Group 59

Momchil Damianov #1965131

Ivayla Ilieva #1953028

The two topics of choice for our group are :

1. Game Design (Point I.)
2. To learn how to use GitHub for version storage of our code

   (Point II.)

https://github.com/Ivayla377/CBL_GAME.git - This is our repository

Product Backlog :

1. Player Character Movement

- Name: Implement player character movement.

- How to Demo: Start the game, control the player character using left and right arrow keys to dodge incoming faces.

- Notes: Learn to handle user input, update player character's position, and detect collisions.

2. Custom Player and Enemy Sprites

- Name: Design and integrate custom player and enemy sprites.

- How to Demo: Show the custom-designed player and enemy sprites in the game.

- Notes: Gain experience in creating and incorporating graphical assets into the game.

3. Scoring System

- Name: Implement a scoring system.

- How to Demo: Display the player's score on the screen and update it as the player successfully dodges faces.

- Notes: Learn to track and update scores in the game.

4. Enemy Spawning

- Name: Create a system for spawning enemies (faces) from the top of the screen.

- How to Demo: Show faces spawning at regular intervals and descending towards the player.

- Notes: Gain experience in spawning and managing game objects dynamically.

5. Collision Detection

- Name: Implement collision detection between the player character and faces.

- How to Demo: Demonstrate that the game registers collisions when the player character touches a face.

- Notes: Learn how to handle collisions and respond accordingly in the game.

6. Background Design

- Name: Create a custom background for the game.

- How to Demo: Show the custom background in the game's display.

- Notes: Gain experience in setting up a visually appealing game environment.

7. Enemy Movement

- Name: Add movement patterns to the descending faces.

- How to Demo: Show the faces moving smoothly from top to bottom.

- Notes: Learn to control and animate enemy movements.

8. Column-Based Spawning

- Name: Implement column-based spawning for the faces.

- How to Demo: Show faces spawning from six imaginary columns at the top.

- Notes: Gain experience in arranging enemy spawns for varied gameplay.

9. Game Over Condition

- Name: Implement a game over condition.

- How to Demo: Show that the game ends when the player character collides with a face, and display the final score.

- Notes: Learn to manage game state transitions.

10. Sound Effects

- Name: Add sound such as background sound

- How to Demo: Demonstrate audio feedback for actions like dodging, collisions, and scoring.

- Notes: Gain experience in integrating sound into the game.

11. High Score System

- Name: Implement a high score system.

- How to Demo: Show the highest score achieved in the game and persistently store high scores.

- Notes: Learn to save and retrieve high scores between game sessions.

Priority: The backlog items are prioritized from top to bottom based on their importance to the game's core functionality and user experience.

I. Game Design

- Stage development

  The game consists of three stages. Each stage is visually distinct from the rest of the stages.

  - Stage changing algorithms

    The change in stages happens according to an algorithm that accepts information from the game loop and responds to the current situation.

    - Score

      The most important (independent) variable used to determine the stage is the score. When a player reaches a certain score, the stage

changes. The last stage has an indefinite duration as the player is allowed to play until they have no more lives left.

- Player & Enemies

  The player and enemies are consequences (dependent variables) that are determined according to the current stage the player finds themselves in. The players and enemies, respectively, all share the same template to which changes are applied in accordance with the stage.

- Score keeping
  - Score criteria

    Score is updated whenever an enemy misses the player and falls through the screen. No points are awarded when an enemy hits the player. From both a mechanical and logical perspective this is a valid rule since when a hit is detected the enemy is reset before passing the bottom border of the panel and because the player has a life deducted in this instance.
  - Storing max score

    The max score of a player is stored in a .txt file named ScoreText.txt. The Panel class implements a method that extracts the score at the end of each game and stores it in the text file if it is bigger than the previous score recorded. If the player plays the game for the first time ever, then the score is automatically recorded at the end of their first try.
- Visuals
  - Player Images

    Player images are 64x64 sprites of each of the enemies' favorite food!
  - Enemy Images

Enemy images are 64x64 pixelated sprites us and one other student from the group. The background in each photo is cut out to give the impression that each enemy sprite has unique outlining. However, the hit box on each enemy is 64x64 pixels.

- Background Design

  The background design was decided on the basis of retro video games that share a similar design, and it is intended to evoke a sense of nostalgia from the player and add to the game experience.

- Miscellaneous

  This category consists of small details that are contributing to the game experience in general.

  - Score display

    The score is displayed using pixelated digits. Similar to the background design, each digit is pixelated in order to bring out the nostalgic features of 64x64 games most of us have experienced in our childhood.

    - Current

      The current score is a series of digits that extend to the hundreds. The score does not extend any further since the final stage of the game is reached at a score that is 10 times smaller than the maximum score to be displayed. Therefore, it would be bothersome for the player to even reach such a score, so we have decided to cap our score at 999.

    - Max

The max score of a player is displayed above the start "button" before the game begins and disappears after the gameplay loop begins.

- Hearts

  The hearts follow the designs used in retro 64x64 games.

- Start "button"

  The start "button" pays homage to the original Super Mario Bros. franchise and their designs and aesthetics.

- Handle Ending Screen

  The ending screen is intentionally more dramatic than what would be adequate for the level of intensity of the game in order to add a comedic side to the game experience. We suggest that the juxtaposition between the light hearted enemy dodging and the seriousness of the ending screen will get a laugh out of the player.

- Game Mechanics

  The game runs on a list of algorithms, the most important of which are listed here.

  - Key Listener

    The class Key Input implements the Key Listener interface and makes uses of the methods "button pressed" & "button released". When specific buttons are pressed and released, ques are sent to the gameplay loop to update accordingly.

  - Mouse Listener
    - Rectangle object and .contains() method
  - Game Thread

The game thread is a thread that takes the JPanel as its argument during construction. The whole JPanel implements the interface Runnable and overrides the method run().

- FPS implementation

  Our code makes use of the sleep method for the thread. In practice, our algorithm splits every second into 60 equal fractions in nano seconds. Then, the algorithm detects how long it takes to complete the update and repaint methods and goes into a sleep mode for the remaining time of that fraction of the second. When the time enters the next fraction of the second, the thread is allowed to run again. Then it sleeps again, etc. This algorithm allows us to create a basic framework for FPS. We have decided to use 60 FPS in our game since it feels smooth and does not burden the computer as much compared to higher FPS settings.

II. Git Version Control

- Name: Use GitHub to store code in a cloud

- How to Demo: Store and manage code using a reliable online platform for group projects

This is our repository: https://github.com/Ivayla377/CBL_GAME.git

- Notes: Learn to save and retrieve versions of code using GitHub and Git

- Creating Issues

- Creating and using labels

- Creating README file