MINOR ASSIGNMENT-02

Game Programming with C++ (CSE 3545)

Publish on: 11-03-2025 **Submission on:** 18-03-2025
Course Outcome: CO_2 **Program Outcome:** PO_2 **Learning Level:** L_3

Problem Statement:

Experiment with classes, class members, objects, constructors and namespaces to learn basic framework of Game Programming using C++ with Simple and Fast Multimedia Library(SFML).

Learning Objectives:

Students will be able to learn the uses of predefined classes, objects, method calls, object as function argument and function returning object.

Answer the followings:

1. State the header file(s) and name of the namespace(s) required to use SFML libary to design game programs. Aslo write the **compilation and execution** commands in Linux-based enviroment.

ANSWER::

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <SFML/System.hpp>

using namespace sf;
```

```
g++ main.cpp -o game -lsfml-graphics -lsfml-window -lsfml-system
./game
```

2. Assume that you have a class as `calculateArea{ ... };`, Write the C++ statement to declare 4 objects of that class.

ANSWER::

```
calculateArea obj1, obj2, obj3, obj4;
```

3. Let say, `void setValues(int, int);` and `void getValues( );` are the public member functions for the class `calculateArea{ ... };`. Write the C++ statements to call both the functions using object(s) of that class.

SFML-C++ statements::

```
obj1.setValues(5, 10);
obj1.getValues();
```

4. Write C++ statements to open a window with 960 pixels wide by 540 pixels high.

SFML-C++ statements::

```
RenderWindow window(VideoMode(960, 540), "SFML Window");
```

5. Write the SFML-C++ statements for the Game loop/ Application loop to stay in the program until the player want to quit for the Game Timber. Additionally enable the **Esc** key to terminate the game loop, when the key is pressed.

SFML-C++ statements::

```
while (window.isOpen()) {
    Event event;
    while (window.pollEvent(event)) {
        if (event.type == Event::Closed ||
           (event.type == Event::KeyPressed && event.key.code == Keyboard::Escape)) {
            window.close();
        }
    }
}
```

6. Consider the two SFML classes, `Texture` and `Sprite`, that will take care of drawing sprites into the screen. Use the two classes to draw an image **`soa.jpeg`** onto the window of size 960×540.

SFML-C++ statements::

```
Texture texture;
texture.loadFromFile("soa.jpeg");
Sprite sprite;
sprite.setTexture(texture);
RenderWindow window(VideoMode(960,540),"Image Display");
while(window.isOpen()){
Event event;
while (window.pollEvent(event)){
if(event.type==Event::Closed)
window.close();}
window.clear();
window.draw(sprite);
window.display();
}
```

7. Write the SFM-C++ statements to fly the image **`soa.jpeg`** across the screen from top center to buttom of the screen. You can make use of the **`Sprite`** class method `setScale` to set the scale factors of the sprite object.

SFML-C++ statements::

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

int main() {
    // Load the texture
    Texture texture;
    if (!texture.loadFromFile("soa.jpeg")) {
        return -1; // Exit if image not loaded
    }

    // Create sprite and apply transformations
    Sprite sprite;
    sprite.setTexture(texture);
    sprite.setScale(0.5f, 0.5f);          // Scale
image to 50%

sprite.setScale(0.5f, 0.5f);
sprite.setPosition(480, 0);  // top center

float speed = 100.0f; // pixels/sec
Clock clock;

while (window.isOpen()) {
    float dt = clock.restart().asSeconds();
    sprite.move(0, speed * dt);  // move down
    // window draw logic here
}
```

8. Write the SFM-C++ statements to fly the image **`soa.jpeg`** across the screen from left to right. You can make use of the **`Sprite`** class method `setScale` to set the scale factors of the sprite object.

SFML-C++ statements::

```
sprite.setScale(0.5f, 0.5f);
sprite.setPosition(0, 270); // middle left

float speed = 100.0f; // pixels/sec
Clock clock;

while (window.isOpen()) {
    float dt = clock.restart().asSeconds();
    sprite.move(speed * dt, 0); // move right
}
```

9. Let us pretend the speed of a **spriteBee** is 200 pixels per second. Calculate the amount of time the Bee will take to cross the entire width of the screen that is, 2000 pixels wide. Also calculate the bit rate per second (i.e. bps) of the **spriteBee**, if 1 pixel contains 8 bits.

Show the computation::

Time = Distance / Speed = 2000 / 200 = 10 seconds

Bitrate = 200 pixels/sec x 8 bits = 1600 bps

10. The `Clock clock; clock.restart()` function restart the clock. The clock is restarted in every frame to know how long each and every frame takes. *In addition, however,* ***`clock.restart();`*** *returns the amount of time that has elapsed since the last time we restarted the clock.* So, compute the distance a `spriteBee` object will cover in a frame assuming the speed of the `spriteBee` is **beeSpeed** pixels/second.

Show the computation::

```
// Assume clock is already created
Clock clock;
// Inside the game loop
while (window.isOpen()) {

    // Restart clock and get elapsed time
    Time deltaTime = clock.restart();
    float dt = deltaTime.asSeconds(); // Convert Time to float seconds

    // Assuming beeSpeed is defined (pixels per second)
    float beeSpeed = 300.0f; // example speed

    // Compute distance the bee will move in this frame
    float distance = beeSpeed * dt;

    // Move the spriteBee by that distance
    spriteBee.move(distance, 0); // Move in x-direction (right)

    // (your other game loop code)
}
```

12. Write SFML-C++ statements to display and set the center of the a message text " **SOA UNIVERSITY**" to the center of the screen of size 1920×1080. Additionally set the character size 100, text color Red and font family `KONIKAP_.ttf`.

SFML-C++ statements::

```
Font font;
font.loadFromFile("KONIKAP.ttf");

Text text("SOA UNIVERSITY", font, 100);
text.setFillColor(Color::Red);

FloatRect textRect = text.getLocalBounds();
text.setOrigin(textRect.left + textRect.width / 2.0f,
            textRect.top + textRect.height / 2.0f);
text.setPosition(1920 / 2.0f, 1080 / 2.0f);
```

13. Construct SFML-C++ statements to draw a red filled rectangle shape of width **X** and height **Y** on the screen 1920×1080 at the center of the screen.

SFML-C++ statements::

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

int main() {
    // Create the window
    RenderWindow window(VideoMode(1920, 1080), "Red Rectangle Centered");

    // Define width and height
    float X = 400.0f; // Example width
    float Y = 200.0f; // Example height

    // Create a rectangle shape
    RectangleShape rectangle(Vector2f(X, Y));

    // Set fill color to red
    rectangle.setFillColor(Color::Red);

    // Set origin to center of rectangle
    rectangle.setOrigin(X / 2.0f, Y / 2.0f);

    // Position rectangle at center of the screen
    rectangle.setPosition(1920 / 2.0f, 1080 / 2.0f);

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

        window.clear();
        window.draw(rectangle); // Draw the rectangle
        window.display();
    }

    return 0;
}
```

14. Construct SFML-C++ statements to draw **FOUR** green filled circle shapes of radius **X** on the screen 1920×1080 close to 4 corners of the screen. Additionally one center stretched circle with red filled of the same radius using **sf::CircleShape** Class Reference.

SFML-C++ statements::

```
float radius = X;
CircleShape corner(radius);
corner.setFillColor(Color::Green);

std::vector<Vector2f> positions = {
    {0, 0},
    {1920 - 2*radius, 0},
    {0, 1080 - 2*radius},
    {1920 - 2*radius, 1080 - 2*radius}
};

for (auto pos : positions) {
    corner.setPosition(pos);
    window.draw(corner);
}

CircleShape center(radius * 2); // stretched
center.setFillColor(Color::Red);
center.setPosition((1920 - 4*radius)/2.0f, (1080 - 4*radius)/2.0f);
```