Fall 2024

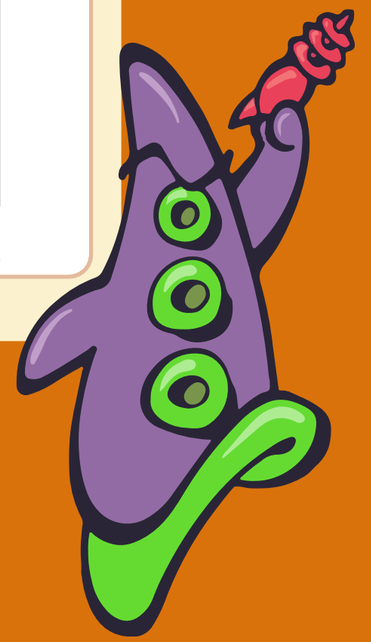
Conceptual Architecture

Oct 4, 2024

Team Null Terminators
CISC 322 A1
A Conceptual Overview of Scumm VM’s Architecture

https://youtu.be/PqZ8G_kK8U0

Explore Presentation...



Our Team

Hayden Jenkins

Presenter
Concurrency
Presentation Slides + Recording

Benjamin Leray

Presenter
Division of Responsibilities
Presentation Script

Ryan Van Drunen

Group Lead
Functionality + Diagrams

Simon Nair

Functionality + Diagrams

Jeremy Strachan

Evolution + Intro/Conclusion

Corey Mccann

Data Flow + External Interfaces

Derivation Process

Basic Components

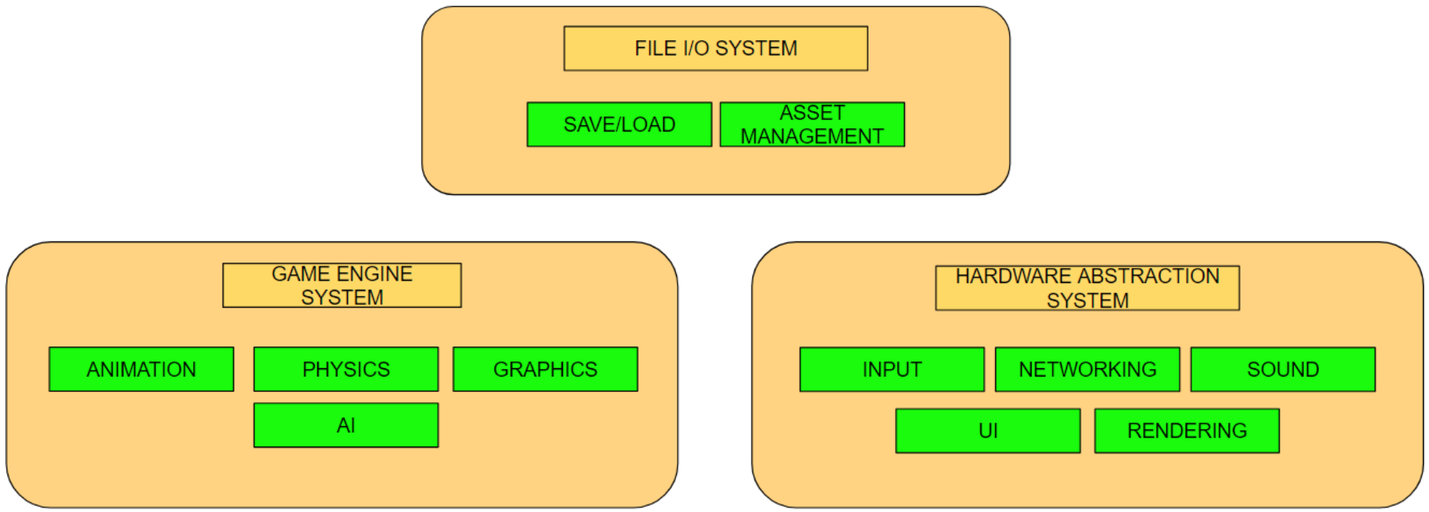
SOUND	GRAPHICS	INPUT
UI	GAME ENGINE	PHYSICS
ANIMATION	ASSET MANAGEMENT	SAVE/LOAD
RENDERING	NETWORKING	AI

Steps

- Utilized existing knowledge to hypothesize core components
- Focused on key systems
- Developed initial framework based on essential game component interactions.

Derivation Process

Initial Components and Subsystems

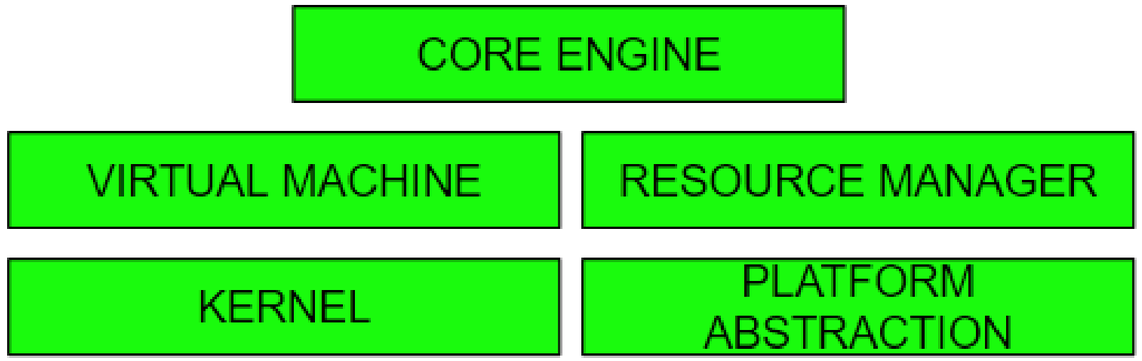


Steps

- Divided architecture into File I/O, Game Engine, and Hardware Abstraction systems
- Components work together to execute core functionalities efficiently
- Structured components using a layered architecture approach

Derivation Process

New Components Found In ScummVM

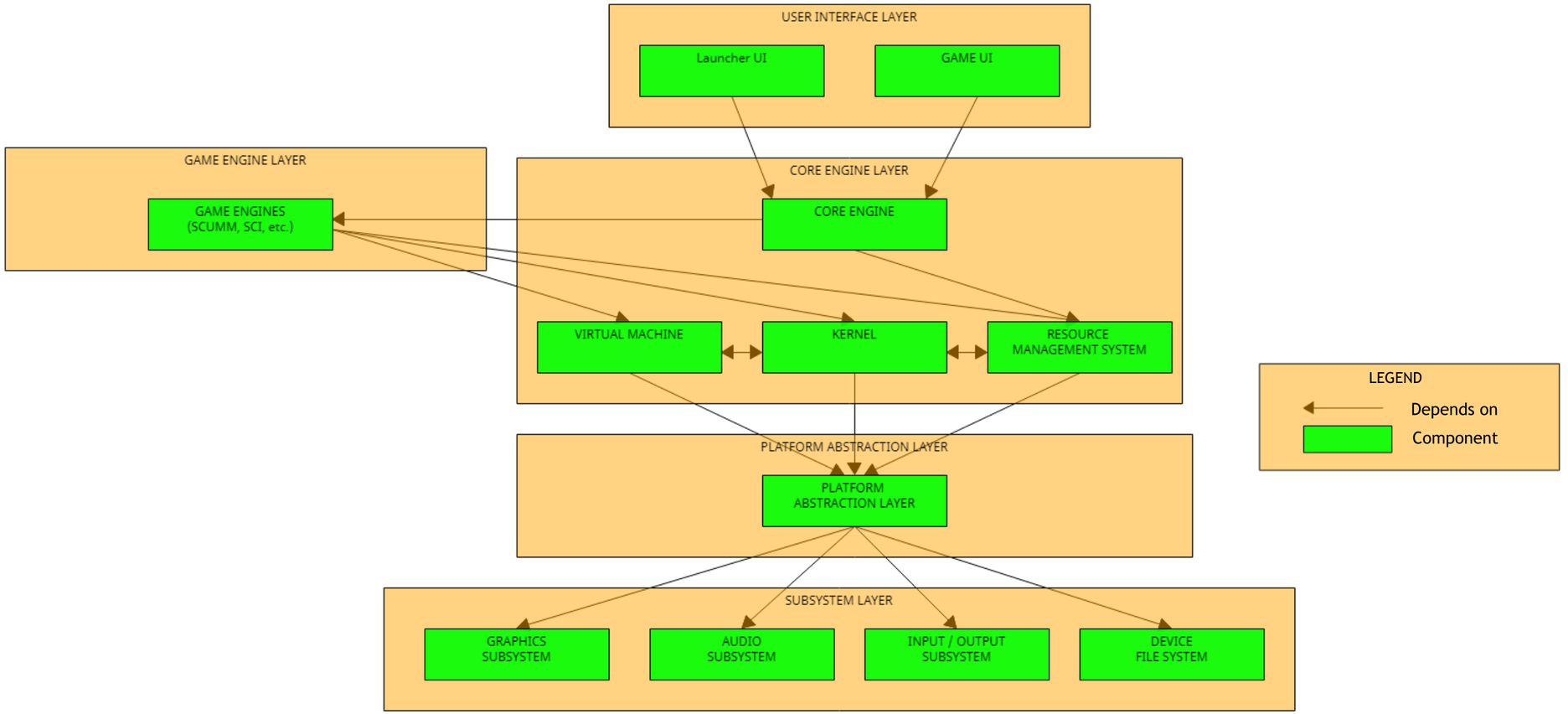


Steps

- Research revealed ScummVM as an abstraction layer
- Core Engine added to support execution across various game engines
- Platform Abstraction ensures compatibility with different operating systems.
- Resource Manager and Kernel manage system resources

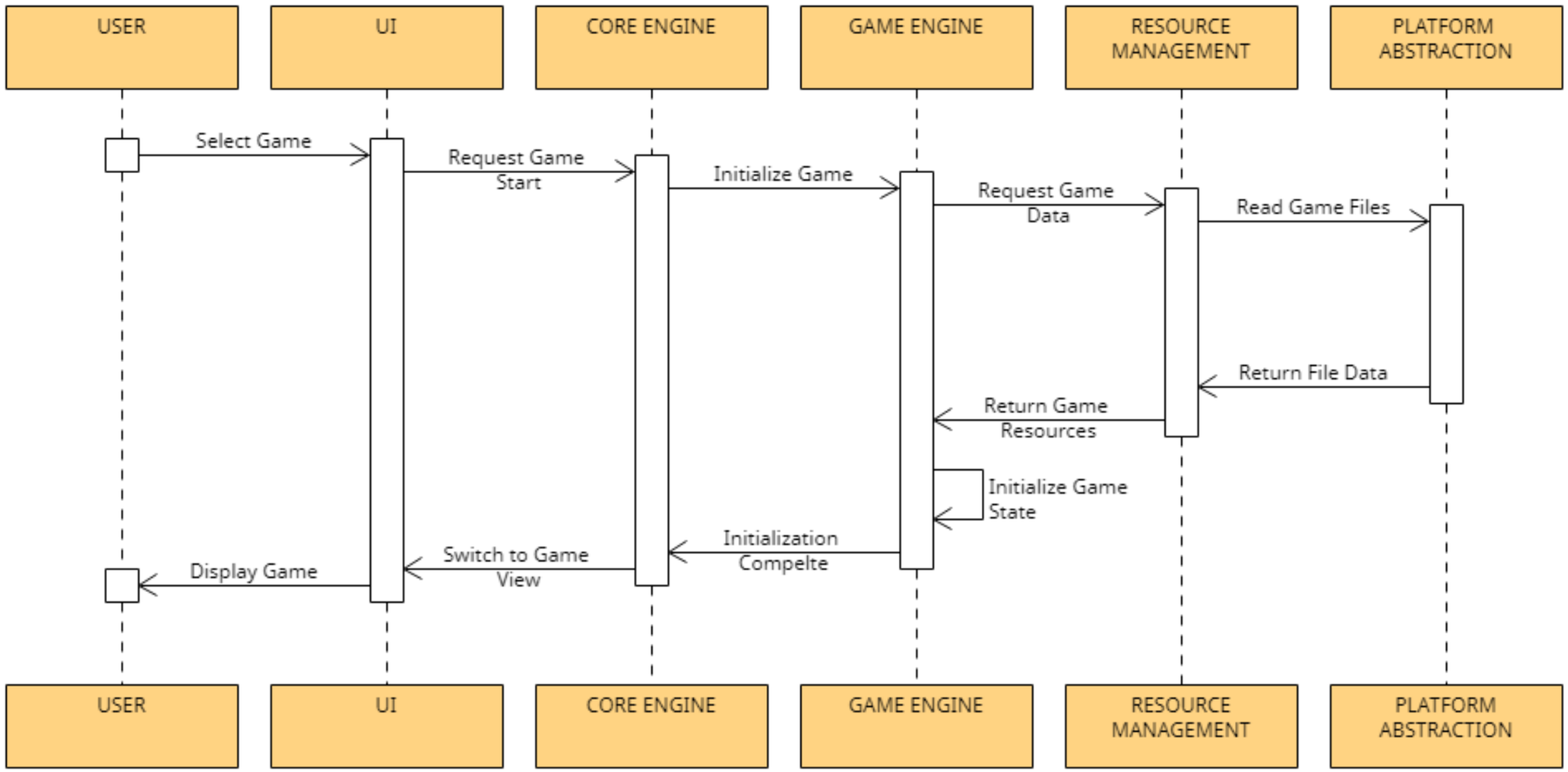
Conceptual Architecture

Complete Conceptual Architecture



Use Case

Game Startup



Concurrency

Concurrency Within ScummVM

- ScummVM primarily a single-threaded approach to maintain compatibility with classic games, but selectively uses concurrency for performance enhancement
- Audio subsystem operates on a separate thread, handling multiple streams to ensure continuous, stutter-free playback
- File I/O operations utilize concurrency to load large game assets (e.g., textures, sprites, audio) in the background, preventing pauses or interruptions in gameplay
- Audio subsystem operates on a separate thread, handling multiple streams to ensure continuous, stutter-free playback,
- User input is managed on a separate thread, allowing real-time processing of mouse clicks and keyboard actions, ensuring responsiveness even when the main game loop is busy
- SCUMM engine simulates concurrency through time-slicing, sequentially executing up to 20 scripts within the main loop, creating the appearance of parallel execution

Limitations and Lessons Learned

Limitations and Lessons Learned

- Limited documentation on ScummVM architecture required extensive research through forums and low level documentation
- Diverse game engines supported by ScummVM complicated the understanding of a unified architecture, requiring investigation into individual engines
- Time management challenges emerged as coordinating efforts among members with very busy schedules requiring careful planning and communication
- Frequent iterations and refinements of the architecture model were needed as new contributions were integrated.

Conclusion

Summary

- Identified ScummVM as an abstraction layer supporting multiple game engines
- Layered Architecture at a high level
- Object-oriented and layered at lower levels
- Discovered concurrency use in audio, input, and asset loading for performance