Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA, Deemed to be University

MINOR ASSIGNMENT-05

Game Programming with C++ (CSE 3545)

Publish on: 01-05-2025 **Submission on:** 08-05-2025
Course Outcome: CO_4 **Program Outcome:** PO_4 **Learning Level:** L_6

Problem Statement:

Experiment with SFML `VertexArray` class to build up a large image efficiently and quickly onto the screen using multiple parts in a single image file (i.e. sprite sheet).

Learning Objectives:

Students will be able to learn `VertexArray` and C++ references to to create a scalable, random and scrolling background for Zombie Arena game.

Answer the followings:

1. Write C++ statements to create 3 references that would refer to **`int`** type **mark** variable.

Code Snippet

```
int mark = 100;
int& ref1 = mark;
int& ref2 = mark;
int& ref3 = mark;
```

2. Find the output of the following code snippet;

```
int main(){
  int num=10;
  int& rnum=num;
  int &r1num=rnum;
  rnum=100;
  cout<<rnum<<" "<<num<<" "<<r1num<<endl;
  return 0;}
```

Output

```
100 100 100
```

3. Find the output of the following code snippet;

```
void update(int& rnum, int vnum, int *pnum){
   rnum=rnum+500;
   vnum=vnum+500;
   *pnum=*pnum+500;
}
int main(){
  int num1=11, num2=22,num3=33;
  update(num1,num2,&num3);
  cout<<num1<<" "<<num2<<" "<<num3<<endl;
  return 0;
}
```

Output

```
511 22 533
```

4. Consider the following C++ code snippet;

```
int& getMax(int &a, int &b) {
    return (a > b) ? a : b;
}
int main() {
    int x=?, y =?;
    int& maxVal = getMax(x, y);
    cout<<maxVal<<endl;
    maxVal = 30;
    cout <<"x = "<< x<< ", y= " <<y;
    return 0;
}
```

Find the output for given x & y

- ❑ 10 10 — 10 10,30
- ❑ 20 20 — 20 20,30
- ❑ 10 20 — 20 10,30
- ❑ 20 10 — 20 30,10
- ❑ 60 40 — 60 30,40
- ❑ 40 60 — 60 40,30

5. Write SFML-C++ code snippet to declare a vertex array with **Quads** type primitive and size of the vertex array $10 \times 10 \times 4$.

Code Snippet

```
VertexArray va;
va.setPrimitiveType(Quads);
va.resize(10 * 10 * 4); // 10×10 tiles, 4 vertices per tile
```

6. Assume that **background_sheet.png** sprite sheet is given to you. Write SFML-C++ code snippet to draw 3 tiles (mud-1, grass, mud-2) onto the screen. you are free to decide the position of each vertex in the current quad and texture co-ordinates will be selected as per the given sprite sheet.

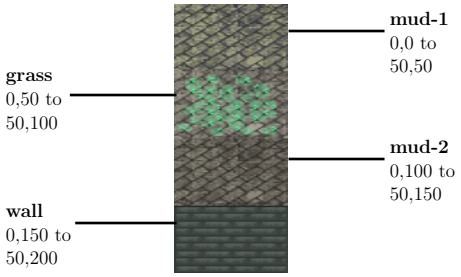

Figure 1: Background sprite sheet with texture coordinates for mud-1, grass, mud-2 and wall

Code Snippet

Code Snippet

```
#include <SFML/Graphics.hpp>
using namespace std;
using namespace sf;

int main() {
    RenderWindow window(VideoMode(300, 100), "Tile Drawing");

    Texture tileset;
    if (!tileset.loadFromFile("sheet.png"))
        return -1;

    VertexArray tiles(Quads, 3 * 4); // 3 tiles × 4 vertices each

    // Lambda function to define a tile (quad) with position and texture
    auto setTile = [](Vertex* quad, Vector2f pos, IntRect texRect) {
        quad[0].position = pos;
        quad[1].position = pos + Vector2f(texRect.width, 0);
        quad[2].position = pos + Vector2f(texRect.width, texRect.height);
        quad[3].position = pos + Vector2f(0, texRect.height);

        quad[0].texCoords = Vector2f(texRect.left, texRect.top);
        quad[1].texCoords = Vector2f(texRect.left + texRect.width, texRect.top);
        quad[2].texCoords = Vector2f(texRect.left + texRect.width, texRect.top + texRect.height);
        quad[3].texCoords = Vector2f(texRect.left, texRect.top + texRect.height);
    };

    // Set 3 different tiles: mud-1, grass, mud-2
    setTile(&tiles[0], Vector2f(0, 0), IntRect(0, 0, 50, 50));       // mud-1
    setTile(&tiles[4], Vector2f(60, 0), IntRect(0, 50, 50, 50));     // grass
    setTile(&tiles[8], Vector2f(120, 0), IntRect(0, 100, 50, 50));   // mud-2

    RenderStates states;
    states.texture = &tileset;

    while (window.isOpen()) {
        Event event;
        while (window.pollEvent(event))
            if (event.type == Event::Closed)
                window.close();

        window.clear();
        window.draw(tiles, states);
        window.display();
    }

    return 0;
}
```

Code Snippet

7. Design a function with the given prototype **`displayBackground(VertexArray& rVA, IntRect arena);`** to draw the background over the window as per the following structure using the above sprite sheet in question 6.

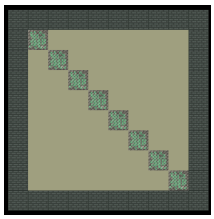

Figure 2: Arena Background

Code Snippet

Code Snippet

```
#include <SFML/Graphics.hpp>
using namespace sf;
void displayBackground(VertexArray& rVA, IntRect arena) {
    const int TILE_SIZE = 50;
    const int TILE_TYPES = 3; // floor, wall, blood
    const int VERTS_IN_QUAD = 4;

    int worldWidth = arena.width / TILE_SIZE;
    int worldHeight = arena.height / TILE_SIZE;

    // Resize the vertex array to fit the level size
    rVA.setPrimitiveType(Quads);
    rVA.resize(worldWidth * worldHeight * VERTS_IN_QUAD);

    // Position index in vertex array
    int currentVertex = 0;
    for (int x = 0; x < worldWidth; ++x) {
        for (int y = 0; y < worldHeight; ++y) {
            // Define position of the current quad
            int xPos = x * TILE_SIZE;
            int yPos = y * TILE_SIZE;

            rVA[currentVertex + 0].position = Vector2f(xPos, yPos);
            rVA[currentVertex + 1].position = Vector2f(xPos + TILE_SIZE, yPos);
            rVA[currentVertex + 2].position = Vector2f(xPos + TILE_SIZE, yPos + TILE_SIZE);
            rVA[currentVertex + 3].position = Vector2f(xPos, yPos + TILE_SIZE);

            // Choose tile type
            int tileType;
            // Border: wall tile
            if (x == 0 || y == 0 || x == worldWidth - 1 || y == worldHeight - 1) {
                tileType = 1; // wall
            }
            // Diagonal line: blood/special tile
            else if (x == y) {
                tileType = 2; // blood or special
            }
            else {
                tileType = 0; // floor
            }

            // Set texture coordinates
            int texX = tileType * TILE_SIZE;
            int texY = 0;

            rVA[currentVertex + 0].texCoords = Vector2f(texX, texY);
            rVA[currentVertex + 1].texCoords = Vector2f(texX + TILE_SIZE, texY);
            rVA[currentVertex + 2].texCoords = Vector2f(texX + TILE_SIZE, texY + TILE_SIZE);
            rVA[currentVertex + 3].texCoords = Vector2f(texX, texY + TILE_SIZE);

            // Move to the next quad
            currentVertex += VERTS_IN_QUAD;
        }
    }
}
```

8. Design a function with the given prototype **`displayBackground(VertexArray& rVA, IntRect arena);`** to draw the background over the window as per the following structure using the above sprite sheet in question 6.

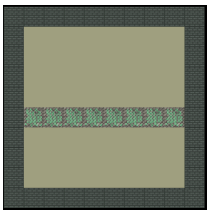

Figure 3: Arena Background

Code Snippet

Code Snippet

```
#include <SFML/Graphics.hpp>
using namespace sf;

void displayBackground(VertexArray& rVA, IntRect arena) {
    const int TILE_SIZE = 50;
    const int TILE_TYPES = 3; // wall, grass, path

    // Set size of the vertex array
    rVA.setPrimitiveType(Quads);
    rVA.resize(arena.width * arena.height * 4);

    // Vertex index
    int vertexIndex = 0;

    for (int x = 0; x < arena.width; ++x) {
        for (int y = 0; y < arena.height; ++y) {
            // Position of current tile
            int xPos = x * TILE_SIZE;

int yPos = y * TILE_SIZE;

            // Access 4 vertices of the quad
            rVA[vertexIndex + 0].position = Vector2f(xPos, yPos);
            rVA[vertexIndex + 1].position = Vector2f(xPos + TILE_SIZE, yPos);
            rVA[vertexIndex + 2].position = Vector2f(xPos + TILE_SIZE, yPos + TILE_SIZE);
            rVA[vertexIndex + 3].position = Vector2f(xPos, yPos + TILE_SIZE);

            // Decide tile type
            int tileType;

            if (x == 0 || y == 0 || x == arena.width - 1 || y == arena.height - 1) {
                tileType = 0; // Wall
            } else if (y == arena.height / 2) {
                tileType = 2; // Path
            } else {
                tileType = 1; // Grass
            }

            int tu = tileType;

            // Set texture coordinates
            rVA[vertexIndex + 0].texCoords = Vector2f(tu * TILE_SIZE, 0);
            rVA[vertexIndex + 1].texCoords = Vector2f((tu + 1) * TILE_SIZE, 0);
            rVA[vertexIndex + 2].texCoords = Vector2f((tu + 1) * TILE_SIZE, TILE_SIZE);
            rVA[vertexIndex + 3].texCoords = Vector2f(tu * TILE_SIZE, TILE_SIZE);

            // Move to next quad
            vertexIndex += 4;
        }
    }
}
```

9. Design a function with the given prototype **`displayBackground(VertexArray& rVA, IntRect arena);`** to draw the background over the window as per the following structure using the above sprite sheet in question 6.

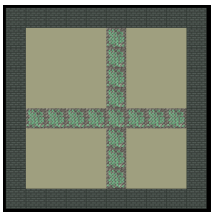

Figure 4: Arena Background

Code Snippet

Code Snippet

```
#include <SFML/Graphics.hpp>
using namespace sf;

void displayBackground(VertexArray& rVA, IntRect arena) {
    const int TILE_SIZE = 50;
    const int TILE_TYPES = 3; // wall, grass, path

    rVA.setPrimitiveType(Quads);
    rVA.resize(arena.width * arena.height * 4);

    int vertexIndex = 0;

    for (int x = 0; x < arena.width; ++x) {
        for (int y = 0; y < arena.height; ++y) {
            int xPos = x * TILE_SIZE;
            int yPos = y * TILE_SIZE;

            // Define quad position
            rVA[vertexIndex + 0].position = Vector2f(xPos, yPos);
            rVA[vertexIndex + 1].position = Vector2f(xPos + TILE_SIZE, yPos);
            rVA[vertexIndex + 2].position = Vector2f(xPos + TILE_SIZE, yPos + TILE_SIZE);
            rVA[vertexIndex + 3].position = Vector2f(xPos, yPos + TILE_SIZE);

            // Tile type logic
            int tileType;

            if (x == 0 || y == 0 || x == arena.width - 1 || y == arena.height - 1) {
                tileType = 0; // Wall
            } else if (x == arena.width / 2 || y == arena.height / 2) {
                tileType = 2; // Path
            } else {
                tileType = 1; // Grass
            }

            int tu = tileType;

            // Set texture coordinates
            rVA[vertexIndex + 0].texCoords = Vector2f(tu * TILE_SIZE, 0);
            rVA[vertexIndex + 1].texCoords = Vector2f((tu + 1) * TILE_SIZE, 0);
            rVA[vertexIndex + 2].texCoords = Vector2f((tu + 1) * TILE_SIZE, TILE_SIZE);
            rVA[vertexIndex + 3].texCoords = Vector2f(tu * TILE_SIZE, TILE_SIZE);

            vertexIndex += 4;
        }
    }
}
```

5. Let you have pressed the keys: **W**, **Return**, **Num0** and **Num1** respectively. Write a code snippet to handle the events by polling and show the type of key has been pressed.

Code Snippet

```
#include <SFML/Graphics.hpp>
#include <iostream>
using namespace std;
int main() {
    RenderWindow window(VideoMode(400, 300), "Key Press Events");

    while (window.isOpen()) {
        Event event;
        while (window.pollEvent(event)) {
            // Handle window close
            if (event.type == Event::Closed)
                window.close();

            // Handle key presses
            if (event.type == Event::KeyPressed) {
                switch (event.key.code) {
                    case Keyboard::W:
                        cout << "W key pressed" << endl;
                        break;
                    case Keyboard::Return:
                        cout << "Return (Enter) key pressed" << endl;
                        break;
                    case Keyboard::Num0:
                        cout << "Numpad 0 key pressed" << endl;
                        break;
                    case Keyboard::Num1:
                        cout << "Numpad 1 key pressed" << endl;
                        break;
                    default:
                        cout << "Some other key pressed" << endl;
                        break;
                }
            }
        }
        window.clear(Color::Black);
        window.display();
    }
    return 0;
}
```

6. Assume **ON** and **OFF** are two states in a game with sprites **player.png** and **bloater.png**. Initially game is in **ON** state and the sprite, **player**, is drawn onto the game window. Game state can be changed with the key pressed **Return**. Construct a program to draw player sprite in ON state and bloater sprite in OFF state. `window.clear(Color::Red);` may be used to change in default background color.

Code Snippet

Code Snippet

```
#include <SFML/Graphics.hpp>
#include <iostream>
using namespace sf;
enum GameState { ON, OFF };
int main() {
    // Create the game window
    RenderWindow window(VideoMode(600, 400), "Game State Toggle");
    // Load textures
    Texture playerTexture;
    if (!playerTexture.loadFromFile("player.png")) {
        std::cerr << "Error loading player.png\n";
        return -1;
    }
    Texture bloaterTexture;
    if (!bloaterTexture.loadFromFile("bloater.png")) {
        std::cerr << "Error loading bloater.png\n";
        return -1;
    }
    // Create sprites
    Sprite playerSprite(playerTexture);
    Sprite bloaterSprite(bloaterTexture);
    // Optional: Set sprite positions
    playerSprite.setPosition(200, 150);
    bloaterSprite.setPosition(200, 150);

    GameState currentState = ON;

    // Game loop
    while (window.isOpen()) {
        Event event;
        while (window.pollEvent(event)) {
            // Close event
            if (event.type == Event::Closed)
                window.close();

            // Toggle state with Return key
            if (event.type == Event::KeyPressed && event.key.code == Keyboard::Return) {
                currentState = (currentState == ON) ? OFF : ON;
            }
        }

        // Clear the screen with red background
        window.clear(Color::Red);

        // Draw appropriate sprite based on state
        if (currentState == ON)
            window.draw(playerSprite);
        else
            window.draw(bloaterSprite);

        // Display the window
        window.display();
    }

    return 0;
}
```