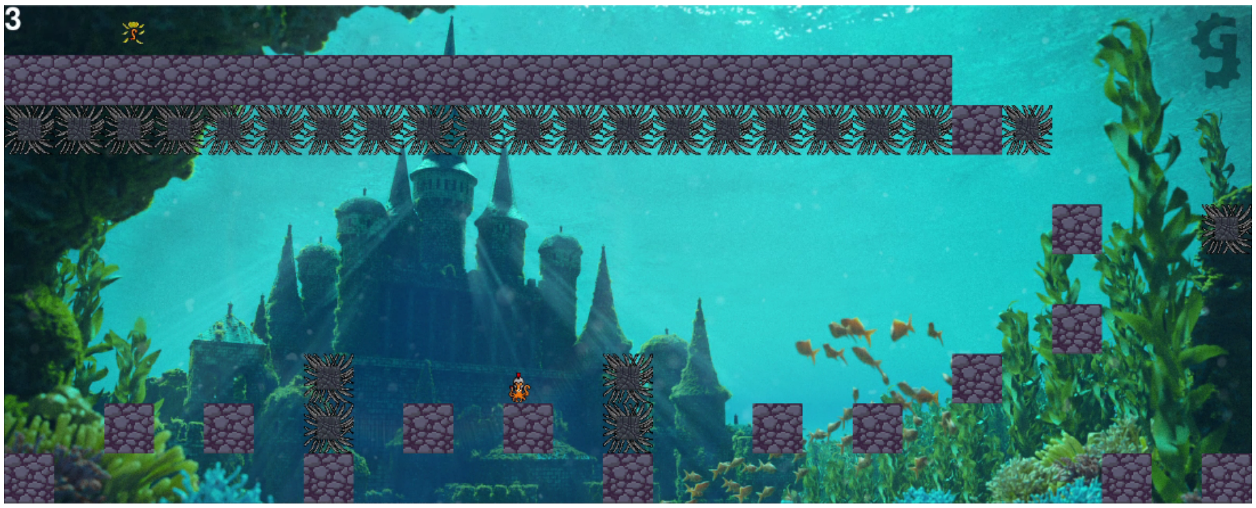
Part 1 - Introduction
Squid Knight
Game Demo: https://youtu.be/rx5uhD_c40M
GitHub: https://github.com/DonNamTran/Squid-Knight



How to run the game:
1. Download **all** the files from the GitHub page.
2. Ensure the file structure is the same as on the GitHub page.
3. Ensure you have the correct versions of python and pygame (more information down below)
4. There are multiple ways to run the program:
    a. Through any IDE that can run python, simply run the "gameLoop.py" file.
    b. Run the "./gameLoop.py" command through the terminal while inside the directory with all the python files.

Necessary software:
- Python version 3.11.0
- Pygame version 2.3.0
- Should be able to run on any OS that has both python and pygame installed.

Controls:
A - strafes left, can be used midair
D - strafes right, can be used midair
Space - jumps:
- can be held down for less than half a second for a small jump
- can be held down for over half of a second and less than 2 seconds for a medium jump
- can be held down for over 2 seconds for a large jump

Part 2 - Game Design

Mechanics/Technology

The core gameplay loop of our game is jumping up the screen to progress the level, and if the player misses their jumps then they will fall down and lose progress (similar to Jump Knight). The core mechanics of our game are jumping, falling, and the lives, the player needs to jump to progress through the zones and will fall if they miss their jumps. The player can also die if they touch enough obstacles and lose all their lives. Our game's main gimmick is the jumping ability, and the powerups. These are the main ways that the player will progress through the game. The player can charge their jump to go higher. The main way our game is different from others games like Jump Knight are the powerups that we implemented. We have the ghost powerup that allows the player to phase through the bottom of platforms, the jump boost powerup that allows the player to jump higher, and then an extra life powerup.

Story

The game doesn't really have any *explicit* story, but there is one if the player wants to look into it. The main character, Squid Knight, got one of his tentacles taken from the main antagonist, Captain Baha, and must progress through the levels to retrieve his stolen tentacle.

Player Experience

This game is designed to be pretty frustrating. The player can easily lose large chunks of progress if they miss a jump. The challenges our players face will be falling down and losing their progress, as well as dodging the spike tiles that will cause them to lose lives and eventually lose the game. The main "reward" the player receives as they progress is simply more lives, and a sense of pride and accomplishment. In terms of feedback, there will be sound effects and animations for when the player touches power ups, jumps, takes damage etc. Audio elements the player will get are all the sound effects that happen when they take damage, touch power ups, jump etc. Visually, the player gets updated whenever they take damage and receive a power up.

Part 3 - Game Design Changes

Originally, we wanted to create a platformer because that is a genre that we all enjoy playing. The original concept we had for the game was very similar to Jump Knight, just with a squid main character. We had originally planned to have features like a level select, and enemies, but neither of these made the final cut as they were too difficult to implement. We did however add a checkpoint/save-state system as a compromise. Our original plan for the game mechanics was just to jump, fall, and damage from certain obstacles. The change we made to our game mechanics was moving from a damage/hp system to a pure lives system as this would increase the difficulty for the player. Everything else ended up the same we wanted. For our game's gimmick, we wanted a jump/charge jump, and the constant threat of falling. We decided to add power ups to our game to help differentiate it from other games in the genre as well as make the gameplay more interactive.

Part 4 - Game Development & Documentation

Outline

Ex:File

- **Classes (variables)**
  - **Methods**

- **model**
  - **Model (viewObject)**
    - **setViewObject()**
  - **Tile (image, rect, img)**
    - **update()**
  - **TileDanger (image, rect)**
    - **update()**
  - **JumpPowerUp (TIME, image, img, rect)**
    - **scroll()**
  - **GhostPowerUp (TIME,image, img, rect)**
    - **scroll()**
  - **LifePowerUp (TIME, image, img, rect)**
    - **scroll()**
  - **WinPowerUp (image, img, rect)**
    - **scroll()**
  - **Level (currLevel, actualPlayer, surface, world_shift, beforeSwitch_Y, powerupNum)**
    - **setup_level()**
    - **side_collision()**
    - **top_collision()**
    - **powerUp_collision()**
    - **scroll_y()**
    - **checkSave()**
    - **run()**
    - *Saves to file and reads from file to determine your progress in the game*
  - **Player (image, img, rect, lives, direction, horizontalSpeed, gravity, jump_speed, jumping, num, space_start_time, space_hold_time, ghosting, hurt)**
    - **physics()**
    - **changePos()**
    - **jump()**
    - **update()**
- **view**
  - **View (level_map, file1, saveState, tileSize, screen_w, screen_h, screen, screenImg, screenColor, clock, titleImg, text)**
    - **updateScreen()**
    - **lose()**

        - **win()**
        - *All world is rendered from the beginning, just level scrolling is how the map updates*
- **eventController**
    - **EventController (model, view, player)**
        - **input()**
        - *Takes input from user’s keyboard*
- **gameLoop**
    - **main()**
    - **gameLoop (model, view, events, level)**

Known bugs

- The ghost power up can cause some teleporting issues when moving rapidly through the platforms and strafing at the same time.

Tools used

- VSCode Liveshare, github, discord, and google drive.

Part 5 - Group Member Roles, Tasks, and Performance

Marcus Macias-Ayala - Physics, platform collision, jump/charged jump, level scrolling, Part 4 of Game Doc
Maddux Rhodes - Level design, all the artwork, level scrolling, savestates. Part 4 and Part 1 of Game Doc
Don Tran - Powerups, sound design, platform collision, obstacle collision, lives. Part 1, Part 2, Part 3 and Part 5 of Game Doc

Final Timeline
Milestone 1: March 30 - physics done, jumping done, basic movement around levels done, basic powerup implementation, lives added

Milestone 2: April 18 - level scrolling, basic artwork, charged jump, power ups finished.

Final Submission: May 3 - Levels fully designed, all artwork done, sound effects added, game over screens, game fully done.