Objectives

- Compare SFML with GLFW and GLAD for handling `GraphicsView`, `GraphicsScene`, and `GraphicsItem`.
- Study and document the structure and functionality of `GraphicsView`, `GraphicsScene`, and `GraphicsItem`.
- Conduct an in-depth analysis of the pros and cons of SFML and GLFW+GLAD for graphics rendering.
- Provide a progress update to the team lead and push the updated code to GitHub for review.
- Begin working on new R&D requirements for replacing Qt libraries.

Activities

1. **Comparison of SFML and GLFW+GLAD**
    - Key differences and applications:
        - **High-Level vs. Low-Level**:
            - SFML is a high-level library ideal for 2D rendering, with built-in abstractions for views, scenes, and sprites.
            - GLFW and GLAD are low-level libraries focused on OpenGL context creation and function loading, requiring manual implementation of abstractions.
        - **Ease of Use**:
            - SFML has a beginner-friendly API and short learning curve.
            - GLFW+GLAD demands OpenGL knowledge, making it harder for beginners.
        - **Flexibility and Performance**:
            - SFML is efficient for simple 2D graphics but less flexible.
            - GLFW+GLAD provide granular control, optimizing advanced rendering and 3D performance.
        - **Setup and Resources**:
            - SFML is straightforward to set up with built-in support for audio, networking, and image loading.

        - GLFW+GLAD setup is complex, with additional dependencies for audio and other functionalities.
    - **Suitability**:
        - SFML is ideal for 2D games and multimedia applications.
        - GLFW+GLAD are better suited for advanced 2D/3D applications requiring high customizability.
2. **Studying `GraphicsView`, `GraphicsScene`, and `GraphicsItem`**
    - **GraphicsView**:
        - Provides a viewport for scene rendering and manages transformations like zoom and pan.
        - Includes functions like `zoom()`, `pan()`, `fitInView()`, and `mapToScene()`.
    - **GraphicsScene**:
        - Manages `GraphicsItem` objects and renders them in sequence.
        - Supports functions like `addItem()`, `draw()`, and `itemAt()` for hit testing.
    - **GraphicsItem**:
        - Serves as the base class for graphical elements with transformations (position, rotation, scale).
        - Includes derived classes like `RectItem` (rectangles) and `TextureItem` (images).
        - Handles rendering and hit testing with `draw()` and `contains()`.
3. **Pros and Cons of SFML vs. GLFW+GLAD**
    - **SFML**:
        - Pros:
            - Beginner-friendly.
            - Integrated modules for graphics, audio, and networking.
            - Simple event system.
        - Cons:
            - Limited flexibility and performance for advanced rendering.
            - Abstractions introduce slight overhead.
    - **GLFW + GLAD**:
        - Pros:

        - High customizability and extensibility.
        - Optimized for performance-critical and advanced rendering tasks.
        - Larger OpenGL-focused community for support.
    - Cons:
        - Steep learning curve due to OpenGL complexity.
        - Requires additional libraries for non-graphics tasks like audio.

4. **Team Lead Meeting**
    - Provided a progress update and pushed the code to GitHub for review.
    - Discussed the R&D findings and implementation methods.
5. **New R&D Requirements**
    - Updated focus for free replacement libraries:
        - **UI**: ImGUI
        - **Signals**: Boost.Signals2
        - **Graphics**: SDL (to replace `GraphicsView`, `GraphicsScene`, and `GraphicsItem`).
        - **I/O**: OpenGL
    - Additional tasks:
        - Add new functionalities beyond the current implementation.
        - Focus on detailed comparison of the graphics aspect.
6. **Started Work on New R&D Requirements**
    - Began studying SDL's capabilities for replacing `GraphicsView`, `GraphicsScene`, and `GraphicsItem`.
    - Planned integration of OpenGL for I/O and rendering functionalities.

Achievements

- Conducted a detailed comparison of SFML and GLFW+GLAD, highlighting their strengths and limitations.
- Documented the structure and functionality of `GraphicsView`, `GraphicsScene`, and `GraphicsItem`.
- Pushed updated code to GitHub for team lead review.

- Initiated work on new R&D requirements with a focus on SDL and OpenGL integration.

Problems & Solutions

1. **Problem**: Steep learning curve for GLFW+GLAD compared to SFML.
   - **Solution**: Continued in-depth study of OpenGL fundamentals to bridge knowledge gaps.
2. **Problem**: Limited abstraction in GLFW+GLAD for graphics scenes and items.
   - **Solution**: Began exploring SDL as a potential replacement for these abstractions.