Penguin Game

Group 7

Member Name	Percentage of Contributions	Justifications (Primary Responsibility)
Cai Mengran	20%	report & flow charts & presentation slides
Dai Luqiao	20%	report & coding & presentation slides
Ding Yi	20%	report & flow charts & presentation slides
Hu Xuanzheng	20%	report & coding & presentation slides
Jonathan Then	20%	report & coding & presentation slides

* All members have presented equal contributions to this project.

Table of Contents

1 Algorithm Design

1.1 Top-level Flow Chart

The top-level menu flow chart in *Figure 1* shows the entire process of the penguin game. The menu allows users to select game modes, which are the flying penguin game and the jumping penguin game. The corresponding flowcharts are shown in the Appendices, and more details about these two scenarios will be discussed in the next section.

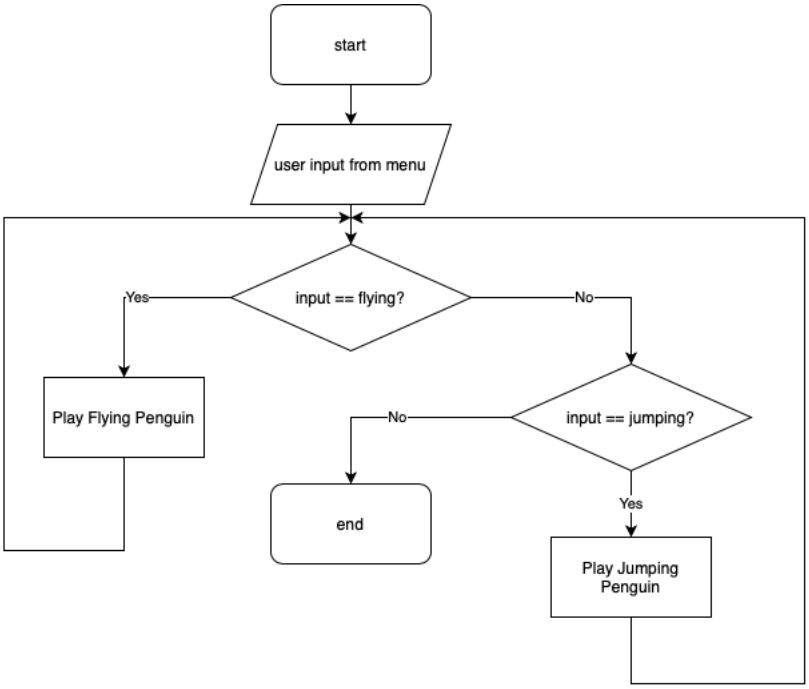


Figure 1: menu flowchart

1.2 User-defined Functions

pygame, pygame_menu, sys, os, random, and time modules were used to implement the penguin game. The `'menu.py'` file implemented a menu interface, which provided users with visually displayed choices and a structured view of the game. When users selected `'Flying Penguin'` mode, `'flyPenguinGame.py'` would run. By contrast, if users selected the other mode, it allowed users to play a penguin game with jumping status. The game could be terminated by a `'Quit'` or a close button. Game instruction is shown bellows.

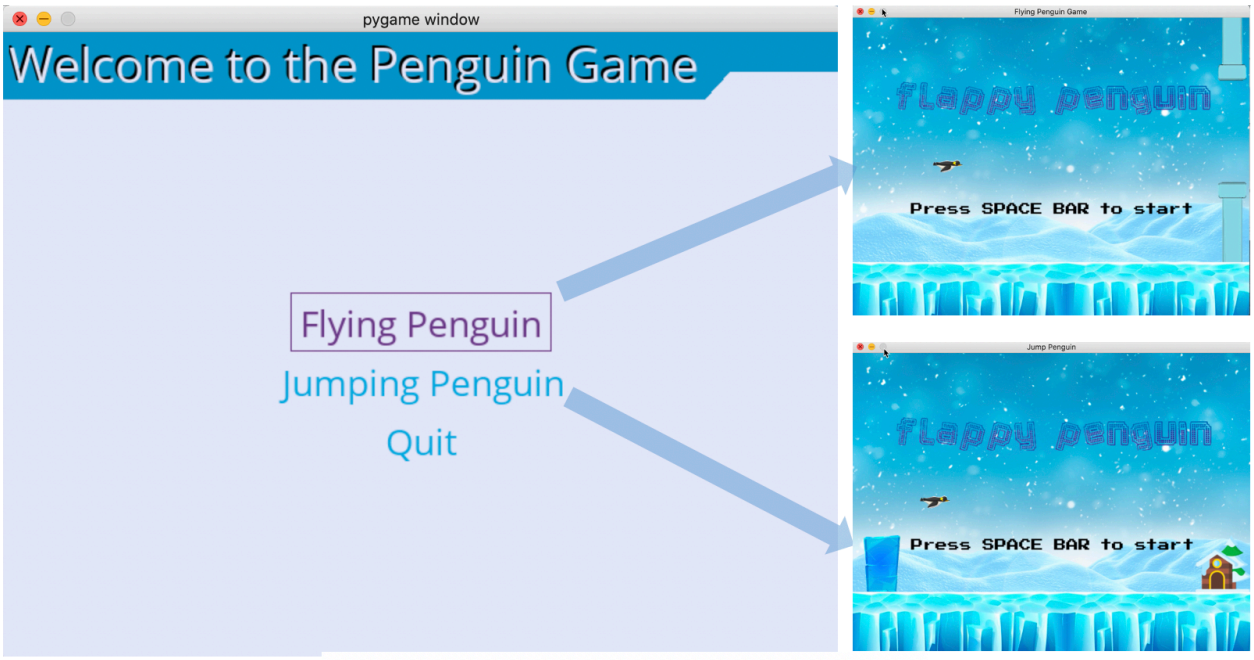


Figure 2: game instruction

Key events capture, penguin movement, obstacles movements, collision decisions and immunity, score and health points calculation functions were implemented in both modes. The key events handler function was used to capture users' actions interacting with a game. Users could start to play a game by pressing a space key and exit the game through an escape key or a close button.

The implementation of penguin movement and obstacles were different between flying and jumping modes. In the flying mode, an initialized velocity along the y-axis was defined as -9, and the minus symbol represented the penguin moving upwards and 9 was the penguin's initial speed. When flying, without pressing a space key, the penguin would fall due to the gravity simulation. By contrast, when users pressed the required key, velocity values would change from -9 to -8, which made the penguin move upwards. `'createPipe()'` function was used to randomly generate the upper and down pipes. Initially, it consisted of two upper pipes named `'up_pipes'` and two down pipes called `'down_pipes'`. After creating, pipes moved to the left by velocity named `'pipeVelX'` with a value of -4. Similarly, a negative sign meant a pipe moved leftwards. It can be seen in the following coordinate system when a pipe moved to the left, its corresponding

x-axis value decreased. The penguin in both modes remained unchanged horizontally, but the movements of obstacles made it look like flying or jumping. Design logic was the same, but in the jumping mode, the penguin's initial velocity was 0, so that the penguin could glide across the ice.

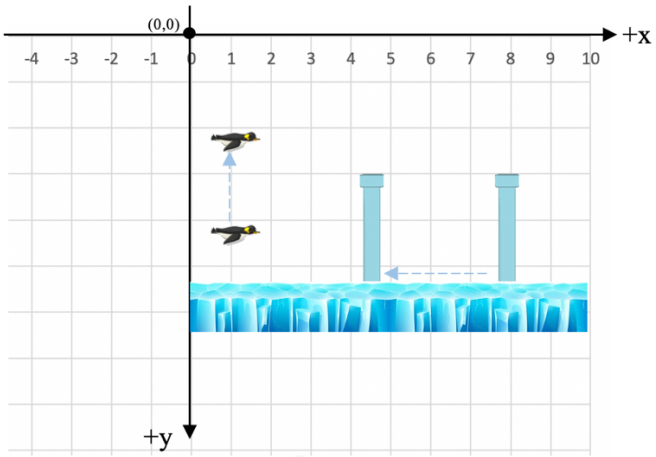


Figure 3: coordinate system used in a game

Regards to collision decisions and immunity functions, there were four different collision types in the flying mode but only one collision case in the jumping mode, which was illustrated in *Figures 4* and *5*. Collision immunity was used to protect users. If the time difference between the last collision time and the current time spot was larger than the predetermined value, its protection would terminate and the `'collision_immune'` status would change from True to False. Every time the penguin occurred a collision, the collision immunity function would be activated, and health points would reduce by 1. As for the score calculation function, every time the penguin was over the obstacles, the score would increase by 1. Otherwise, the score remained unchanged. Once `'collision_immune'` equaled True and health points equaled 0, the game was over. The `'game_over(final_score)'` function was used to display users' final score, and then users could press a space keyboard to restart.

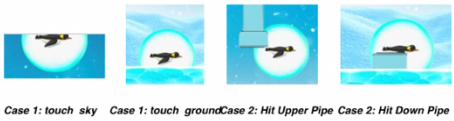


Figure 4: collision cases in flying penguin

Figure 5: collision case in jumping penguin

2 Program Testing

2.1 Invalid Keyboard Handler

When users pressed a key except for a space keyboard, the `'handleInvalidKey()'` function would be run to remind users. The 'Invalid Keyboard' (shown as bellows) would be displayed for 1 second through a `'time.sleep(1)'` function.

Figure 6: invalid key for flying mode

Figure 7: invalid key for jumping mode

2.2 Real-time Programming Testing

Since a game was developed in this project, most of the programming tests were real-time tests. It meant that the robustness of the project and whether it behaved as expected were based on the real-time running results. To illustrate, when the penguin collided, a circle of lights would surround it (saw as bellows).

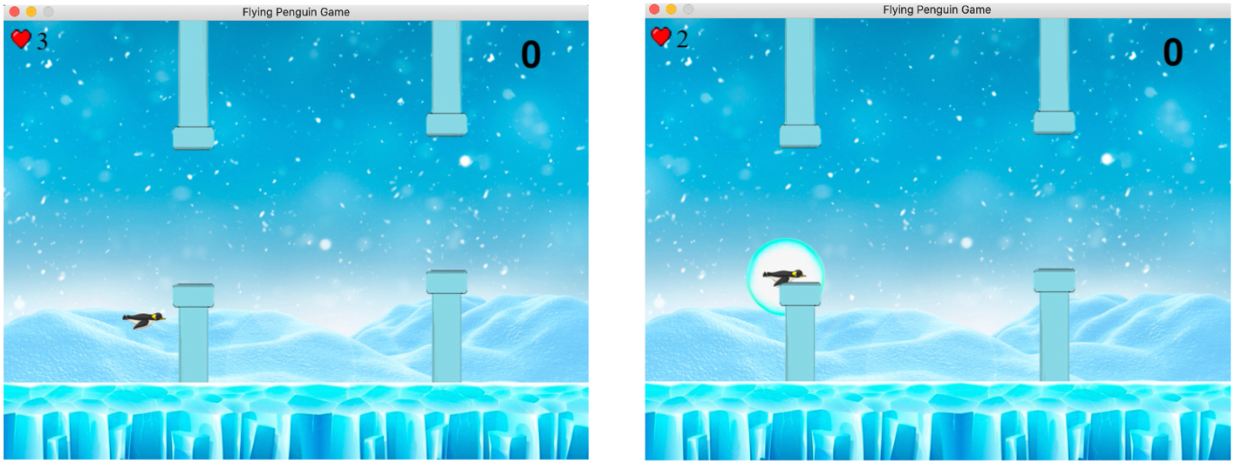


Figure 8: collision

3 Reflection

3.1 Difficulties Encountered and Solutions

There were two primary difficulties. One was from image selection, and the other one was from algorithm designs.

In the User Interface (UI) design, it was difficult to directly find perfect images that could be used in the game from the Internet. Hence, some photo-retouching software like Beautify application was used to modify images' color and sizes. However, for some images, the image effect would become worse after modifying the size that the game needed. The following pictures demonstrated this case. The original size of the pillar was 296x303mm with a resolution of 28 pixels/cm, and the size that a game needed was 11x68mm with a resolution of 47 pixels/cm. After modifying the size, the pillar became thinner and taller. Under this circumstance, one of the methods was to find another image. The other solution was to change the original pipe image color.

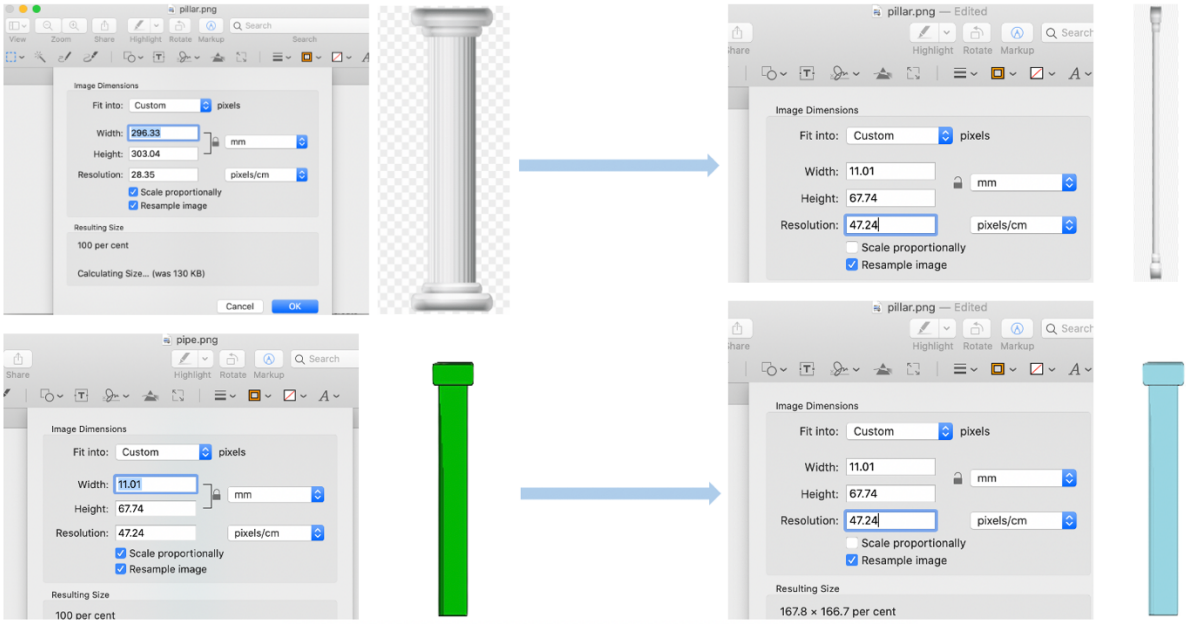


Figure 9: images modifications

The other difficulty was designing algorithms including pipe constructions with random heights, collusion decisions, and score calculation algorithms. The way to solve this problem was to spend more time understanding the underlying algorithms behind the

game through some website sources like YouTube and Bilibili websites. Sometimes, the function of a statement or function may not be understood. At that time, one of the methods was to change its values, observe the running results, and try many times until having a good understanding of that.

3.2 Knowledge Learnt from This Course

It had been a fruitful journey to take this course. Basic Python programming syntax including for/while loop, how to design a flowchart, how to design an algorithm, and how to solve programming problems by self-learning were learnt from the course. Most importantly, how to work on programming projects and how to collaborate with others were learnt through the mini project. The major achievement in this course was developing a penguin game. All group members within the project team would like to extend our sincere appreciation to professor Dr. Zhang, the TA Dr. Chen, and all other classmates that help with mutual peer learning.

3.3 Further Improvement Suggestions

Further improvements stem from the collision design, the UI design, the speed acceleration function, and the user experience.

The collision design can be optimized by team self-created images of penguins and obstacles. In the current collusion design, sometimes the penguin does not look like having a collision with obstacles when health points decline by 1. This is because the algorithm for collisions is computed based on the width and height of the image. Each image is regarded as a rectangle, but in fact, the image the user sees is irregular like a lodge obstacle. As for the UI design, the menu interface can be more attractive with further adjustments. Concerning the speed acceleration function, speed acceleration is currently limited. There is a lot of room for improvement in acceleration algorithms. Furthermore, more penguin dynamic features can be added to improve the user experience. For instance, the penguin could flutter its wings during flying, or the penguin would roll its eyes once it hits an obstacle, or a little ghost-like penguin’s shaded color would rise from the penguin’s body once it has lost all health points.

4 References

Collision Detection in PyGame - GeeksforGeeks. (n.d.). GeeksforGeeks. https://www.geeksforgeeks.org/collision-detection-in-pygame/

Collision Detection in PyGame - GeeksforGeeks. (n.d.). GeeksforGeeks. https://www.geeksforgeeks.org/collision-detection-in-pygame/

Flappy Bird In Python Pygame With Source Code - Page 7 Of 8 - CopyAssignment. (n.d.). CopyAssignment. https://copyassignment.com/flappy-bird-in-python-pygame-with-source-code/7/

How can I make one python file run another?(n.d.). Stack Overflow. https://stackoverflow.com/questions/7974849/how-can-i-make-one-python-file-run-another

How to make Flappy Bird Game in Pygame? - GeeksforGeeks. (n.d.). GeeksforGeeks. https://www.geeksforgeeks.org/how-to-make-flappy-bird-game-in-pygame/

pygame-menu — pygame-menu 4.2.8 Documentation. (n.d.). pygame-menu — pygame-menu 4.2.8 Documentation. https://pygame-menu.readthedocs.io/en/4.2.8/

Python pygame font ffont | Learn-codes.net. (n.d.). Programming Tutorial & Code Examples. https://www.learn-codes.net/javascript/python-pygame-font-ffont/

Images

Background (Image by kjpargeter on Freepik)

Free Photo | 3d winter snowy landscape. (n.d.). Freepik. https://www.freepik.com/free-photo/3d-winter-snowy-

landscape_3640526.htm#query=ice%20background&position=5&from_view=search

Ice Floor (Image by upklyak on Freepik)

Free Vector | Game grounds with texture of ice, water and lava. (n.d.). Freepik. https://www.freepik.com/free-vector/game-grounds-with-texture-ice-water-lava_21329661.htm#query=ice%20floor&position=8&from_view=search

Penguin (Image by brgfx on Freepik)

Free Vector | Flying penguin animal cartoon sticker. (n.d.). Freepik. https://www.freepik.com/free-vector/flying-penguin-animal-cartoon-sticker_20500107.htm#page=2&query=penguin&position=0&from_view=search

Pillar Image

Download pillar clipart png photo png - Free PNG Images. (n.d.). TopPNG. https://toppng.com/pillar-PNG-free-PNG-Images_53750

Instructions

Press SPACE BAR to continue | Pixel Art Maker. (n.d.). Pixel Art Maker. http://pixelartmaker.com/art/112d2ed1d2d2316

Snow Cabin (Winter icons created by Freepik - Flaticon)

Lodge free icons designed by Freepik. (2022, September 16). Flaticon. https://www.flaticon.com/free-icon/lodge_1240067

Iceberg – Pointed (Freepik)

Free Vector | Set of different drawn icebergs. (n.d.). Freepik. https://www.freepik.com/free-vector/set-different-drawn-

icebergs_9683376.htm#query=icebergs&position=6&from_view=search

Iceberg – Flat (Image by upklyak on Freepik)

Free Vector | Ice game buttons set. (n.d.). Freepik. https://www.freepik.com/free-vector/ice-game-buttons-set_12876458.htm#page=2&query=iceberg&position=29&from_view=search

Heart

Heart Pixel Art. (n.d.). Free PNG Download - Transparent PNG Images Free Download - SUBPNG / PNGFLY. https://www.subpng.com/png-srqdy0/

Shield

Energy Shield Png - Energy Shield Transparent PNG Image | Transparent PNG Free Download on SeekPNG. (n.d.). SeekPNG.com. https://www.seekpng.com/ipng/u2q8a9i1r5r5e6a9_energy-shield-png-energy-shield-transparent/

Sounds

Bird Flap Sound

BirdFlap2.wav by AgentDD. (n.d.). Freesound. https://freesound.org/people/AgentDD/sounds/246224/

Death Sound

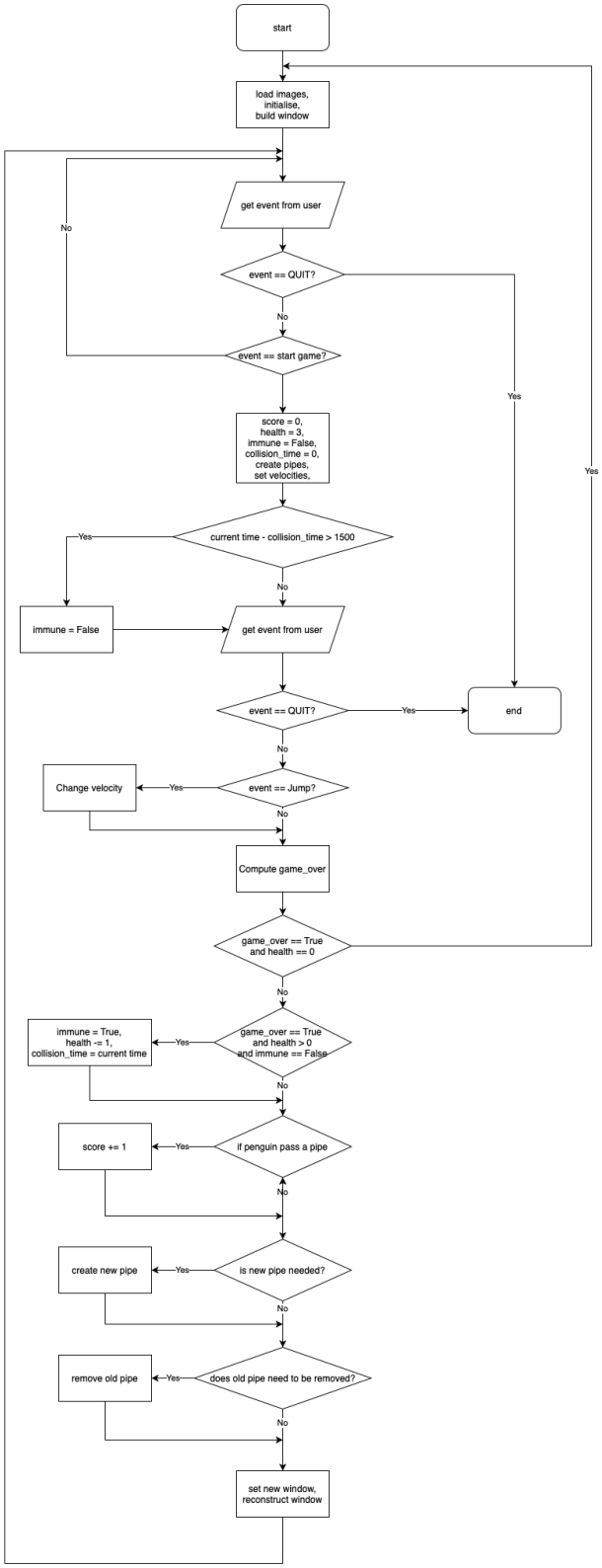
Dino Game in Python with Source Code (n.d.). https://download.code-projects.org/details/5a0d2fba-3abf-44fa-8a03-c4bd97cc3f78

Hurt Sound

Menu Tick.wav by LorenzoTheGreat. (n.d.). Freesound. https://freesound.org/people/LorenzoTheGreat/sounds/417792/

5 Appendices

5.1 Flowchart for Flying Penguin



5.2 Flowchart for Jumping Penguin