Faculty of Computer and Information Technology
Computer Science Department

Final Project: AI244 & CS412

Weight: 15%, Due: May 31, 2023 @ 11:59pm
NO LATE PROJECT SUBMISSION WILL BE ACCEPTED

Notes:

- You can select any project of your choice from these four games.
- This is an individual project.
- Use high-quality images.
- Use sound effects in your game.
- Record a demo video to explain how your game works. The demo duration is 5 minutes at most.
- The e-learning system uploads files up to a maximum of 20 MB. If your project is larger than 20 MB, please upload it to a sharable Drive and provide a valid link along with your submission. (**Note:** We will check the modification date)
- The maximum mark for the project is **15 points**.

Submission guidelines:

- Submit a zipped folder containing your project.
- Name the project folder with the selected project name.
- Comment your name and ID at the top of the main.py file.
- Make sure that your program interprets and runs without errors. Otherwise, you will get **ZERO**.

Collaboration Policy

- This is an **individual project**. We take academic integrity seriously.
- The official policy is that it is acceptable to discuss general strategies with your fellow students, but when it comes to writing code, you should do it entirely alone. Do not share your code with anyone.
- Any student who turns in work copied (in whole or in part) from another student will result in both students getting a **ZERO**.

Program evaluation Criteria

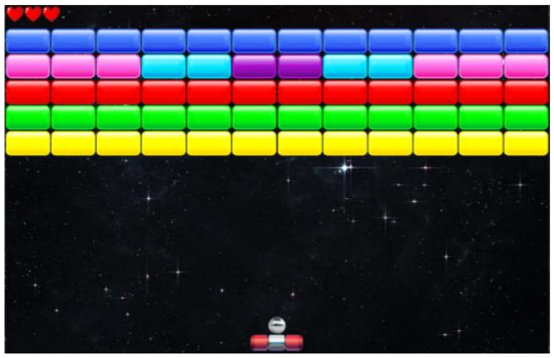
1. **Proper commenting and documenting: [10%]** your code must have comments to explain the purpose of each declared variable, method, and class.
2. **Creativity: [30%]** be creative, add **your own touches** to the game. You can use the design and colors you want. Try to add a **new feature(s)** to the game.
3. **Execution and output: [60%]** your code is executable, does not crash, and produces the expected game based on the provided requirements and rules below.

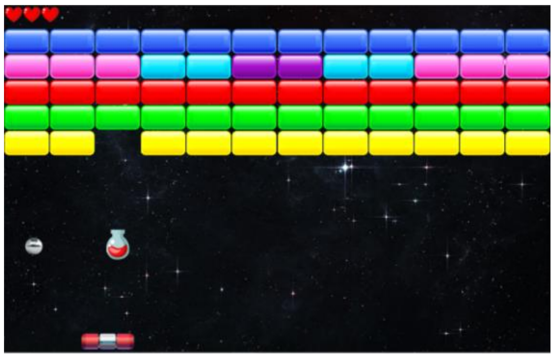
Project no. 1: “SMASH IT” Game

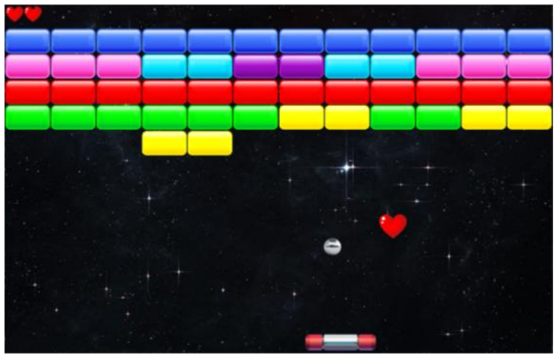
SMASH IT is a video game where the player controls a paddle or platform at the bottom of the screen, which is used to bounce a ball and break a wall of bricks located at the top of the screen. To implement this game, you need to do the following:

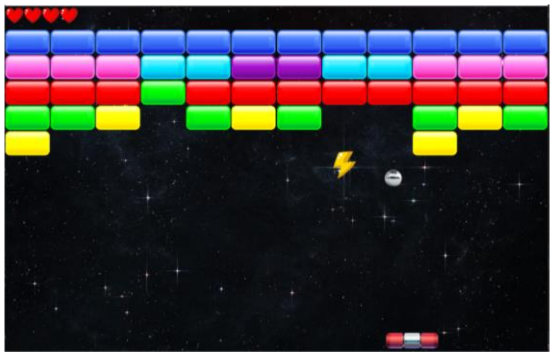
- As the player clears each level of the wall, the bricks become harder with new types of bricks that require multiple hits to break.
- The player usually starts with a set number of lives, which represents the number of times they can let the ball fall off the bottom of the screen before the game is over.
- The player loses a life each time the ball falls off the screen.
- The player can move the paddle left and right to hit the ball and prevent it from falling off the screen using arrow keys on the keyboard.
- To get an extra life, the player must usually collect a heart that appears randomly when certain bricks are destroyed.
- To extend the paddle, power-ups that increase the size of the paddle are represented by this icon . To collect the power-up, the player needs to hit the brick that contains it with the ball. Once collected, the player’s paddle will extend in size, making it easier to hit the ball and keep it in play.
- To speed up the paddle, collect a speed power-up by hitting specific bricks with the ball. This power-up increases the paddle's speed.
- The game ends when the player breaks all the bricks or runs out of lives.

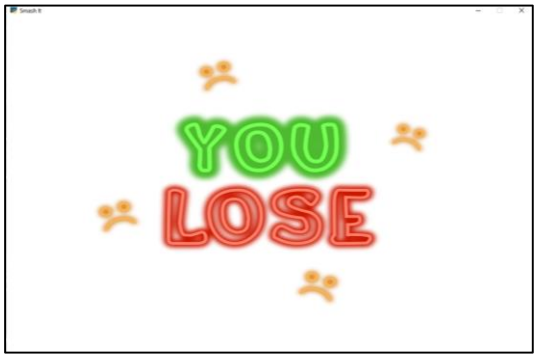
Sample Run:

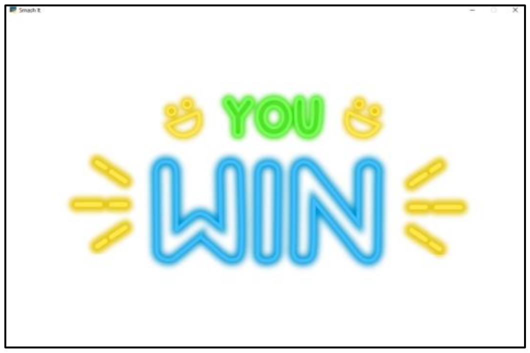












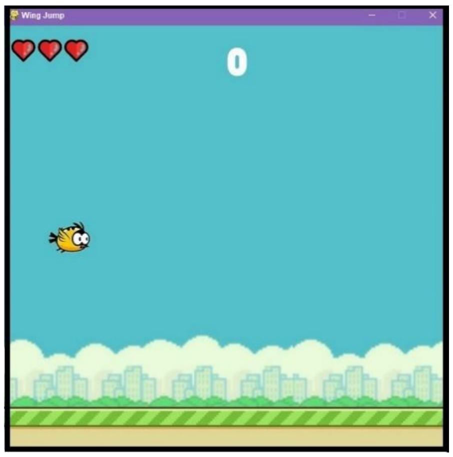
Project no. 2: “WING JUMP” Game

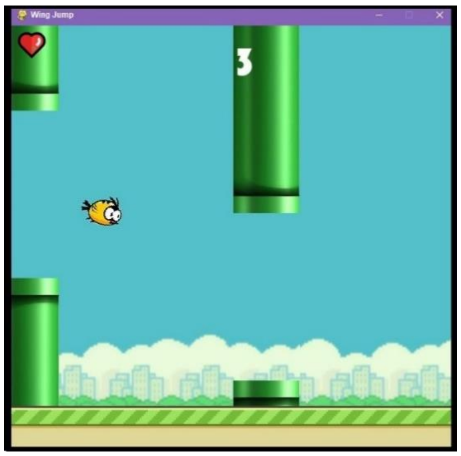
Wing Jump is a video game where the player controls a bird by tapping the screen to make it flap its wings and avoid obstacles, such as pipes. The objective is to fly as far as possible without crashing into the pipes.

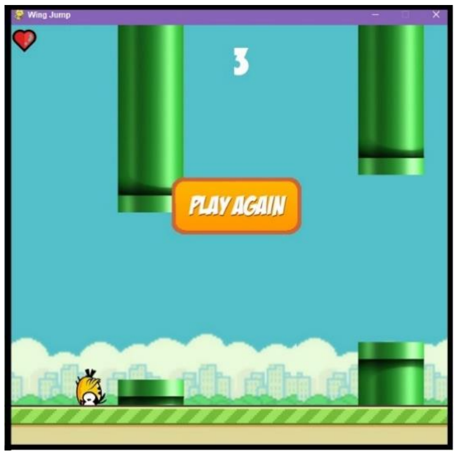
Plan the game mechanics and structure: Before diving into coding, it's important to have a clear idea of the game mechanics and how the game will be structured. This includes things like the bird's movement, the placement of obstacles, scoring system, and game over conditions. To implement this game, you need to do the following:

- Create the graphical assets: The game's graphics will play a crucial role in the player's experience. Create the bird, obstacles, and any other graphical assets that will be used in the game.
- Set up the game window: Use the GUI library to set up the game window and any necessary event handlers. This will involve creating the game loop, displaying the graphics, and handling user input.
- Implement the bird's movement: Use event handling to detect user input (e.g., mouse clicks or key presses) and update the bird's position and movement accordingly. You'll also need to implement gravity so that the bird falls when it's not flying.
- Implement obstacle spawning and collision detection: Generate the obstacles (e.g., pipes) and move them across the screen. Use collision detection to determine when the bird collides with an obstacle and trigger the game over condition.
- Implement the scoring system: Determine how the player earns points (e.g., by passing through gaps in the pipes) and update the score accordingly.
- Add a "life" system: When the player starts the game, they begin with a certain number of lives (e.g., three). Each time the bird collides with an obstacle, they lose a life. If the player runs out of lives, the game ends.
- Add a "Play Again" button: add a button labeled "Play Again" that the player can click to start a new game. When the button is clicked, the game should reset, and the player should start with the same number of lives as before.
- Test the game thoroughly: Test the game extensively to ensure that it works as intended and is enjoyable to play. This may involve tweaking game mechanics, adjusting difficulty levels, and fixing any bugs that are discovered.

Sample Run:





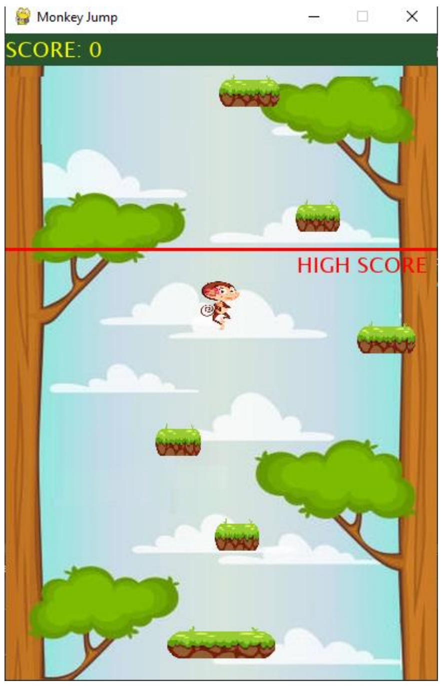


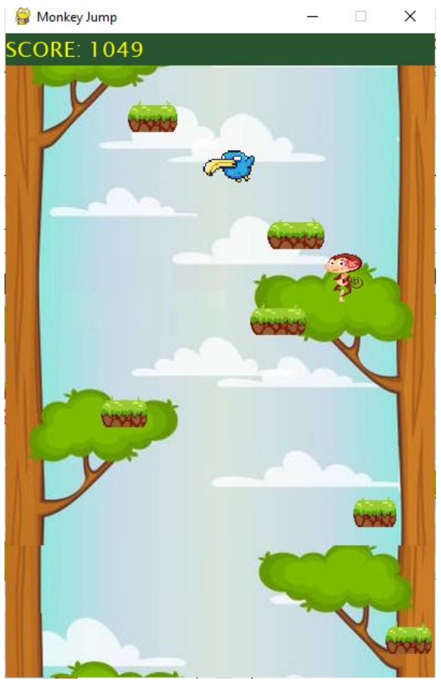
Project no. 3: “MONKEY JUMP” Game

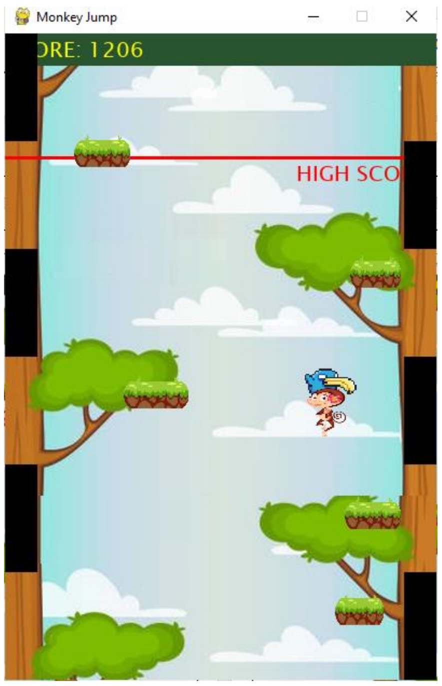
In **Monkey Jump**, the player controls a monkey as it jumps from platform to platform in an endless vertical scrolling environment. The goal is to climb as high as possible without falling off the bottom of the screen or being hit by birds. To implement this game, you need to do the following:

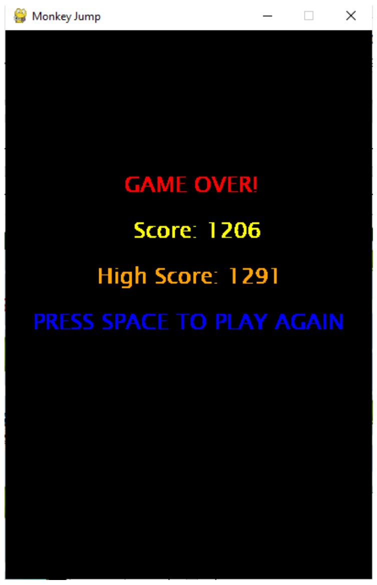
- The player is constantly bouncing on the platforms, and the only control that you have is to move him left and right using the arrow keys on the keyboard.
- The player will continue climbing higher and higher until he falls off and you have to start all over again.
- There is a score counter, which slowly increases as the player jumps up, and this shows how far the player has climbed.
- There is also a high score in this game at the top of the screen. It is always drawn as a line across the screen to basically mark the high score point as a sort of achievement line. When the player passes that line, you set a new high score.
- The score also allows control of the difficulty, so for the first thousand or so points everything is stationary; there are no enemies (birds).
- If the player got a little bit further up the platforms, the birds appeared, and these birds basically flew across the screen.

Sample Run:









GAME OVER!

Score: 1206

High Score: 1291

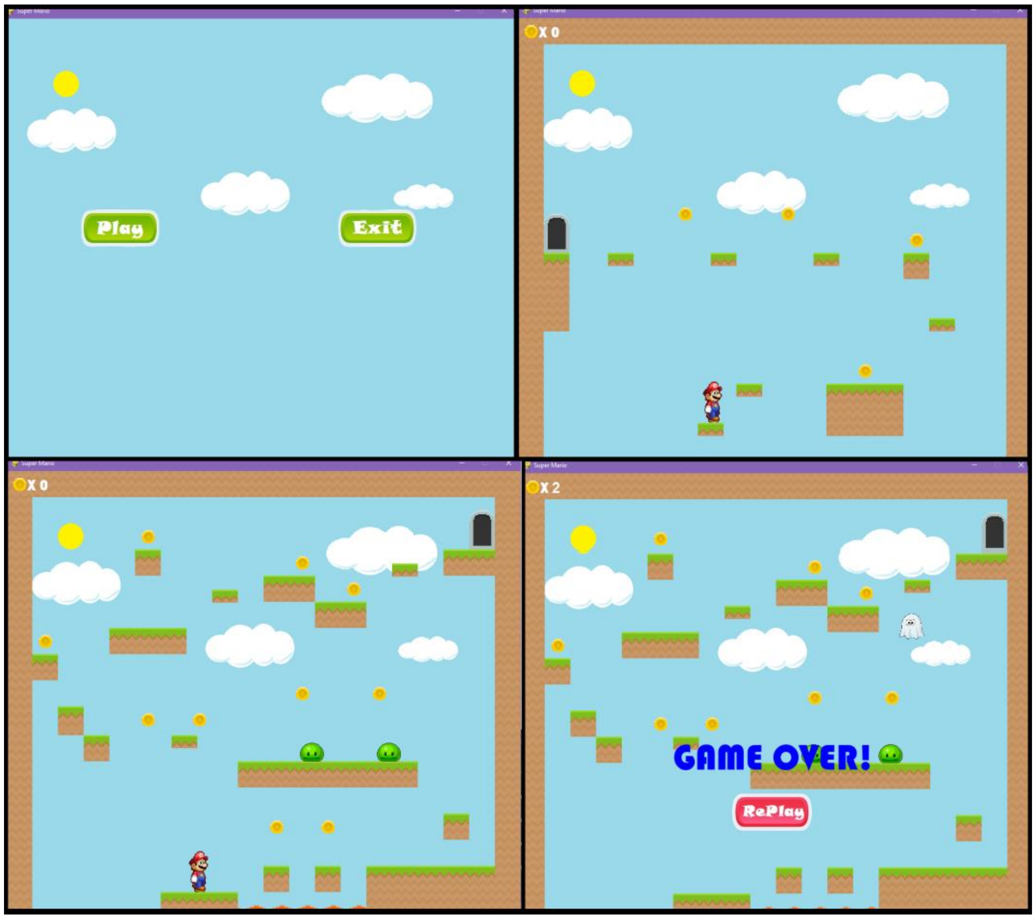
PRESS SPACE TO PLAY AGAIN

Project no. 4: “SUPER MARIO” Game

Super Mario is a popular video game franchise. In this game, the player typically controls Mario as he runs, jumps, and fights his way through levels filled with enemies and obstacles. Along the way, the player collects gold coins as he progresses through the game.

- Start simple: It's easy to get carried away with ambitious ideas when developing a game, but it's important to start with a simple prototype and build from there. For example, you could start by creating a basic platformer with a single level and one or two enemies. To implement this game, you need to do the following:
- Use object-oriented programming: Object-oriented programming (OOP) is a powerful paradigm for game development; it allows to create reusable and modular code. Use classes to represent game objects such as Mario, enemies, and power-ups.
- Use Pygame's built-in functionality: Pygame provides a lot of useful functionality out of the box, such as sprite groups, collision detection, and event handling. Take advantage of these features to simplify your code and speed up development.
- Make the game more user-friendly by adding start, exit, and replay buttons. The "start" button should load the first level of the game, the "exit" button should close the game window, and the "replay" button should reset the game to the first level.
- Create a class for the gold coins: Use OOP to create a class for the gold coins. This class should include attributes such as the coin image, the position of the coin on the screen, and a flag to indicate whether the coin has been collected.
- Place the coins in the level: Choose locations in the level to place the gold coins. You can place them in easy-to-reach locations or in hidden areas that require the player to use their platforming skills to reach them.
- Detect collisions with Mario: Use Pygame's collision detection functionality to detect when Mario collides with a gold coin. When this happens, update the coin's "collected" flag and increment the player's score.
- Display score: Create a score counter and display it on the screen. Update the score whenever the player collects a gold coin.
- Test early and often: Test your game frequently to make sure everything is working as expected. Try to break the game by doing things like jumping off the screen or clipping through walls. Fix any bugs you find as soon as possible.
- Add polish: Once you have the basic gameplay mechanics working, focus on adding polish to the game. This can include things like sound effects, animations, and visual effects. These details can make the game feel more immersive and engaging.

Sample Run:


Play
Exit
X 0
X 0
X 2
GAME OVER!
RePlay