Task Progress Update Report

Name: LIM SHI KAI (Sky)

Update Date: 17-01-2025

---

1. Overview of Tasks

Task 1 : R&D on wxWidgets, Boost.Signals2, SFML

Objective : Research and evaluate wxWidgets, Boost.Signals2, and SFML as potential replacements for Qt libraries in the project and provide insights for the team lead to assess feasibility.

Status : Completed

Details :

- **wxWidgets**:
    - Installed and configured wxWidgets following a detailed tutorial video for guidance.
    - Documented the installation process step-by-step for ease of reference by the team.
    - Implemented UI components using wxWidgets, replacing Qt's `QWidget` with `wxFrame`, `wxPanel`, etc.
    - Identified and resolved rendering and layout issues during the initial integration phase.
    - Shared findings with the team lead regarding the advantages of wxWidgets for cross-platform applications and its ease of use for basic UI needs.
- **Boost.Signals2**:
    - Integrated Boost.Signals2 to replace Qt's signal-slot mechanism.
    - Demonstrated robust signal-slot implementation in the project to the team lead.
    - Showcased practical use cases, such as linking UI actions to internal events, to highlight its suitability and efficiency in the current architecture.
- **SFML**:
    - Utilized SFML for 2D rendering and scene management in place of Qt's rendering framework.
    - Integrated SFML's `sf::RenderWindow` for OpenGL context and event handling.

    - Tested SFML's capabilities for managing transformations like zooming, panning, and scaling using `sf::Transform`.
- **Integration and Debugging**:
    - Converted an existing Qt-based codebase into a hybrid project using wxWidgets, Boost.Signals2, and SFML.
    - Configured Visual Studio project settings for successful library integration:
        - Set up include paths, library directories, and linker options for Debug and Release builds.
        - Debugged and resolved issues related to missing SFML libraries and runtime DLLs.
    - Documented the complete integration process and highlighted best practices for future use.
- New Requirements on another R&D after Team Lead Discussion
    - UI: ImGui
    - Signals: Boost.Signals2
    - Graphic View/Scene/Item: GLFW+GLAD

Task 2 : R&D on ImGUI, Boost.Signals2, GLFW+FLAD

Objective : Research and evaluate ImGUI, Boost.Signals2, and GLFW + GLAD as potential components for replacing Qt libraries in the project, and discuss the implementation progress with the team lead.

Status : Completed

Details :

- **ImGUI**:
    - Integrated ImGUI as a replacement for the Qt-based UI framework.
    - Created dynamic UI elements, including menus, toolbars, and dialogs.
    - Implemented modal dialogs for file operations:
        - "Load Image" for selecting `.txt` files.
        - "Save Image" for saving files as `.png`.
    - Enhanced the interface with collapsible control panels, organized sections, and dynamic updates.
    - Adjusted layout settings to maintain a clean and user-friendly experience, ensuring all UI components are functional and visually consistent.

- **Boost.Signals2**:
    - Used Boost.Signals2 to replace Qt's signal-slot mechanism for handling events.
    - Configured signal-slot connections for real-time communication between UI controls and backend functionalities.
    - Verified proper integration of Boost.Signals2 into the new architecture by testing event-driven interactions like button clicks and dynamic UI updates.
- **GLFW + GLAD**:
    - Integrated GLFW to manage OpenGL contexts and handle user input.
    - Configured GLAD for loading modern OpenGL functions.
    - Implemented rendering logic using GLFW's windowing system and OpenGL API, replacing Qt's rendering mechanisms.
    - Addressed setup challenges, including linking library dependencies and resolving runtime issues.
- **Team Lead Discussion**:
    - Provided a progress update on R&D during a meeting via Microsoft Teams.
    - Presented the integration of ImGUI, Boost.Signals2, and GLFW + GLAD, showcasing their roles in replacing Qt libraries.
    - Demonstrated the transformed application's core functionalities:
        - UI rendering with ImGUI.
        - Event handling with Boost.Signals2.
        - Rendering context management with GLFW + GLAD.
    - Highlighted completed milestones and discussed next steps, including the integration of SDL for future enhancements.
- **Integration and Documentation**:
    - Documented the installation and linking processes for ImGUI, Boost.Signals2, and GLFW + GLAD.
    - Included step-by-step instructions for replicating the setup, troubleshooting potential issues, and ensuring seamless integration in future builds.
- **New R&D requirements:**
    - Add new functionalities beyond the current implementation.
    - Remained ImGUI, Boost.Signals2 for UI and also Signal.
    - Use SDL for rendering and scene management.

Task 3 : Comparison between SFML & GLFW+GLAD

Objective : Evaluate and compare SFML and GLFW+GLAD for their suitability in managing GraphicsView, GraphicsScene, and GraphicsItem.

Status : Completed

Details :

- Conducted an in-depth analysis of SFML and GLFW+GLAD, focusing on their graphics rendering and event handling capabilities.
- Studied the structure and implementation of GraphicsView, GraphicsScene, and GraphicsItem to identify requirements for replacement.
- Compiled findings in a comparison table to highlight the strengths and limitations of each library.
- Shared the some results with the team lead during a progress update meeting to facilitate informed decision-making.

Aspect	**SFML**	**GLFW + GLAD**
Abstraction Level	High-level, beginner-friendly API.	Low-level, requires OpenGL knowledge.
Learning Curve	Short and intuitive.	Steep due to OpenGL complexity.
Performance	Efficient for simple 2D rendering; slight overhead from abstractions.	Optimized for advanced rendering; high customizability.
Flexibility	Limited; focuses on predefined modules.	Highly flexible; suitable for 2D and 3D applications.
Event Handling	Built-in event system; easy to use.	Callback-based event handling; requires manual implementation.
Rendering Support	2D rendering with built-in abstractions like views and scenes.	Both 2D and 3D rendering; requires manual scene and view management.
Setup Complexity	Simple setup with minimal dependencies.	Complex setup; requires external libraries for additional features.

Documentation	Beginner-friendly and easy to follow.	Assumes prior OpenGL knowledge; extensive community resources.
Cross-Platform Support	Windows, macOS, Linux.	Windows, macOS, Linux.
Use Cases	Ideal for 2D games and multimedia applications.	Suitable for advanced 2D/3D applications with custom rendering needs.

Task 4 : Code sharing via GitHub

Objective : Share the updated codebase with the team via GitHub for collaboration, review, and feedback.

Status : Completed

Details :

- Prepared the updated project code, which integrates ImGui, Boost.Signals2 and GLFW+GLAD for upload to the team's GitHub repository.
- Encountered an issue with large file sizes preventing successful push to GitHub.
    - **Resolution:** Implemented Git Large File Storage (LFS) to manage and upload large files.
    - Followed guidance from a video tutorial to configure and use Git LFS efficiently.
- Increased Git buffer size and adjusted HTTP post-buffer settings to optimize the push process.
- Verified that all files were successfully uploaded, including the updated libraries, configuration files, and project code.
- Organized the repository structure for clarity, ensuring all team members could easily access and navigate the files.
- After a Microsoft Teams progress meeting, I shared the GitHub repository link with the team lead for review and feedback.

Task 5 : R&D on ImGui, Boost.Signals2, and SDL

Objective : Research and evaluate ImGui, Boost.Signals2, and SDL are replacements for Qt libraries, focusing on rendering, event handling, and UI functionalities.

Status : In Progress

Details :

- Start the study about the SDL libraries.

2. Roadblocks/Challenges

- **Integration Complexity**:
  Integrating GLFW+GLAD required a deeper understanding of OpenGL, making the transition from Qt challenging due to the steep learning curve.
- **Rendering Issues**:
  Debugging grayscale rendering and ensuring proper OpenGL texture management posed initial difficulties, especially with texture configuration and shader adjustments.
- **Zoom and Pan Functionality**:
  Maintaining consistent zoom and pan behavior in the OpenGL viewport required extensive debugging and modifications to matrix transformations and input handling.
- **File Operation Challenges**:
  Implementing robust file dialogs for loading `.txt` files and saving `.png` files required addressing compatibility issues and ensuring seamless error handling.
- **GitHub Large File Uploads**:
  Encountered issues with pushing large project files to GitHub due to size limitations, requiring the use of Git Large File Storage (LFS) for resolution.
- **Library Comparisons**:
  Balancing the strengths and limitations of SFML and GLFW+GLAD for graphics-related tasks required in-depth analysis and practical testing to provide informed recommendations.

3. Conclusion

- **Successful R&D Implementation**:
  Completed the research and implementation of wxWidgets, Boost.Signals2, SFML, ImGui, and GLFW+GLAD as alternatives to Qt, demonstrating their feasibility for UI, signal-slot, and rendering functionalities.
- **Improved Application Features**:
  Enhanced the project with new features, including a dynamic ImGui-based control panel, modal file dialogs, and seamless zoom, pan, and rendering functionalities.
- **Streamlined Collaboration**:
  Organized and shared the updated codebase via GitHub, ensuring accessibility for the team and enabling collaboration and feedback from the team lead.
- **Comparison Findings**:
  Documented a detailed comparison between SFML and GLFW+GLAD, aiding in the decision-making process for future development paths.
- **Next Steps**:
  Focus on integrating SDL for graphics scene management as per new R&D requirements while continuing to refine the current architecture for performance and usability.