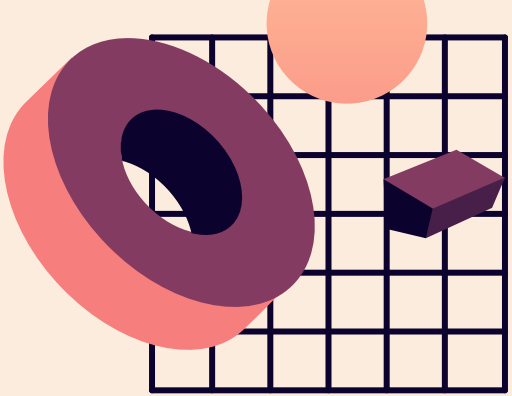


YouTube Video Link

https://youtu.be/kDSv-G857xg

ScummVM Presentation

Group #25 - 2024/10/11 - CISC 322/326

Group Lead: Elill Mathivannan

Presenters: Henry Xiu, Amaan Javed

Members: Momin Alvi, Elias Frigui, Ahmad Tahir

Agenda

- Overview
- Derivation Process
- Architectural Styles
- System Breakdown and Interactions
- Evolution of System
- Control and Data Flow
- Concurrency
- Division of Responsibilities
- Use Cases
- Lessons Learned
- Conclusion

Derivation Process

- How:
  - Looked at the source code repository
  - Organized components
  - Based on procedure calls, drew their relationships
  - Made calculated assumptions about the styles used
- Reasoning about the styles:
  - **Layered Architecture:** Modularity + Scalability
  - **Interpreter Style:** Adaptability
  - **Publish-Subscribe:** Asynchronous communication

Alternatives

- Some alternatives we considered:
  - **Microservices:** Too much complexity for the use case
  - **Client-Server:** Too much latency, not necessary

Architectural Styles

- Three main architectural styles:
  - **Layered Architecture**
    - Presentation Layer
    - Interface Layer
    - Game Engine Layer
  - **Interpreter Architecture**
  - **Publish-Subscribe Architecture**

Layered Architecture

- **Presentation Layer**
  - Handles user interactions
- **Interface Layer**
  - Bridges game engines and platform specifics
- **Game Engine Layer**
  - Implements various game engines

Interpreter Architecture

- Allows processing of game-specific scripts
- Acts as a bridge between game logic and subsystems
- Facilitates the addition of new game engines

Publish-Subscribe Architecture

- Supports independent event handling across subsystems
- Subsystems subscribe to relevant events
- Improves real-time responsiveness

System Breakdown and Interactions

- Main components:
  - Game Engine Layer
  - VM Subsystem
  - Graphics Subsystem
  - Audio Subsystem
  - Input Subsystem
  - UI Subsystem
  - Common Utilities

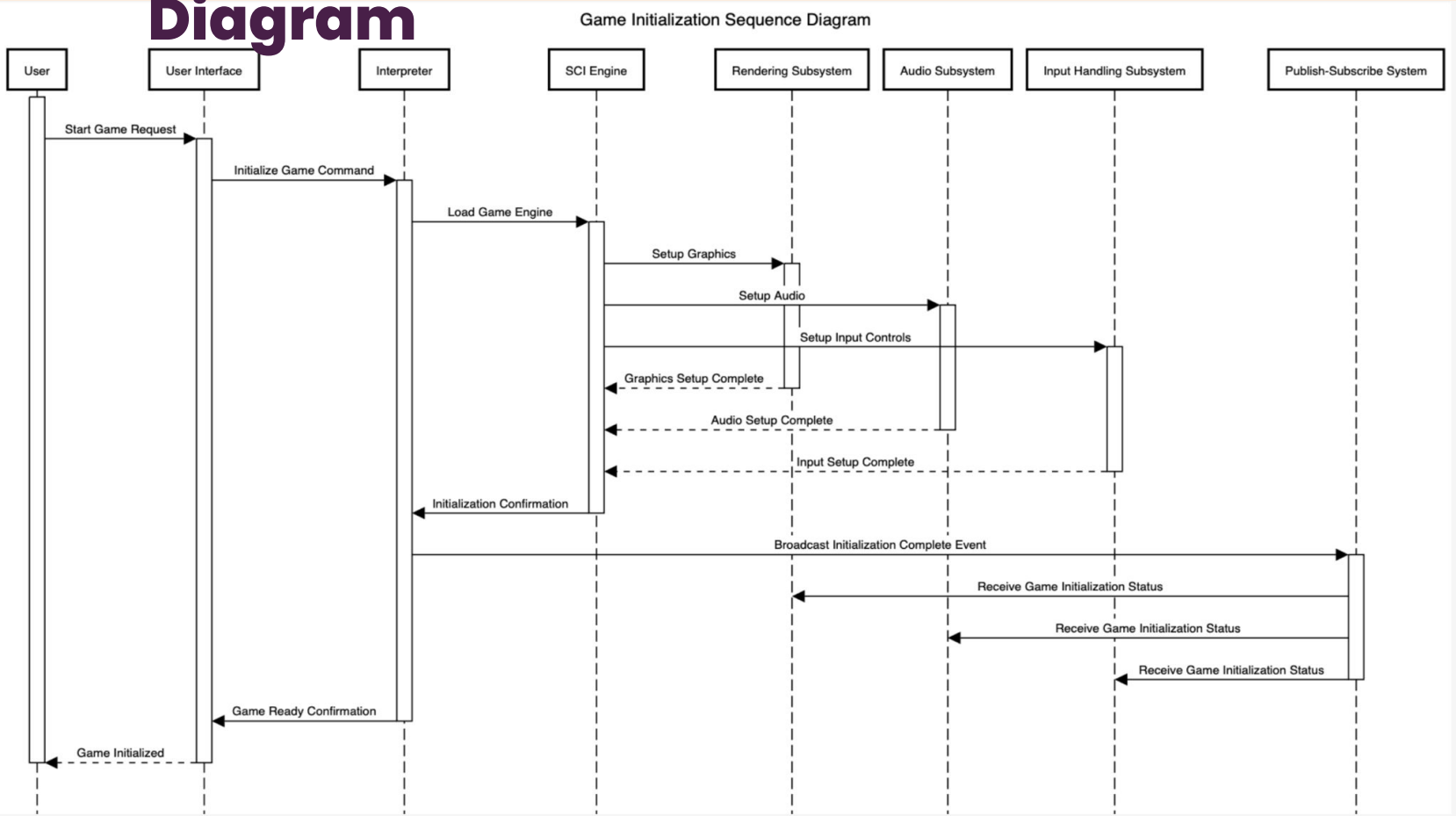
Evolution of System

- Layered Architecture Components
  - Implementing changes on the layer level
- Interpreter Architecture Components
  - Allows for new interpreters for different game engines
- Publish-Subscribe Architecture Components
  - Allows for asynchronous communication between subsystems

Control and Data Flow

- Within layered architecture:
  - Flows from the topmost layer, to the bottommost
  - Presentation Layer -> Interface Layer -> Game Engine Layer
- Between components:
  - Lower levels (hardware + platform-specific operations) to top levels (game logic layers) managed by interpreter
- Within publish-subscribe:
  - Events are published by the interpreter
  - Relevant subsystems subscribe to these events to receive updates

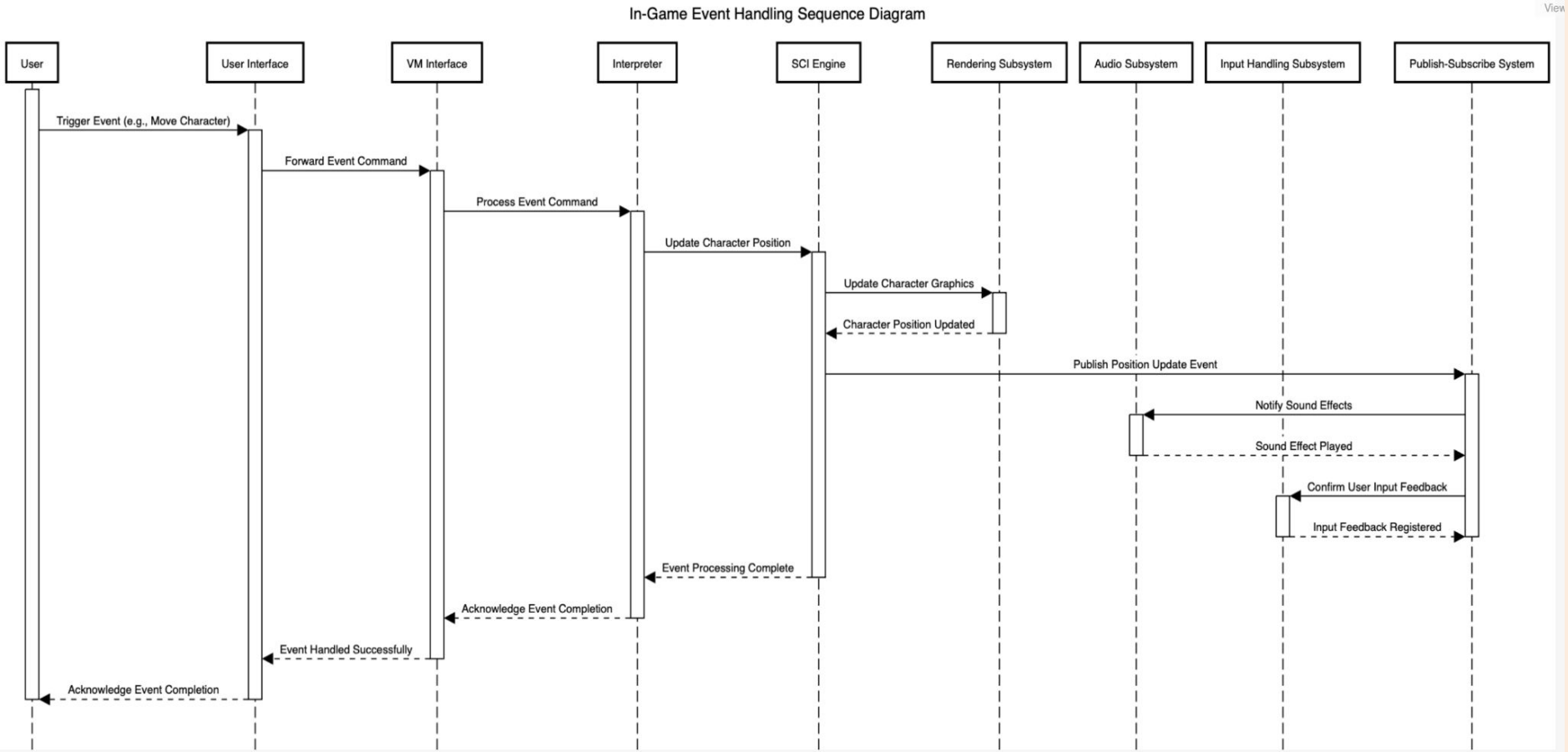
Game Initialization Sequence Diagram



Concurrency

- **Interpreter** processes scripts while subsystems run independently
- **Layered architecture** supports parallel operations
- **Publish-subscribe** allows multiple events to be handled concurrently

Game Event Handling Sequence Diagram



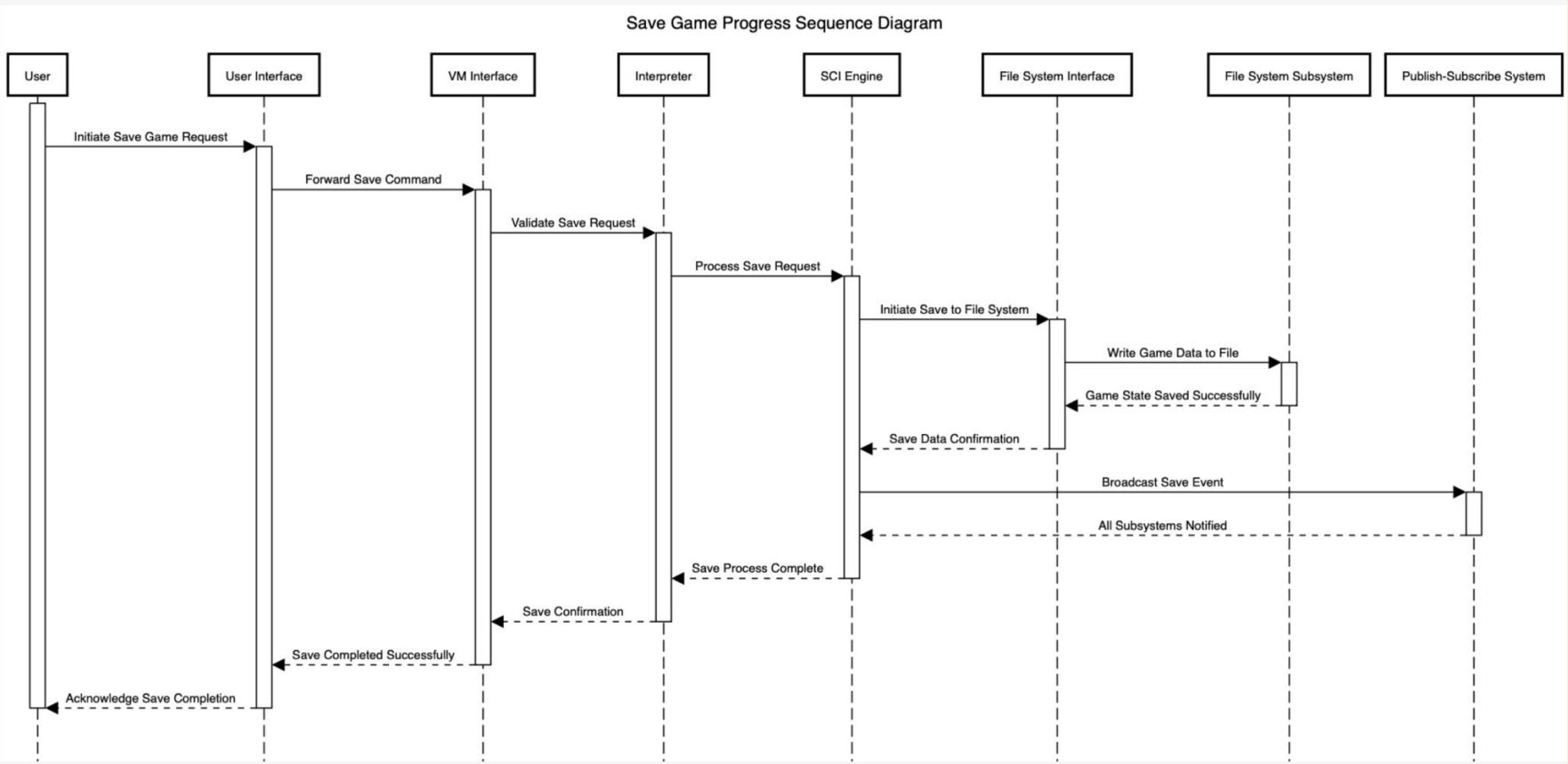
Division of Responsibilities

- Layered style supports independent work on different layers
- Game engine developers focus on interpreters
- Subsystem developers handle event-based architecture
- Summary:
  - Prompts specialization of developers in their area of expertise

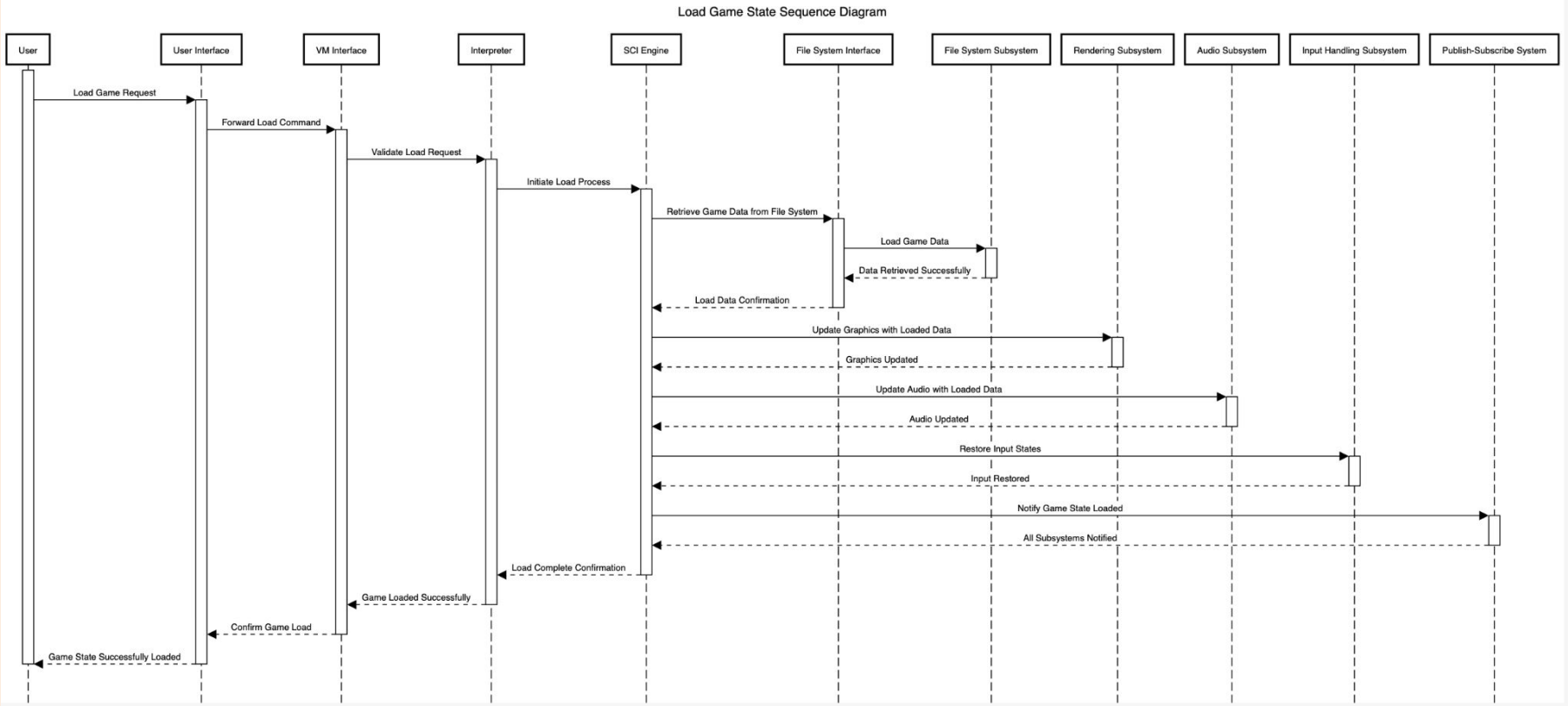
Use Cases

- We will look at two use cases:
  - Save operation
  - Load operation
- Diagrams depict sequences for saving and loading game states
- Key interactions are annotated

Save Game Use Case



Load Game Use Case



Lessons Learned

- **Modularity:** Simplifies maintenance and scaling
- **Performance Considerations:** Address early
- **Documentation:** Essential for development
- **Concurrency:** Requires careful design to ensure thread safety

Conclusion

- ScummVM uses layered, interpreter, and pub-sub styles
- Modular architecture enables cross-platform support
- Performance improvements will enhance capabilities for demanding games
- Plans for future enhancement: interpreter efficiency and event handling

Thank you