Politechnika Świętokrzyska
Kielce University of Technology

Computer Graphics

Project: 2D game engine

Ahmmed binas

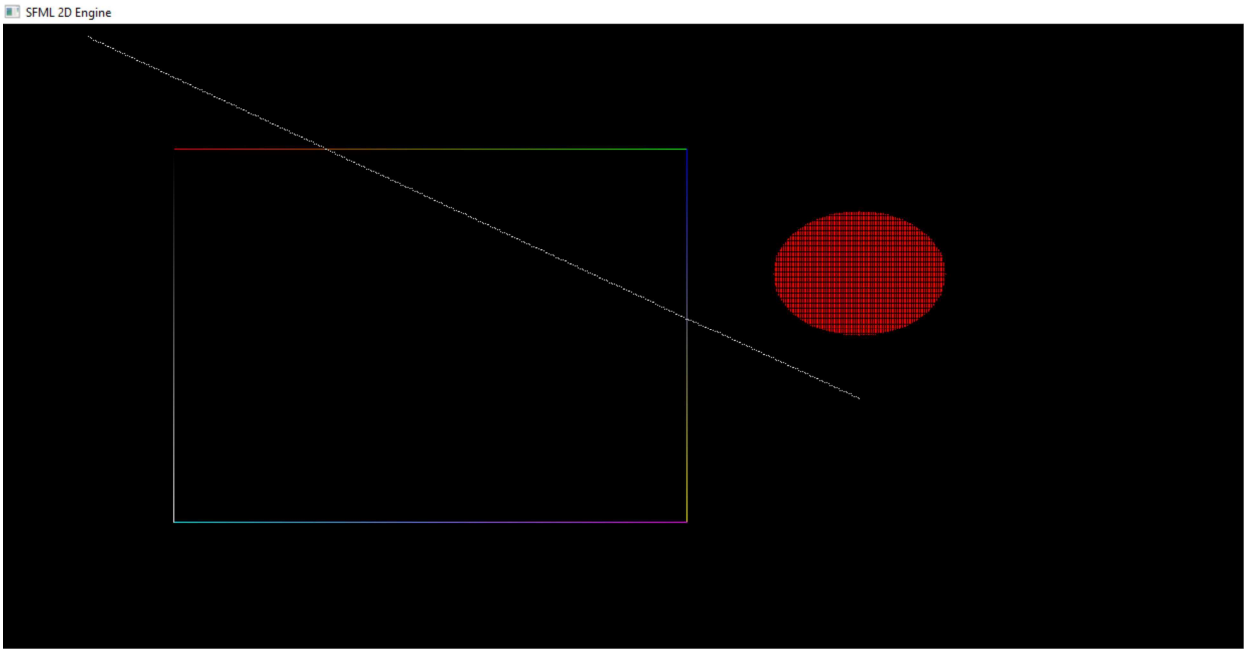
kaliyanathoppu parambu

mgr inż. Konrad Sich

14th December 2023

i-INTRODUCTION

In this SFML-based 2D engine, I utilized C++ to create a versatile framework. The engine incorporates features like geometric rendering, keyboard input handling, and dynamic object animations. Employing the SFML library for graphics and window management, it offers a foundation for game development. The presented code demonstrates a rudimentary rendering of geometric shapes, player input, and dynamic bitmap animations within an SFML window. The output visually showcases rendered lines, rectangles, and an animated bitmap sprite. The technology stack includes C+ +, SFML, and object-oriented design for a flexible and extensible game engine structure.



The code implements an incremental line drawing algorithm for drawing lines efficiently without using Bresenham's algorithm. The algorithm utilizes incremental updates to determine the next pixel in the line, improving performance. This approach iterates through points while considering the slope and efficiently connects them. The PrimitiveRenderer class encapsulates this function, providing a modular and reusable approach for drawing lines in SFML

The circle drawing algorithm in the code uses a simple approach that outlines the circle by iterating through angles and calculating corresponding points on the circumference. The code leverages trigonometric functions (cosine and sine) to determine the coordinates of these points. For circle filling, the algorithm checks whether each pixel falls within the circle's boundary and fills accordingly. The PrimitiveRenderer class encapsulates these circle-drawing functions, offering a modular and reusable approach for drawing and transforming geometric shapes in

SFML.

Player class:

Moving the player using key

explaining the implimentation:

The Player class is implemented with a constructor initializing the player's position, speed, and a BitmapHandler object. Movement methods update the player's position and delegate movement to the BitmapHandler. Input handling responds to arrow keys, updating the player's position accordingly. The update method includes a hardcoded move speed and calls bitmapHandler.update(). The draw method renders the player's rectangle and delegates drawing to the associated BitmapHandler.

GAMEOBJECT CLASS:

Point2D Class:

- Purpose: Represents a 2D point with x and y coordinates.
- Implementation:
  - Constructor initializes the point with provided x and y values.

LineSegment Class:

- Purpose: Represents a line segment between two points.
- Attributes:
  - start and end are instances of the Point2D class, defining the start and end points of the line.
- Constructor:
  - Initializes a LineSegment instance with given start and end points.
- Getter and Setter Methods:
  - Allow retrieval and modification of the start and end points.
- Drawing Method (draw):
  - Converts the start and end points to integer coordinates.
  - Uses PrimitiveRenderer to draw a line between the two points in a specified color.
- Static Zigzag Drawing Method (drawZigzagLine):
  - Takes start and end points, the number of segments, and an amplitude as parameters.
  - Divides the line into segments and draws a zigzag pattern by alternating the amplitude.

GameObject Class:

- Purpose: Represents a game object with rendering capabilities.
- Attributes:
  - renderer is an instance of the PrimitiveRenderer class.
  - sprite is an SFML sprite.
- Methods:
  - Drawing (draw and draww):
    - draw uses PrimitiveRenderer to draw a rectangle.

- draww draws an SFML sprite.

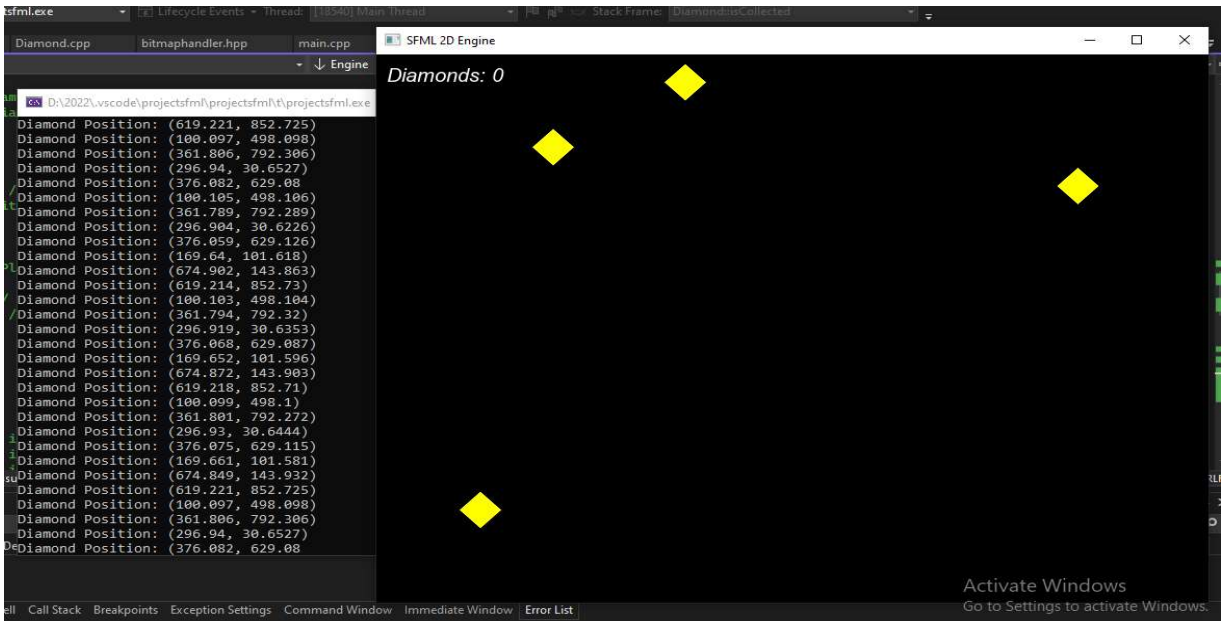
- Transformation Methods (translate, rotate, scale):
  - Delegate to the corresponding methods in the PrimitiveRenderer class.
- Texture Setting Method (setTexture):
  - Sets the texture of the SFML sprite.

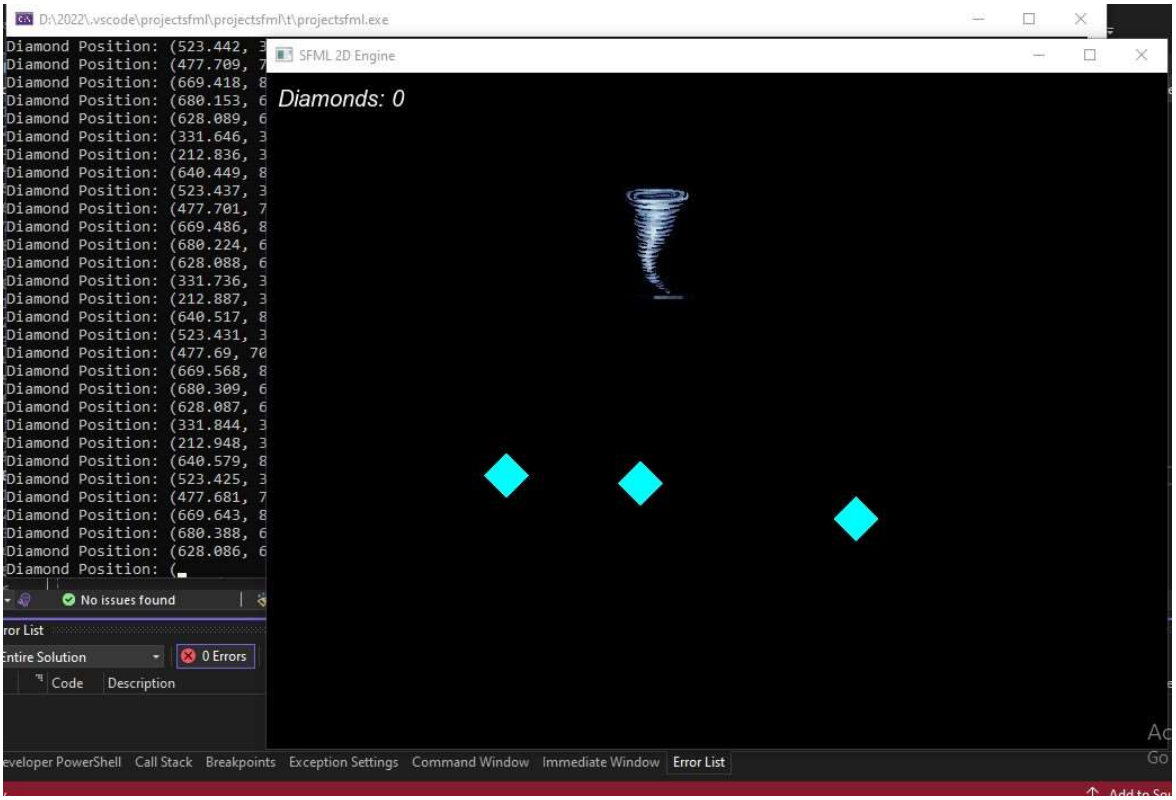
PrimitiveRenderer Class:

- Purpose: Handles primitive rendering operations like drawing lines and rectangles.
- Methods:
  - Drawing Methods (drawRectangle, drawLine, etc.):
    - Implement various drawing operations for lines and rectangles using SFML.

In summary, you've created classes that encapsulate the representation and manipulation of geometric entities in a 2D space. The Point2D class represents points, the LineSegment class represents lines, and the GameObject class provides a framework for rendering and transforming game objects. The PrimitiveRenderer class is responsible for low-level rendering operations.

DEMO SIMPLE TORNADO GAME:





-summarizing the game-

I implemented using SFML and C++, features a player-controlled sprite navigating a dynamic environment with animated diamonds. The player can move the sprite in response to keyboard input, collecting diamonds for points. The diamonds exhibit animated movement, enhancing the gameplay experience. The code employs object-oriented design with a modular structure, allowing for easy expansion and modification. Overall, Tornado game showcases interactive elements, dynamic animations, and leverages the SFML library for a visually engaging 2D gaming experience.

Conclution::::::::;

In the development of a graphics application using Point 2D C++ and SFML, the implementation encompasses crucial classes like for representing coordinates, LineSegment for drawing lines and zigzag patterns, and GameObject facilitating rendering and transformation of objects. The PrimitiveRenderer class handles low-level rendering operations. Core technologies include SFML for graphics and C++ as the programming language. The application demonstrates proficiency in efficient line and shape drawing, keyboard input handling, and object manipulation. Employing an object-oriented approach, the code fosters modularity and readability, providing a solid foundation for further game development and graphics projects.