A HUD Map for Doom 3 BFG

Basement Gurus

Isaac Chan

Daniel Elmer

Matthew Sherar

Yohanna Gadelrab

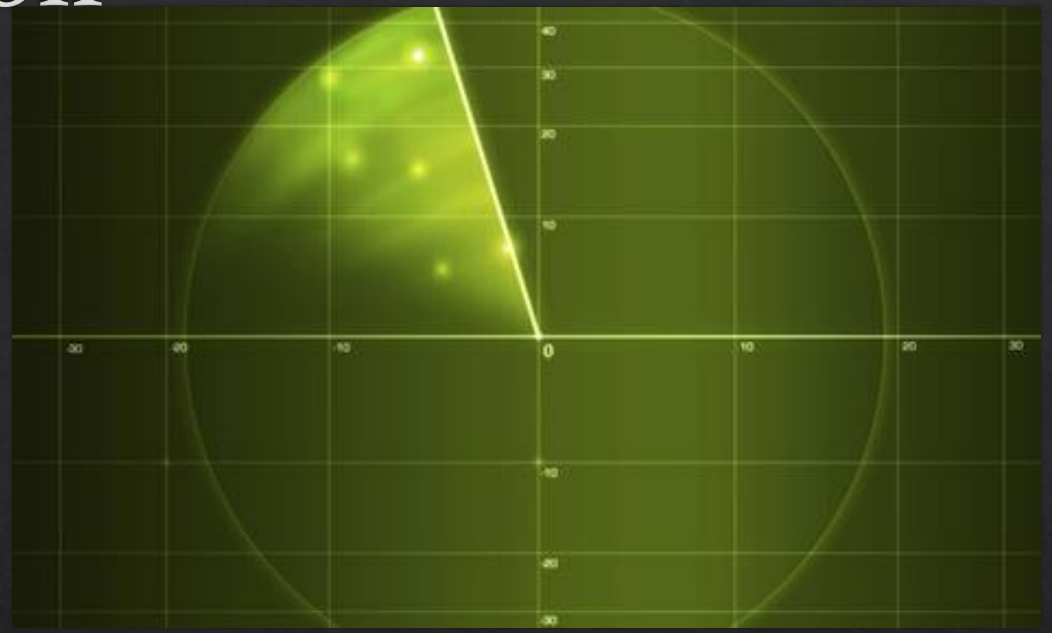
Dylan Liu

Kainoa Lloyd

Overview

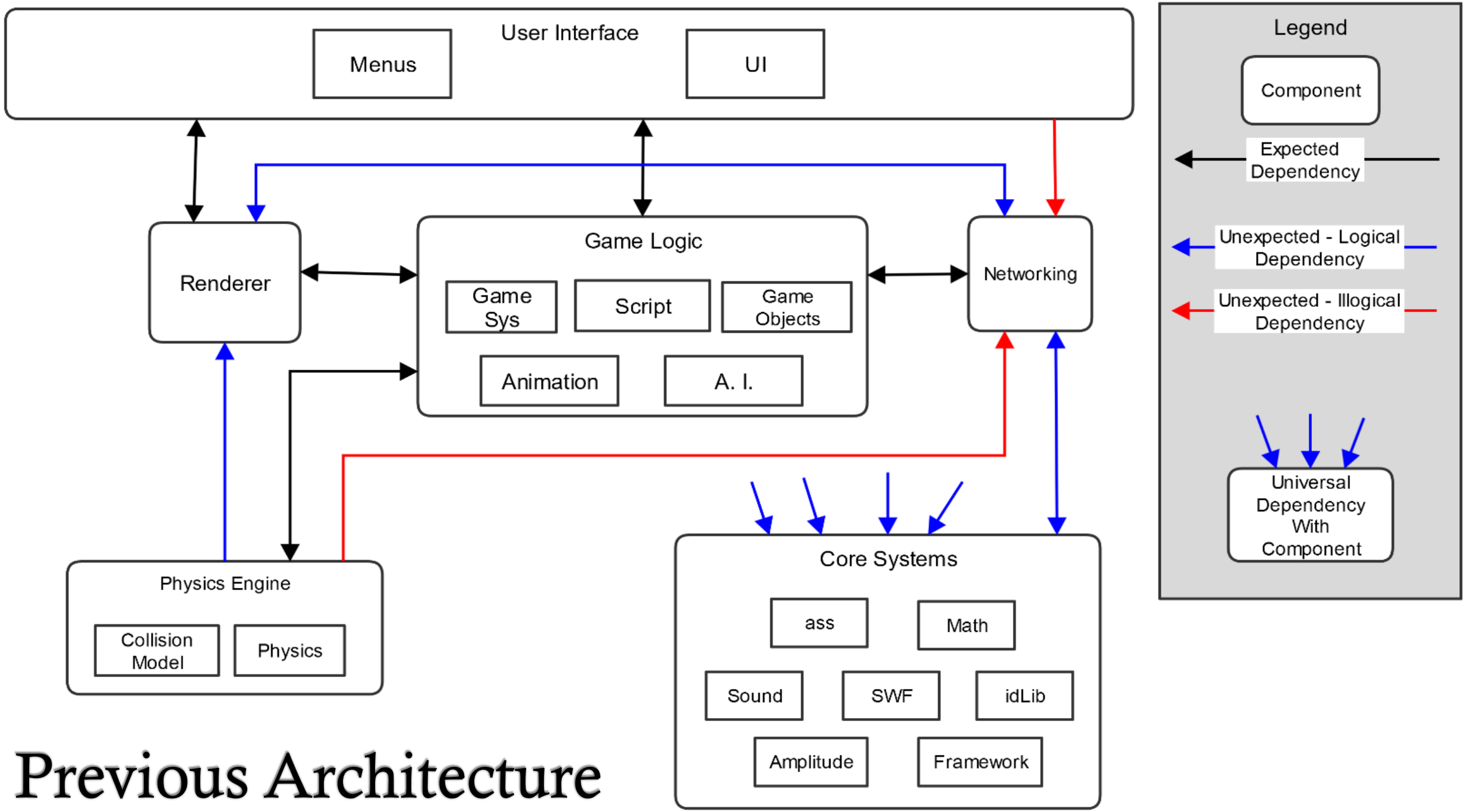
1. Introduction
2. Previous Architecture
3. Proposed Implementation
4. Alternative Implementation
5. SAAM Analysis
6. Impact on Architecture
8. Sequence Diagram
9. Testing
10. Concurrency
11. Lessons Learned
12. Limitations
13. Conclusion

Introduction

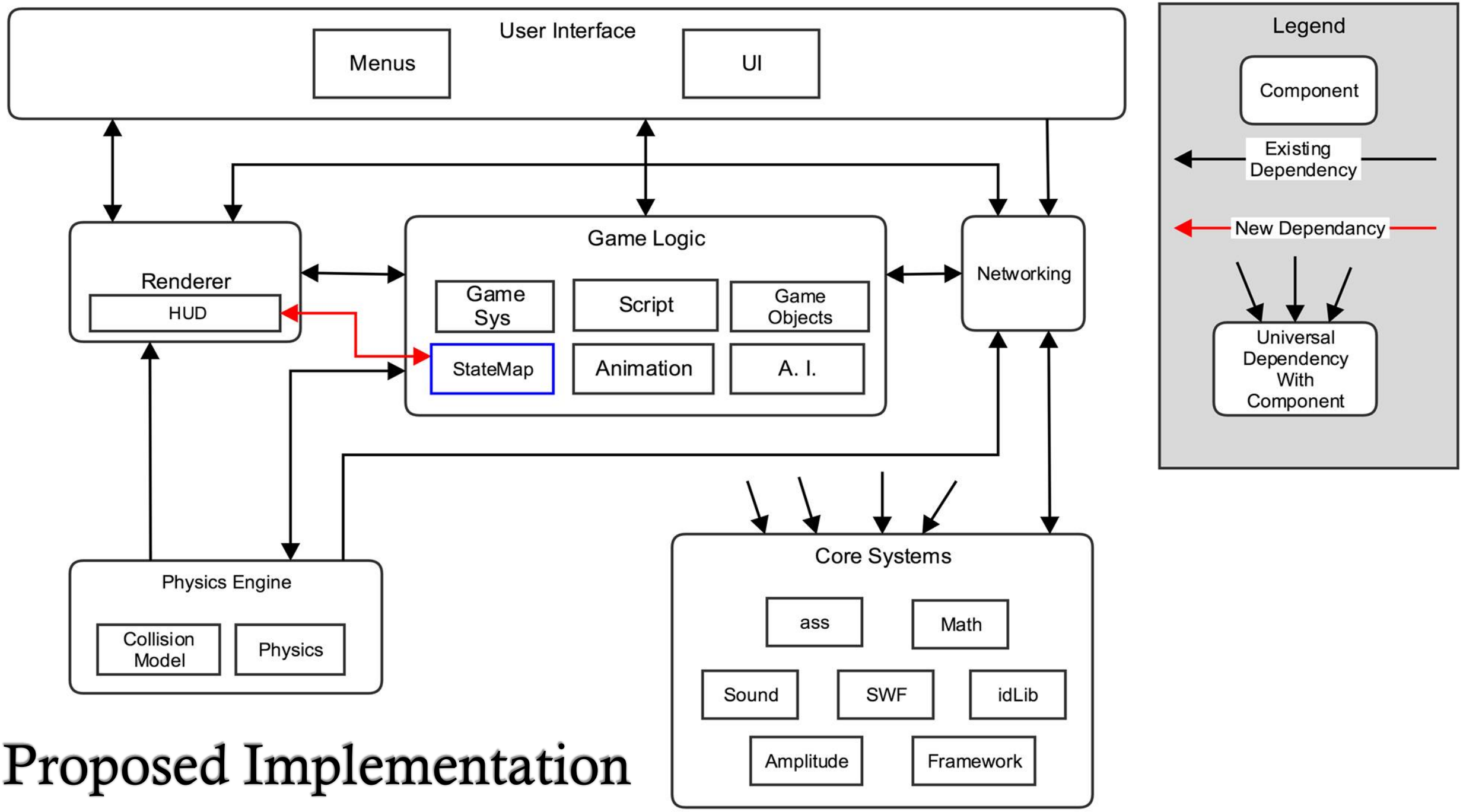


- Proposed feature is a radar display map
- SAAM analysis to two different implementation methods
- Impact on architecture

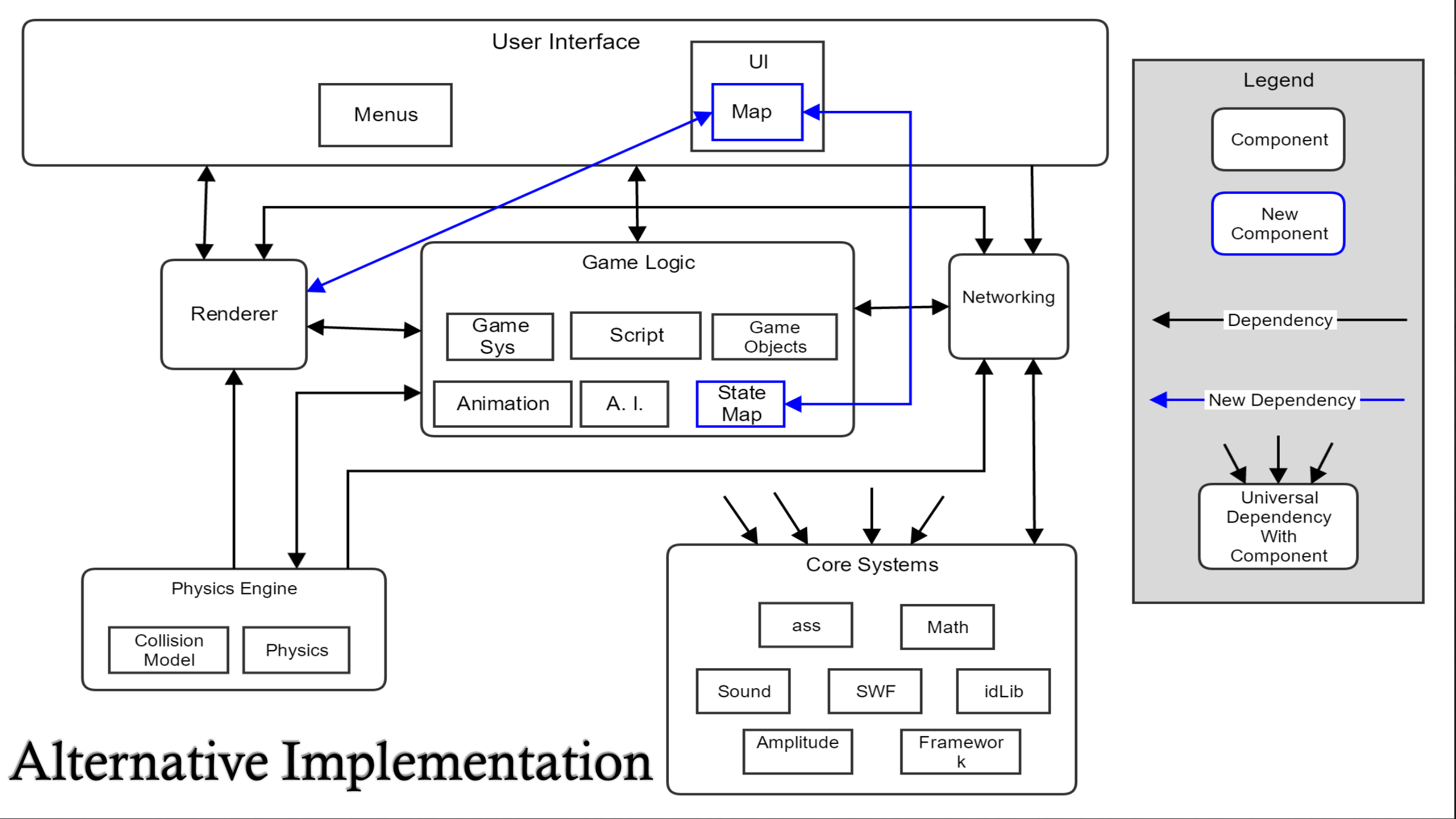

89
92
5
259
CFR JUNCTION 6



Previous Architecture



Proposed Implementation



Alternative Implementation

Software Architecture Analysis Method (SAAM)

Stakeholders

- End user, any players
- Developer

Candidate Implementations

- **Game Logic** highly coupled with **Renderer** for a simpler, but efficient geometric map
- Couples the interaction of **Game Logic** with **U.I** in order to use

Software Architecture Analysis Method (SAAM)

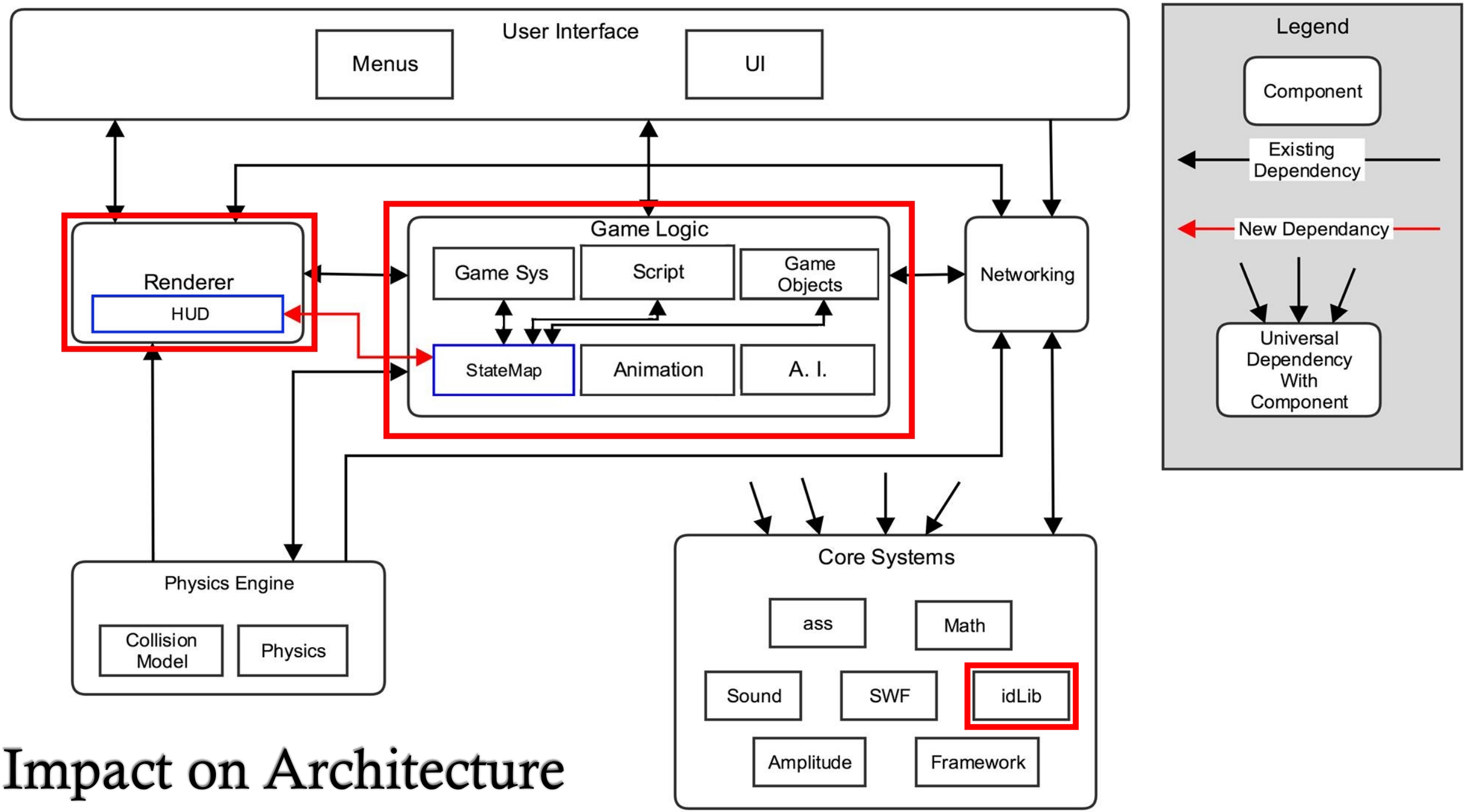
- Non-Functional Requirements (NFRs)
  - Performance
  - Availability
  - Integration
  - Modifiability
  - Testability
  - Resource Constraints

SAAM Proposed Implementation

Non-Functional Requirement	Implementation
Performance	Minimization of unnecessary coupling and maximization of cohesion between complementary subsystems ensures that performance is maximized.
Integration	Minimization of unnecessary coupling ensures ease of integration, as most modifications made will only affect the Game Logic and Renderer subsystems.
Resource Constraints	Minimap has to be updated in real time (i.e. frame by frame) and rendered in high quality - could potentially place stress on the system.

SAAM Alternative Implementation

Non-Functional Requirement	Implementation
Performance	• Potential slow down in performance due to more intensive processing
Integration	• Difficult to implement • Requires modification of the Game Logic and UI subsystems • High coupling in UI subsystem • Relies heavily on the UI classes
Resource Constraints	• Intensive on processing - Renderer • Constant communication between the Game Logic, UI, and Renderer subsystems to produce a more comprehensive map



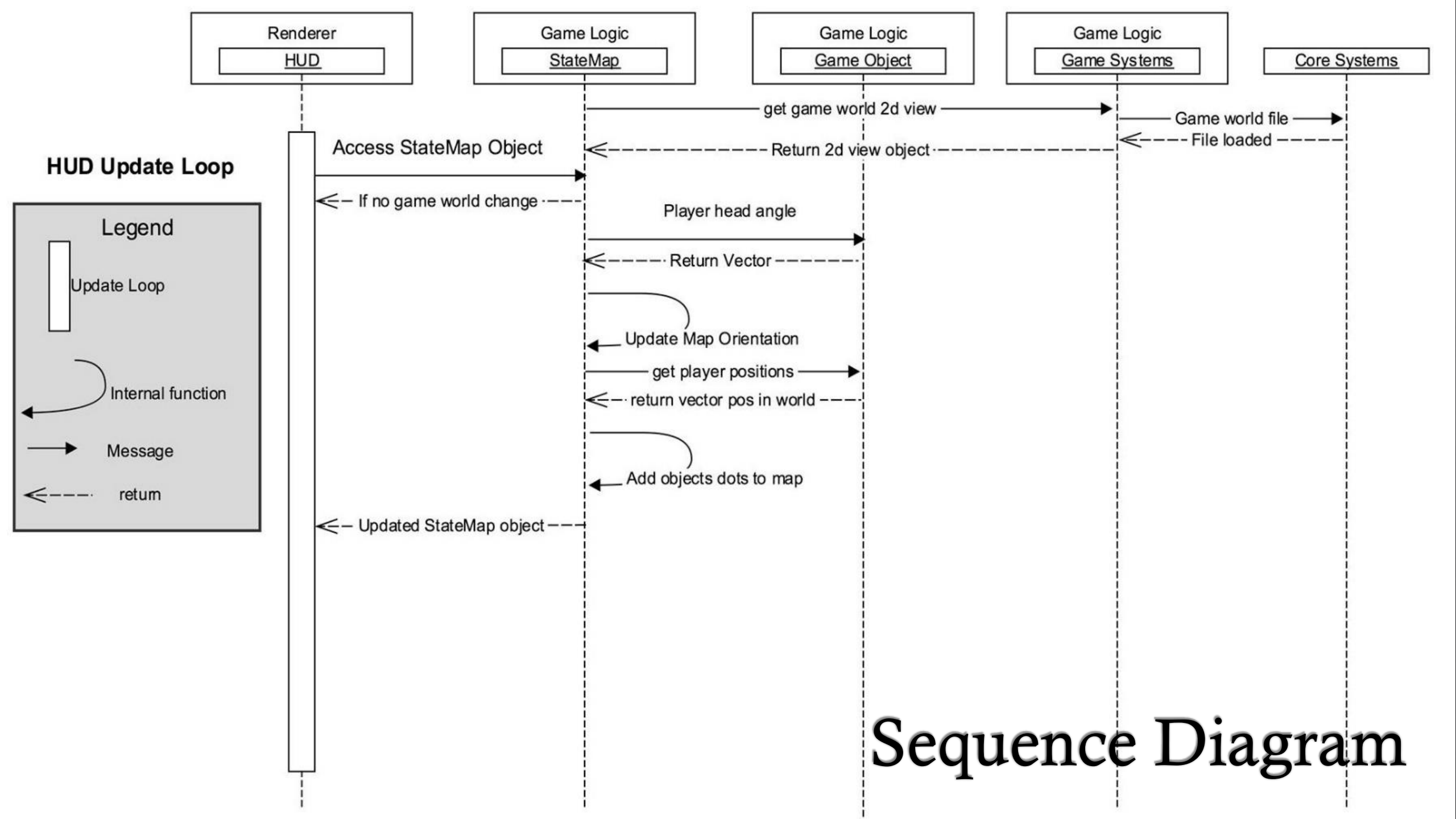
Impact on Architecture

Impact on Architecture

- **StateMap** will need to be highly coupled with the other subcomponents of **Game Logic** in order to obtain the necessary data to form the map. It will be dependent on the game scripts, the scripting engine, and the **Game System** scripts in particular.

- A new interface needs to be written into the Renderer in order to properly interface the received data from **StateMap** to draw the map in a similar way to how HUD elements are drawn.

- This implementation is simpler and is more efficient, and less intensive on the renderer. The simpler geometric map is still heavily reliant on the geometry libraries in the **Core Systems**.



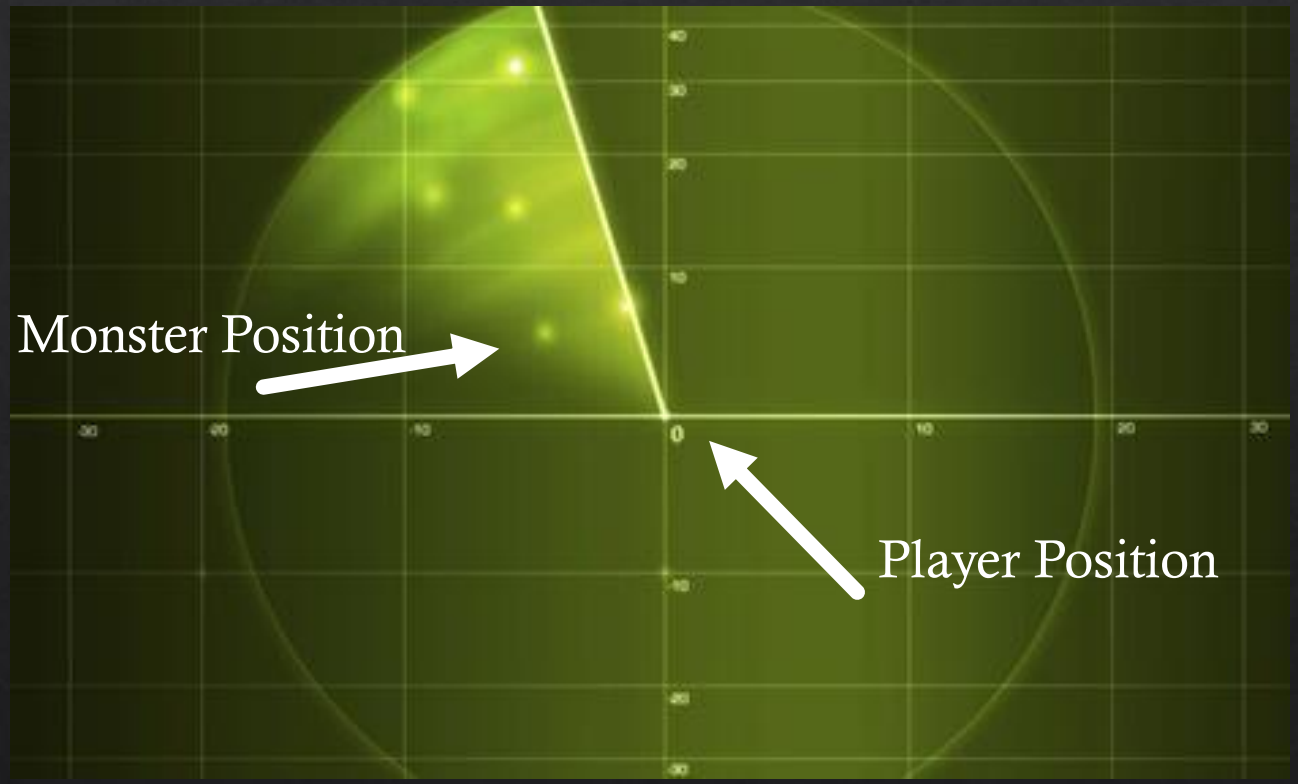
Sequence Diagram

Testing

- Subsystems to be tested:
  - Game Logic
  - Renderer
- Planned tests:
  - Game Logic: Correct generation of state map. Make sure enemies are in the correct position in the HUD map
  - Renderer: Refresh of map. Make sure game map is consistently updating

Concurrency

- Need one thread only
  - Prevent clogging of the game
  - Map state updated every few seconds



Lessons Learned

- How to conduct a SAAM analysis
- Knowledge of concrete architecture made it easier to conduct SAAM analysis and see what components would be impacted
- Object-Oriented Style architecture facilitated easier changes to the system minimizing the number of affected systems

Limitations of Chosen Implementation

- Our chosen implementation is not very portable. It requires a high amount of coupling with the Doom 3 BFG's script files and the required interface in the renderer.
- It also loses the benefits from using already written U.I code, which could help with flexibility and reuse in the system for other purposes.
- The map itself will need to be less detailed if we want to keep it as efficient as possible.

Conclusion

- Add radar type map to Doom 3.
- Chose a simpler implementation for the sake of efficiency, while sacrificing potential code reuse and aesthetic quality.
- Added a new subcomponent to Game Logic (State Map) and added an interface to the renderer that can interpret State Map's data and get the proper mathematical data from the Core System's Geometric Libraries.