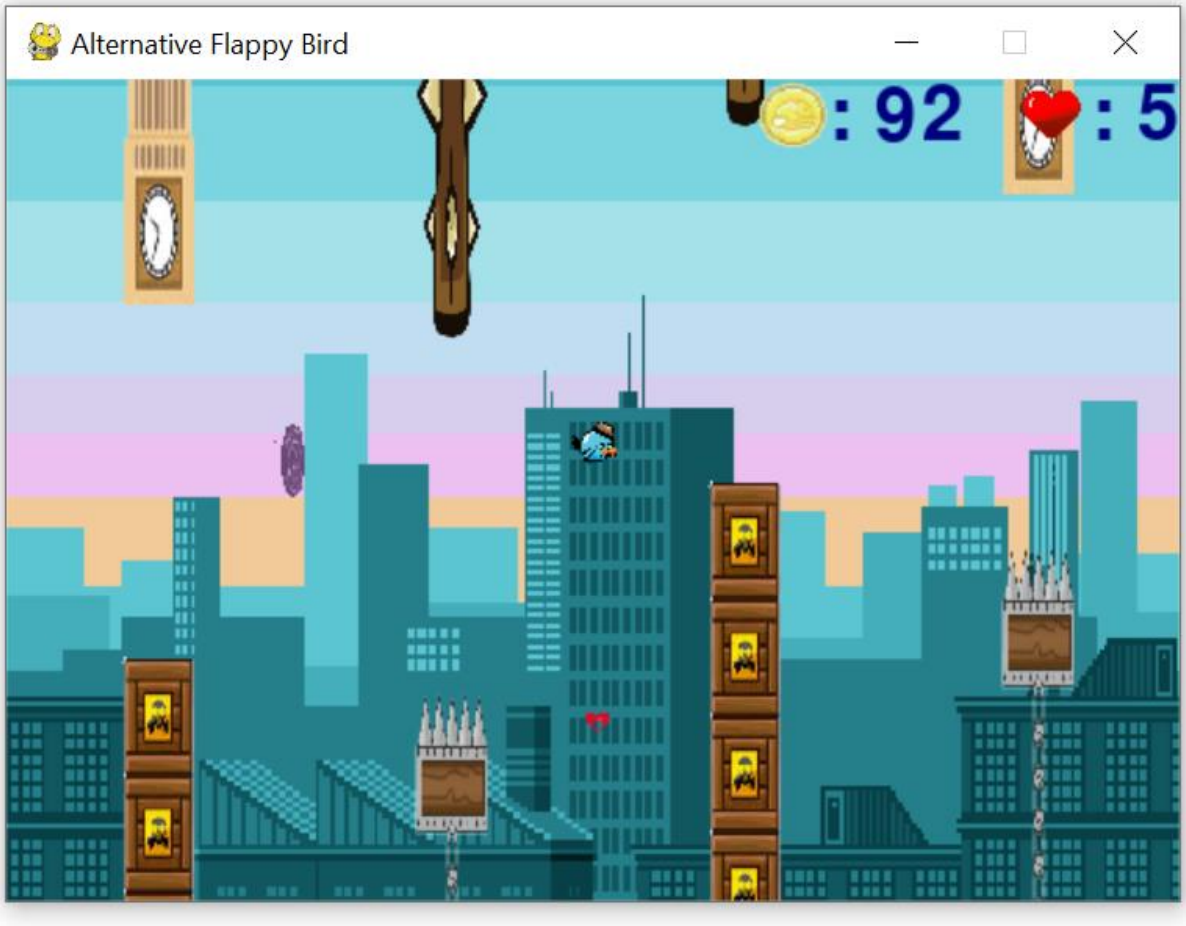
Game Document

Introduction

The name of the game is Alternative Flappy Bird.



To to download, run, and play the game, navigate to a folder and clone the game repository using `git clone https://github.com/xiaolong-tian-19/final_game_project.git`. Navigate inside the game repository, open a terminal, and type `python flappy_bird.py` to run and play the game.

Software version:

- Python: 3.8.10
- Pygame: 2.1.2

Game Control:

- Space bar : start the game & flap wings of the bird

Link to Source Code: https://github.com/xiaolong-tian-19/final_game_project

Link to Demo Video:
https://clemson.zoom.us/rec/share/s7b_I6WkkbzeDXxS5Qr7m469aKdo0Ush0TYmMbAQFgsepJZM9-A3GPxbK-RTh-sv.48wvveIMSqFXsvhg?startTime=1682994588000

Game Design

Mechanics/Technology

- The game play loop of my game has two layers. In the first layer, the game loop initiates a start screen that displays the current and best scores the player has achieved. If the player hits the space bar, the game loop in the second layer starts. This game loop initially creates the bird that the player can control, obstacles for the bird to dodge, powerups the player can collect, etc. In each iteration, this game loop checks to see if there are collisions, creates more obstacles, and so on. It updates the game instances, draw them on the screen, and goes back to the begining of the loop. Once the flappy bird uses up all of its lives, the game loop terminates and return the current score to the game loop in the first layer, which will displays the current and best score. The game loop in the first layer will then wait for the player either to close the game or to hit enter to play again.
- The core mechanics of the game is the flappy bird accelerating downwards until the player flap the bird for it to go up, obstacles that are placed at random heights for the flappy bird to dodge, lives that allow flappy bird to hit obstacles but still stay alive, and portals that will transport the flappy bird forward. The game loop checks to see if the player hits the space bar and updates the flappy bird's velocity and acceleration accordingly, In addition, the game loop checks to see if any obstacle or powerup is leaving the screen. If there is any, the game loop removes them and creates new obstacles at random height with different types (pipes, tree truncks, big ben, etc.) and new powerups at random positions. This allow a possibly infinite gameplay. Furthermore, the game loop checks to see if the flappy bird collides with any powerups, such as lives or portals. If the flappy bird collides with lives, it will add to the number of lives of the bird. The number of lives is substracted if the flappy bird hits an obstacle, and the flappy bird is recevied an period of immune time. If the flappy bird collides with a portal, the game loop moves everything else backward to create the illusion that the flappy bird is transported to the next portal.
- The game's gimmick is the additional lives flappy bird can gain, portals that will transport the flappy bird to another portal, and a variety of alternative obstacles such as tree truncks, big ben, spikes that the flappy bird must dodge. They contribute to the game by making the game a very different experience than the traditional flappy bird game. The alternative obstacles make the game play more visually appealing and add dynamic changes to the game instead of monotonic green pipes. The lives that the flappy bird can gain make the game slightly easier than the classic flappy bird game. The player could collect these additional lives and avoid fustration of making accidental mistakes. The portals made the game exciting and somewhat intense to play. On the one hand, it creates a way for the player to

supercharge if they are skillful. On the other hand, the portals create a atmosphere of uncertainty because the player may not be paying attention as to where they would land when they would enter the portals. The sudden change makes the game very enjoyable.

- The game differs from other game in the genre mainly on its gimmick: the additional lives flappy bird can gain, portals that will transport the flappy bird to another portal, and a variety of alternative obstacles such as tree truncks, big ben, spikes that the flappy bird must dodge. They make playing the game a different experience.
- The game engine is more based on pygame itself. However, pygame does a good job as a simple game engine. It allows easy creation of sprites from custom images. Furthermore, the game implementation heavily takes advantage of the pygame sprite group, particularly grouping of the obstacles and powerups. It allows the game loop to easily keep track of the instances and updates them collectively.

Story

- The story of the game is quite simple. The flappy bird is a law-abiding citizen of the city, and is atempting to dodge different and scary obstacles in the new dytopian society, which at the time experiencing magical effects such as gaining additiona lives and being able to fly through portals. The goal of the flappy bird is to dodge as many obstacles and stay alive for as long as it can.

Player Experience

- I want the player to experience thrill, uncertainty, and joy when playing my game.
- The player will face challenges including obstacles that the flappy bird must dodge and uncertainty regarding where they would land after they enter portals. The player could overcome obstacles by judging where the obstacles are and skillfully controlling when to flap the bird to plan the most optimal path. In addition, the player could collect additional lives that will give them second chances when they unfortunately hits the obstacles. To overcome the uncertainty of where the player would land after enterring portals, the player could either try to avoid portals or have quick reflexes when exiting the portals to react to the new changes.
- The player wil receive scores based on the number of obstacles it has passed. The player would be rewarded with either a bronze, silver, gold, or platium medal bassed on the best score. During the game, the player will also receive additional lives.
- There are a few instances where player will receive feedback to help understanding the state of the game. For example, when the flappy bird collides with an obstacle but have more than one lives, it will be given a short period of immune time. The feedback is visible through an angle ring on the top of the flappy bird to indicate the immunability. After the immune time is exhausted, the angle ring will disappear to indicate that. Another feedback is the lives the flappy bird could collect. After the flappy bird collects a life, the heart representing the life will disappear to reflect it.
- There are not audio elements. However, there are many visual elements to enhance the player's experience. For example, the flappy bird is constantly flapping its wings.

The hearts are appearing and disappearing to give it a more realistic feel. Same goes for the portals which are continuously rotating. The background animation also adds more visual flavor to the game. It represents the city scene in which the flappy bird is flying and have parrallax scrolling effects so that the player could judge the distance of the buildings.

Game Design Changes

- The orignial design and concept was to have an alternative flappy bird game. The story line of the game would be kept the same. As in the original flappy bird game, the player will control the flappy bird and attempt to fly through as many pipe obstacles as possible. Similarly, each successful pass will award the player one point. When the game is over, the player will be presented with different medals for their accomplishment. The 2D game would introduce a more accessible version compared to the original flappy bird game which is known to be difficult and quite additive. New game mechanics and gimmicks were to be introduced to the game to give a different player experience than the original flappy bird game.
- The overall conceptual design did not change over time. The final game design still aims to deliver an alternative flappy bird game that keeps the same story line as in the orignal flappy bird game. The major differences lie in the game mechanics and the gimmicks that were planned for the game. A few of the planned game mechanics and gimmicks were removed, and other new ones were added to the final game design. The game design changes over time because after implementing some of the original game mechanics, I realized that their introduction did not make the game playable. Furthermore, some other game mechanics proved to be quite difficult to implement, particularly related to their animations.
- The original plan for the game mechanics involves player using space bar to flap the bird in order to control the motion of the flappy bird. In addition, the flappy bird may collect additional lives and fly through portals that will transport them forward. Another orignal game mechanic is the dynamite that help the flappy bird to destroy the next pipe obstacle. The game mechanic related to the flappy bird motion didn't change. The player still uses space bar for the flappy bird to flap its wings and accelerate upward. The additional lives and portals were kept in the final game design. However, dynamites that remove the next obstacle was dropped from the final game design. The reason dynamites were removed was because the animation was difficult to implement, particularly destruction of the obstacles because various types of obstacles were used in the final game design. Finding proper animation for those different images representing obstacles were difficult. Futhermore, getting a realisitc feeling for the obstacles destruction was not easy either. Therefore, the game mechanic related to dynamite was dropped, but other original game mechanics were kept.
- Orginally, the game gimmicks were to be introduced to the game includes lives that the flappy bird could acquire, portals that transport the flappy bird to different portals, dynamites that will destroy the next pipe obstacle(s). A day and night mode for the flappy bird game to induce a realistic feeling for the players was also planned. When the player is controlling the flappy bird, the screen will switch from

day (light) to night (dark) and vice versa after a certain amount of time. This will also make the game more exciting to play because of the challenge with getting adjusted to the transition. These orignal game gimmicks changes in a couple of ways. First, the gimmick involving dynamite that will destroy the next pipe obstacle was removed. The day and night mode that was originally planned was not incoporated into the final game design. A few new different game mechanics and visual effects were added to the design. This includes a parrallax scrolling background, different types of obstacles besides just the green pipes, animation of the flappy bird as well as the lives and the portals. There are a few reasons for which these original game gimmicks were removed and new ones are added to the final game design. For instance, the game mechanic about dynamites that could destroy the obstacles were removed because of difficulties related to obstacle destruction animation. The light and dark mode was partly implemented. However, finding the right images to allow a playable and visually appealing game that transition between light and dark mode proved to be very difficult. The new game gimmicks came mostly by experimentation. For example, I tried to replace some of the pipes with flamming torches. However, the right animation that can be placed at random height without distorting the images were not easy to implment. In addition, obstacle animation such as the flamming torches creates problems with regard to collision with the flappy bird as they changes every frame. Furthermore experimentation using various static, but obstacles other than green pipes showed that these different types of obstacles greatly enhance the player experience. By trying out the game, I also realized that some of the game gimmick performed in unexpected ways. For example, the portals that the flappy bird would fly through orignally intends to help the flappy bird. However, after implementation, I realize that it does more than that by introducing uncertainty into the game and makes the game more exciting to play. That's why it was kept.

Game Development & Documentation

- Receiving input from the player is quite simple for the game. The only key that the player uses to in order to play the game is the space bar. The player uses the space bar to start the flappy bird game, and during the game, the player hits the space bar for the flappy bird to flap its wings. Therefore, the game implementation does not use a separate controller for handling user input. Receiving user input and update the game state accordingly is done in the game loop. In each iteration of the game loop, it checks all the events in the event queue. If the event is a key down and the event key is space, it will call the bird.flap() for the bird to flap its wings and add an upward acceleration.
- The state of the game are stored in various objects, specifically pygame sprites, representing entities in the game. For example, the position, velocity, acceleration, animation frame, number of lives, immune status, immune time, and image of the flappy bird are stored in an instance of the sprite class called Bird. Many other models exist in the game, such as sprite classes for obstacles (LowerPipe & UpperPipe), hearts (Heart), portals (Portal), score (Score), background

(Background), etc. Instances of these models are instantiated whenever appropriate, for example, when a new pipe is created and put into the screen. They are deleted and removed as well, such as when the heart is collected by the flappy bird, the instance object representing the heart is removed. Objects of the same type are often group together using the pygame sprite Group to allow simple handling. The state of the game is updated in each iteration of the game loop through the update method of the instances. In the game loop, specific methods are often called whenever necessary to update the state of the game. For example, when the flappy bird collides with an object, one will be substracted from the total number of lives and the immune status of the flappy bird will start. Updating the state of the instances allow the game state to be in sync with events that are happening.

- The screen is updated through the draw() method of the pygame sprite group. Sprite class instances that are models of entities in the game are grouped together based on their type, for example, obstacles (pipe_group), hearts (heart_group), portals (portal_group), etc. At the end of each iteration of the game loop, draw() is called for each of the group in corresponding order. Some group must be drawn after others. This could be portal_group drawn after background, or score_group drawn after pipe_group. Before sprites are drawn, their states and values are updated. This allow the animation frame, and the image of the sprites to be changed before they are drawn, and therefore enables animation of entities such as bird, portals, etc.
- I am not aware of any major bugs or flaws. One thing that presents a nuisance is the score and number of lives displayed on the top right. As the player advances in the game and earns higher score and more lives, the score and number of lives increase and thus have more digits. It could be possible that the text will overun the symbols representing score (a medal) and life (a heart). It was not critial to the functionality of the game, so I did not prioritize this, and I couldn't find enough time to figure out how to do it. One possible way to solve the issue is to move the symbols (medal and heart) based on the number of digits of the score/number of lives or to scale the text font of the score/number of lives so that they will always remain about the same width. One thing that I was not also satisfied about is the transition from the start screen to the main game and vice versa. I think that the transition could be made more natural and smoother.
- I did not have to colloborate, but the main tool I used for coding is VSCode. The coder versioning is done through Git and GitHub. It helps me keeps tracks of what changes has been made and allow me to recover files if I accidentally delete them. I used a variety of other tools to help with making the 2D game as well, particularly with respect to creating the graphic images I want. I heavily used an online website to remove background from images. The same website also allows me to break gif into frames and split sprite sheet into individual images. I also use Adobe products such as Photoshop and InDesign to help with making a few custom images and crop the images so that their size roughly corresponds to their size with background removed.

Group Member Roles, Tasks, and Performance

- I am the only member, so I am responsible for all the work.
- Milestone 1: March 30
  - Create the background of the game - March 16
  - Create pipe obstacles - March 20
  - Create side scrolling of the game - March 29
  - Create all classes represent objects in the game - March 29
  - Create view, controller, and game loop - March 29
  - Updated game document - March 30
- Milestone 2: April 18
  - Add animation to the objects in the game – April 12
  - Adjust the game parameters to make the game more playable – April 16
  - Updated game document – April 16
- Final Game Submission and Presentation
  - Allow the player to play as many times as desired - April 25
  - Add different type of obstacles to the game - April 28
  - Add a parrallax srolling backgroup to the game - April 28
  - Add a start screen for the game - April 29
  - Completed and polished game - May 3
  - Completed game document - May 3
  - Make 30+ seconds demo video - May 3
  - Present your game during the final period for this class - May 4