CSE 1321L: Programming and Problem Solving I Lab

Assignment 5 – 100 points

PyGame

What students will learn:

1) Incorporating prior programming knowledge to game development
2) Developing a larger scale, graphical program
3) Problem solving

Overview: We will be making a basic video game using PyGame. While the result of this project should be fun, the real goal is to give you the opportunity to apply everything you've learned about programming up to this point. There are many fields of software development, but the fundamentals you've learned so far can be applied to all of them.

Assignment 5A: Meteor Evader

In this assignment, you will create a game where the player controls a spaceship and must navigate through space to avoid incoming meteors. The goal is to survive as long as possible by dodging meteors coming from all directions. The game will increase in difficulty over time, with more meteors appearing. The game ends when the player collides with a meteor.

Requirements:

- The game window must be 800 pixels wide by 800 pixels tall.
- Player Control:
    - The player controls a spaceship using the W, A, S, D keys (up, left, down, and right respectively).
    - The spaceship should be represented using a Pygame Rect (e.g., a 40x40 pixel rectangle).
    - Ensure that the player cannot move outside the game window boundaries.

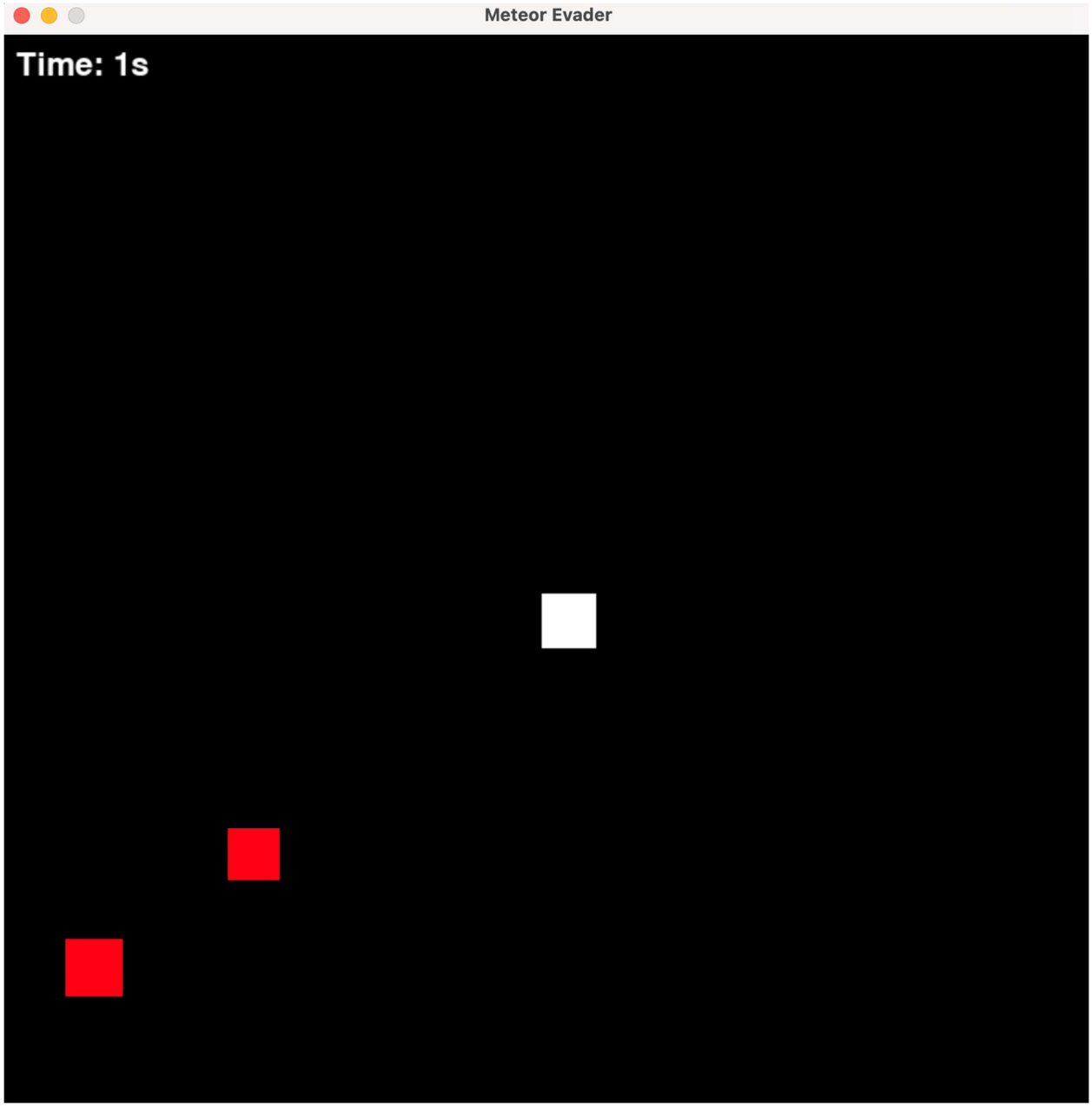
- Meteors should spawn from the edges of the screen (top, bottom, left, and right).
- A random side is chosen for each meteor's spawn location.
- Based on the side from which a meteor spawns:
    - Top: Moves downward.
    - Bottom: Moves upward.
    - Left: Moves rightward.
    - Right: Moves leftward.
- Meteor Movement:
    - Each meteor should have a horizontal (dx) and vertical (dy) speed component to control its movement direction.
    - The speed components are determined based on the side from which the meteor spawns so that it moves inward toward the center of the screen.
    - The meteors should vary in size and speed (randomly generate meteor sizes between 20x20 and 60x60 pixels and randomize their speed components).
- New meteors should spawn every 5 seconds.
- Gradually increase the spawn rate or speed of the meteors as the player survives longer.
- Timer:
    - Implement a timer that displays how long the player has survived.
    - The timer will start when the game loop begins.
    - The timer will stop when the spaceship collides with a meteor (game ends).
    - The game will display the final survival time after a collision and reset the game.
- The longer the player survives, the more meteors should spawn, making the game progressively harder.
- Use Pygame's colliderect() method to detect collisions between the player's spaceship and meteors.
- If the spaceship collides with any meteor, the game should end.
- Implement a scoreboard to display the player's longest survival time.
- The longest survival time should be stored and displayed even after the player loses.

- When the player’s spaceship collides with a meteor, play a crash-like or destruction-like sound.
- Play a sound effect every time a new meteor spawns to alert the player.
- Ensure the sound files are loaded using pygame.mixer.Sound() and play the sounds at the appropriate events during gameplay.
- Gradually speed up or slow down the meteors over time based on how long the player survives.
- Make meteors visually distinct (e.g., different colors or random shapes).
- Optional: Add background music to play throughout the game using pygame.mixer.music.

Hints:

- Use pygame.Rect to represent both the spaceship and meteors for easy collision detection.
- Randomize the spawning side and speed of each meteor using random.choice() and random.randint().
- Use pygame.time.get_ticks() or pygame.time.Clock() to manage meteor spawn timings and track the player's survival time.
- Adjust meteor direction and speed using the dx and dy values based on where the meteor spawns.
- Use pygame.font.Font() to display the timer and score on the screen.
- To load and play sound effects, use pygame.mixer.Sound('filename.mp3').play().
- For background music, use pygame.mixer.music.load() and pygame.mixer.music.play(-1) to loop it continuously.

SAMPLE OUTPUT:



Submission:

1. You will submit all required files to run your game.

2. File names must be correct.

3. Upload all files (simultaneously) to the assignment submission folder in Gradescope.