev.type == QUIT:
            pygame.quit()
            quit()
    pygame.display.update()
```

Basic Python Implementation:

- We used basic ‘while’ looping to keep the window running and the main menu active when it needs to be

```
running = True
mainmenu = True
while running:
    if mainmenu:
        main_menu()
        mainmenu= False
        background = pygame.image.load(gamebackground)
        background.convert_alpha()
    if current_background == 1:
```

Paddles Movement

- The movement is based on manipulating the sprites based on the X and Y axis

```
ev = pygame.event.poll()
if ev.type == pygame.KEYDOWN:
    if ev.key == pygame.K_w:
        blue_paddle_y -= 20
    elif ev.key == pygame.K_s:
        blue_paddle_y += 20
    if ev.key == pygame.K_UP:
        red_paddle_y -= 20
    elif ev.key == pygame.K_DOWN:
        red_paddle_y += 20
```

Puck Physics

- The first two lines of code update the position of the puck based on its current speed in the x and y directions. Then, there are two conditional statements that check if the puck has hit the top or bottom of the screen. Then two conditional statements check if the puck has hit either of the paddles. The last two conditional statements make sure that the puck doesn't go beyond the top or bottom of the screen

```
puck_x += puck_speed_x
puck_y += puck_speed_y
if puck_y < 0 or puck_y > screen_height - puck.get_height():
    puck_speed_y *= -1
if puck_x < 0 or puck_x > screen_width - puck.get_width():
    puck_speed_x *= -1
if (puck_x < blue_paddle_x + blue_paddle.get_width() and
        puck_y + puck.get_height() > blue_paddle_y and
        puck_y < blue_paddle_y + blue_paddle.get_height()):
    puck_speed_x *= -1
if (puck_x + puck.get_width() > red_paddle_x and
        puck_y + puck.get_height() > red_paddle_y and
        puck_y < red_paddle_y + red_paddle.get_height()):
    puck_speed_x *= -1
```

```
if puck_y <= 0:
    puck_y = 0
elif puck_y + puck.get_height() >= screen_height:
    puck_y = screen_height - puck.get_height()

if puck_y + puck.get_height() >= screen_height - 1:
    puck_y = screen_height - puck.get_height()
```

Score Checking

The three conditional statements check if the puck has hit the left or right walls of the screen.

```
if puck_x < 0:
    score2 += 1
    puck_x = screen_width // 2
    puck_y = screen_height // 2
if puck_x + puck.get_width() > screen_width:
    score1 += 1
    puck_x = screen_width // 2
    puck_y = screen_height // 2
```

04

RESOURCES

SOFTWARES USED:

PIXELATOR: Making our own models for sprites, main menu...etc

Adobe Express: Removing background from images

Pycharm: Editing Environment

DOCUMENTATION:

PYGAME: https://www.pygame.org/docs/

05

FAQS

FAQS

- What were some of the big challenges you faced while developing the Hockey-T 64?
- Why did you approach designing the user interface as an image method instead of a real proper main menu for your game?
- How long did it take you to develop the game ?

CREDITS: KARIM BOUAOUINA & MERIAM MGHAIETH