Games and Multimedia

Computer Graphics

Report Computer Graphics Special Exam

Professor: Gustavo Reis

Alexandre Rodrigues

2200728

Table of Contents

Initialization and Setup

1. **SDL and OpenGL Initialization:**

    - The program begins by initializing SDL using SDL_Init(SDL_INIT_VIDEO), which sets up the video subsystem necessary for creating an SDL window.
    - A window is created using SDL_CreateWindow, with specified dimensions and OpenGL settings.
    - The OpenGL context is created using SDL_GL_CreateContext, which is necessary to manage OpenGL operations.
    - GLAD is then initialized to load OpenGL functions using gladLoadGLLoader.

2. **Shader Compilation and Linking:**

    - The vertex and fragment shaders are defined as string literals. These shaders are essential for rendering objects on the screen.
    - compileShader is responsible for compiling each shader from source code. It checks for compilation errors and logs them if any are found.
    - linkShaderProgram links the compiled vertex and fragment shaders into a single shader program. This program is used by OpenGL to render objects with the defined shaders.

Texture Management

3. **Texture Loading with Color Keying:**

    - loadTexture is a function that loads textures from image files using the STB image library. It supports color keying, which makes certain colors in the texture transparent.
    - If color keying is enabled, the function iterates over the image pixels, comparing each pixel's RGB values to the color key. Matching pixels have their alpha value set to 0 (fully transparent).
    - The loaded texture is then sent to the GPU, where it is stored and used for rendering. OpenGL texture parameters like wrapping and filtering are also set in this function.

Sprite Animation and Rendering

4. **Sprite Animation Structure:**

    - SpriteAnimation is a structure designed to manage animated sprites. It stores essential information such as texture ID, frame count, current frame, frame duration, and position on the screen.
    - The structure helps in controlling the animation playback by advancing frames based on the elapsed time (elapsedTime), and it is used later during rendering.

5. **Sprite Animation Update:**

    - The updateSpriteAnimation function updates the current frame of an animation based on the time elapsed since the last frame update. It ensures that animations loop correctly by resetting the frame to the first one after reaching the end.
    - updateTextureCoords calculates and updates the texture coordinates for the current animation frame. This is critical for correctly mapping the appropriate part of the texture to the quad being rendered.

6. **Rendering Objects:**

    - The renderObject function handles rendering textured objects, including animated sprites and static backgrounds. It sets the shader uniforms for model, view, and projection matrices, which are used to transform the object's position and orientation on the screen.
    - The texture is bound, and the object's VAO (Vertex Array Object) is used to render it using the glDrawElements function.

7. **Text Rendering:**

    - The RenderText function is responsible for rendering text on the screen using a sprite sheet font. It calculates the appropriate texture coordinates for each character and renders them as quads.

- The function iterates through the string, rendering each character by adjusting the x position and updating the texture coordinates accordingly.

Main Game Loop

8. **Setting Up the Game World:**

    - The main loop initializes several components, including background rendering, sprite animations, and text rendering.
    - The glm::ortho function is used to set up an orthographic projection matrix, which is ideal for 2D rendering.
    - The view matrix is set to identity, meaning no additional transformations are applied to the scene.

9. **Handling Events and Updating the Game:**

    - The main loop continuously checks for SDL events using SDL_PollEvent. If a quit event is detected, the loop breaks, and the game exits.
    - The loop also manages frame timing, ensuring consistent animation speeds regardless of the hardware performance.
    - Each frame, the static background is rendered first, followed by the animations, and finally, the text.

10. **Final Rendering and Cleanup:**

    - After rendering all game objects, SDL_GL_SwapWindow is called to display the rendered frame on the screen.
    - Upon exiting the main loop, all OpenGL and SDL resources are cleaned up to prevent memory leaks.