Ryan Hatch  
CS-330  
Glitter Bomb Explanation and Reasoning for Code Edits  
August 24, 2024

**Glitter Bomb Game and Explanation of the Codes Changes:**In this project, I developed a 2D animation game where the player controls a paddle to launch `glitter bombs` at destructible bricks.

The main goal was to create a fun and engaging experience while meeting the assignment's requirements. Here I have outlined the key changes and the decisions that I made to be able to bring this game to life and add a little flavor to its engagement.

**Brick Arrangement and Interaction:**

I designed a grid of bricks using the `createBrickWall` function, arranging them into multiple rows and columns. Each brick got a randomly assigned color and was set to be either reflective or destructible.

When a glitter bomb hits a brick, the brick's state changes. For destructible bricks, the color shifts with each hit, and after a certain number of hits, the brick disappears.

**Glitter Bomb Physics:**

I implemented physics-based movement for the glitter bombs. They launch from the paddle and travel in a direction that the player controls using the arrow keys, letting players strategically aim to hit the bricks.

When the bombs hit the edges of the screen or collide with bricks, they change direction. Additionally, their speed increases slightly with each collision, gradually making the game more challenging.

**Collision Detection and Response:**

The collision detection system tracks interactions between glitter bombs and bricks, as well as between multiple bombs. When a bomb hits a brick, its direction changes randomly, and the brick's state is updated—whether it's a color change or complete destruction.

When bombs collide with each other, they change color, creating a dynamic visual effect that highlights the interaction.

**Player-Controlled Paddle:**

A manually controlled paddle is positioned at the bottom of the screen. Players can move the paddle left or right using the arrow keys and launch glitter bombs by pressing the spacebar.

The paddle acts as the launching point for the bombs, making it the main tool for interacting with and targeting the bricks on the screen.

**Logical Flow and Modularity:**

* The code is organized into classes like `Brick`, `Circle`, and `Paddle`, each encapsulating the behavior and properties of the respective game elements. This modular design makes the code easier to understand, maintain, and expand upon.
* The main function manages the game loop, ensuring that all elements are updated and rendered correctly with each frame. It handles input, movement, and collision detection, following a logical flow from player input to game actions and visual updates.

**Coding Best Practices:**

* The code is formatted according to industry-standard practices, with proper indentation, spacing, and clear, descriptive comments. Each function and method is designed with a single responsibility in mind, which enhances readability and makes the code easier to maintain.
* Comments are added throughout the code to clarify the purpose of key sections, explain the reasoning behind specific decisions, and describe the functionality of various components. This approach ensures that the code is not only functional but also easy to follow for anyone who reviews it.

**By applying these strategies, I was able to develop a fully realized 2D animation game that's both engaging and meets all the assignment requirements.   
  
The code is well-structured, it makes sense logically and was well thought out.   
  
The program is also free of syntax errors, which helps to make sure that the user has a smooth and enjoyable gameplay experience.**