Pangolin: A look at the conceptual Architecture of Super Tux Kart

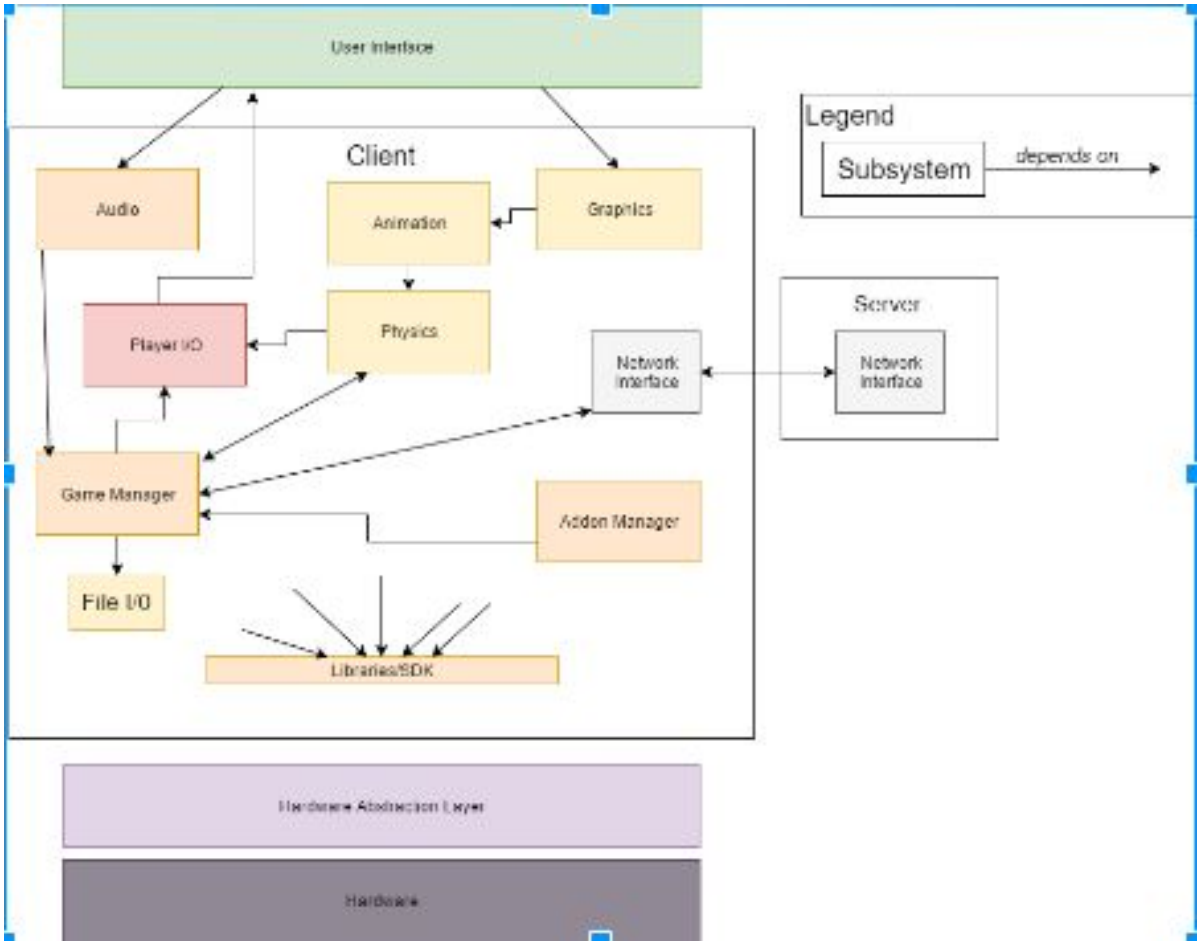
A CISC 326 Project by:
Mohammed Gasmallah
Russell Dawes
Caleb Aikens
Leonard Ha
Vincent Hung
Joseph Landy

Overview

- Architectural Style
- Conceptual Architecture
- Derivation Process
- Subsystems
- Use Case Diagram
- Sequence Diagram
- Considered Alternatives
- Concurrency
- Potential Development Team Issues
- Lessons Learned
- Conclusion

Previous Architectural Style Hypothesized

A layered style of architecture with possibility of including more client/server interactions as well.



Layered?

Using a layered architectural style gives us some huge advantages for developing:

- Easy separation of modular parts through layered logic
- Separation of team to work on different segments
- Helps link User Interface with Game Client and Hardware Abstraction Layer

Possible Client/Server

Super Tux Kart is a prime game to include some networking as networked races would be very popular.

Some talk of the implementation is being made and the Client/Server architecture is the preferred model.

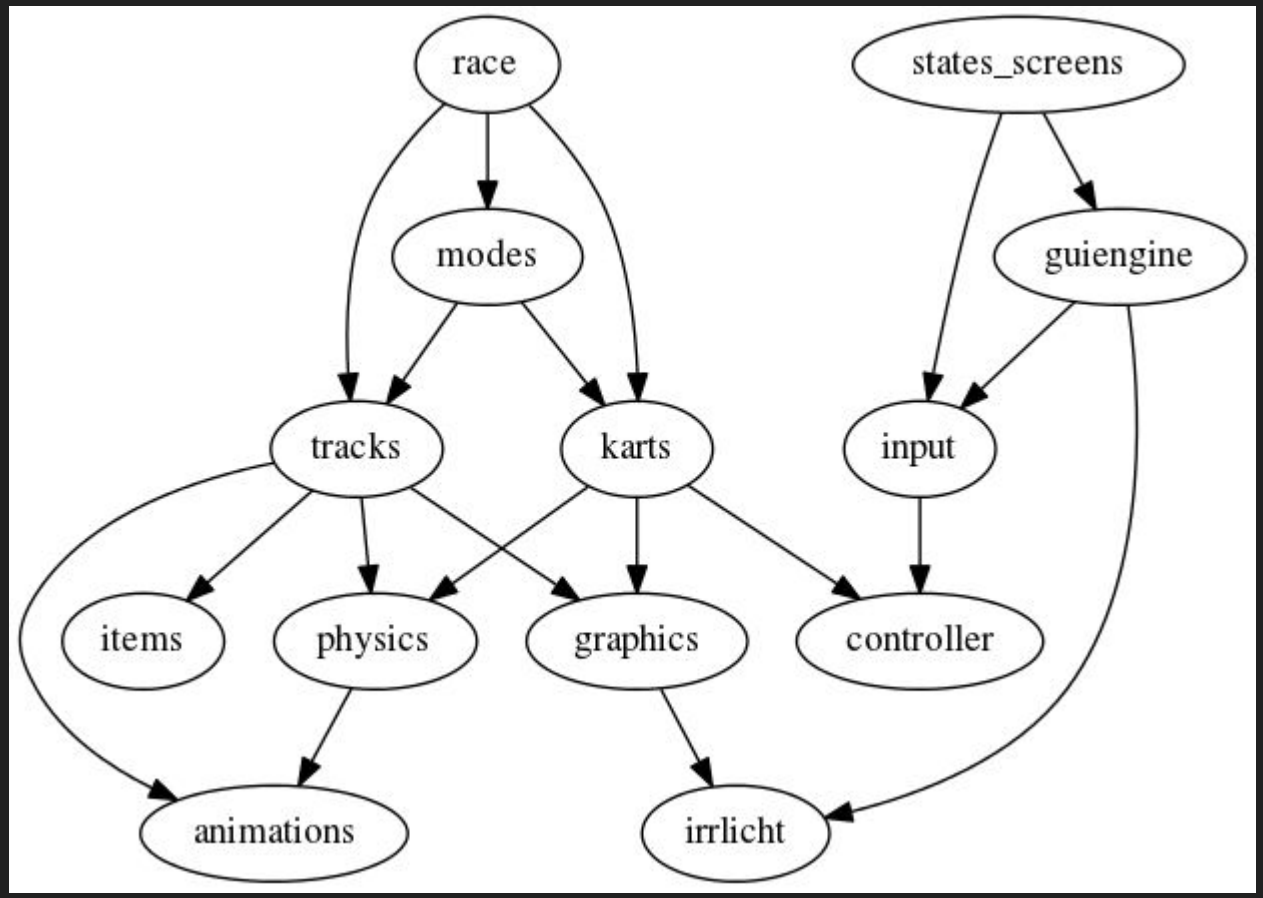
Security and accessibility are easily maintained with the client/server model.

Cheating becomes harder.

Centralization makes abstraction of processes easier.

Derivation Process

- Used high level module documentation as base of operations
- Compared possible architectures with reference game engine architecture
- Identify dependencies



Module interaction diagram from documentation

Subsystems

The subsystems in our conceptual architecture are:

- File I/O
- Addon Manager
- UI
- Audio
- Graphics
- Physics
- Game Manager
- Player I/O
- Libraries
- Network Interface

Subsystems cont.

- File I/O
  - Component the game uses to save data
    - High Scores
    - Story Progress
    - Replays
  - Is only dependent on the Game Manager
    - Anything that would be saved is information contained by the Game Manager
- Addon Manager
  - An internal part of the game that allows for browsing and installing of additional content
    - new tracks
    - Modes
    - karts
  - Is only dependent on the Game Manager, because the effects of the addons just alter values interpreted by the Game Manager

Subsystems cont.

- UI
  - The component through which the player interacts with the game
    - Menus
    - In Game Overlay
    - Sounds
    - Video
  - What the player uses to make decisions for future inputs
  - Is dependent on the Audio subsystem and the Graphics subsystem
    - Audio subsystem plays sounds triggered by UI elements
    - Graphics subsystem used to render UI elements to screen.
- Audio
  - Manages what sounds should be played and at what volume
  - Depends only on the Game Manager

Subsystems cont.

- Graphics
  - Handles rendering of content from Game Manager.
  - Wrapper over Irrlicht/Antarctica
    - Open source renderer for video games
    - Antarctica is a new renderer using some elements of Irrlicht, but with modern features
  - Adds additional functionality
    - Particle systems
    - Systems to simplify adding elements common in STK to the graphics context.
- Physics
  - Wrapper over Bullet
    - Open source physics library
    - Used by numerous games, engines and applications
  - Required by Game Manager to determine what the effects of game objects are upon each other through physics simulations.

Subsystems cont.

- Game Manager
  - Every action and reaction passes through the Game Manager which delegates the appropriate component to deal with it
  - Contains all active information about the current state of the game such as
    - Karts in play
    - What track the player is on
    - What mode is being played
    - Score/Position/Time
  - The Game Manager is affected by:
    - File I/O
    - Network Interface
    - Physics
    - Player I/O

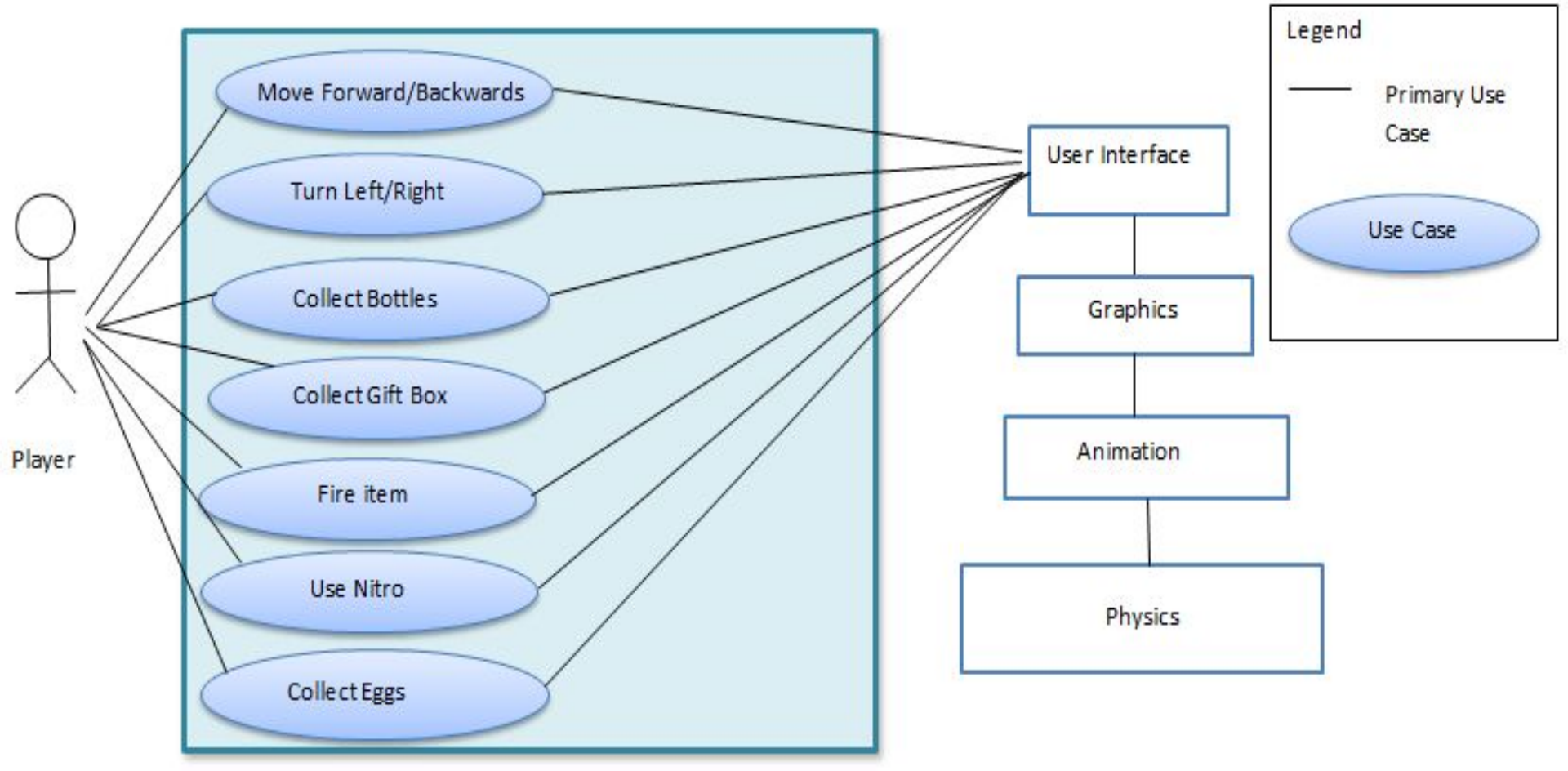
Subsystems cont.

- Player I/O
  - The players raw inputs are received by this subsystem and are translated into commands that the Game Manager will recognize
  - This component only relies on the UI for a range of allowed actions, such as whether or not the player is interacting with a menu or if they're in a race
  - As a more metaphorical link, the player's inputs will depend on what the player sees from the UI, such as turning if they are driving towards a cliff
- Libraries
  - Contains all of the assets for the programming of the game including
    - OpenGL
    - Bullet
    - Irrlicht
  - Also contains language dependencies
  - Almost all of the components in the Client are dependent on this subsystem

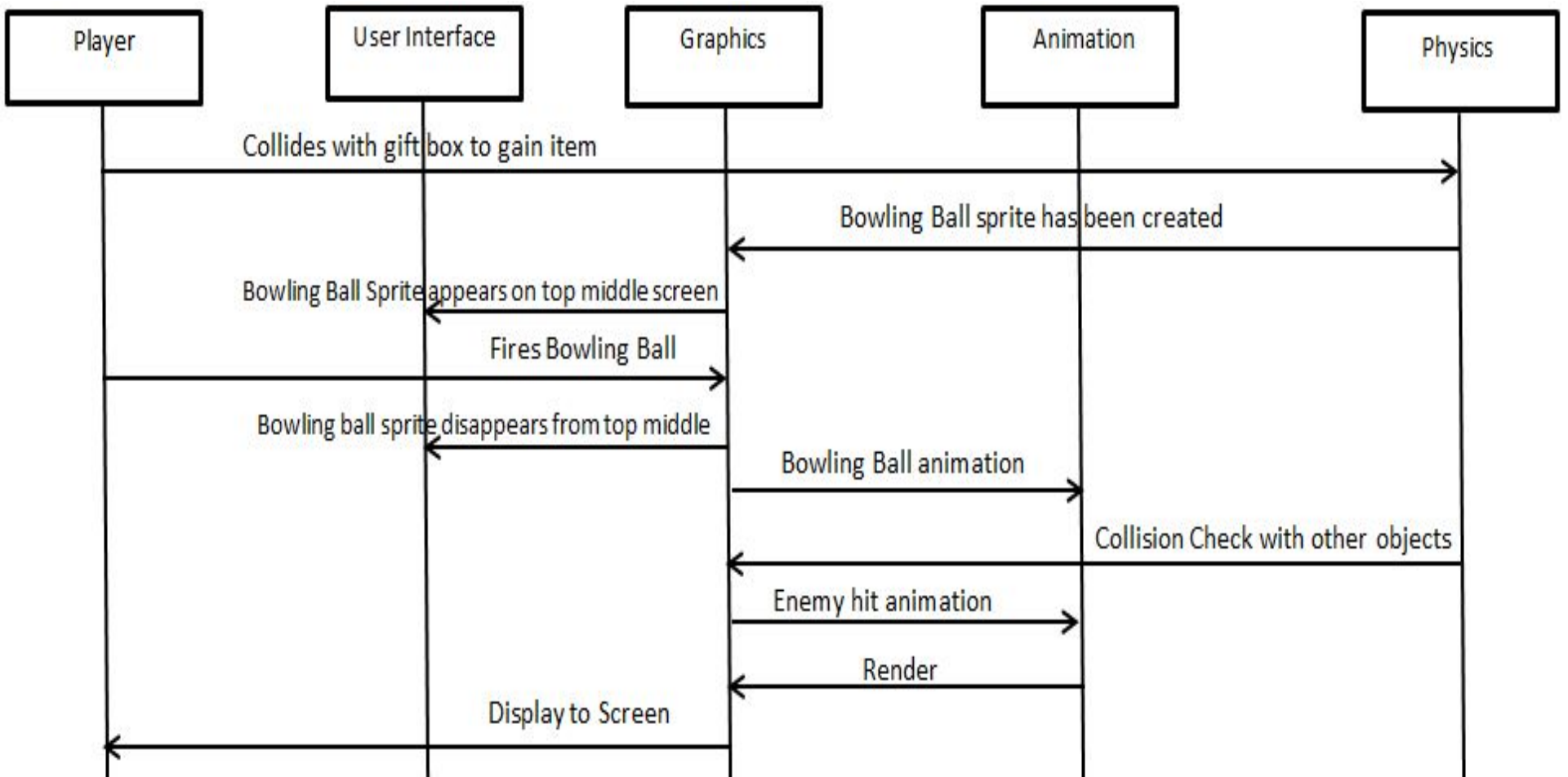
Subsystems cont.

- Network Interface
  - This is a mostly unused component at this time
  - This component takes one client's state of the game and reports it to the server
  - The server then sends back its understanding of the state of the game based on all of the reports from different clients
  - The local client adjusts the state of the game based on the new information
  - Currently the only network implementation that the game has is a friend system which allows you to send and accept friend requests, and to view your friends' profiles
  - This component has a dependency on the Game Manager, as the Game Manager contains the state that the Network Interface wants to broadcast

Use Case Diagram



Retrieving a Bowling Ball from a Gift Box and Hitting an Opponent with it



Alternatives?

Thought about using a primarily object oriented style, but settled on the fact that most likely a layered architecture was used as the overall architecture of the game engine.

The networking was thought for a while to be perhaps peer to peer because of certain naming conventions used in the documentation. This again was settled by the fact that the documentation seemed clear on the disadvantages of using a peer to peer style.

Concurrencies

From the Architecture, we can deduce the following concurrencies:

- Physics engine and Network Interface
- File I/O and Network Interface
- Player I/O and Network Interface
- Audio Engine and Graphics Engine

Development Team Issues

Team disagreements

- Although released in 2004, SuperTuxKart was unplayable and unsupported for two years.

Issues with switching 3D engines

In 2010, the game switched to Irrlicht Engine from PLIB which brought along many issues:

- A total rewrite of the GUI was needed
- Fans of the game were asked to test the game
- The entire process took approximately a year and a half - advancing technology was being developed as the engine was being implemented

Lessons Learned

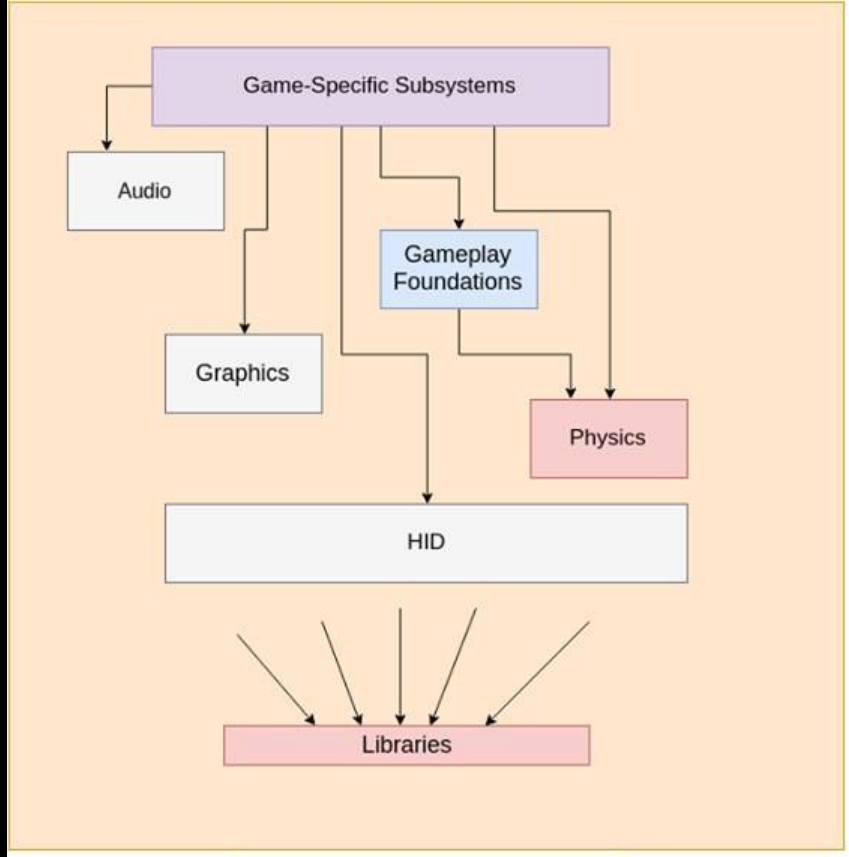
- Designing a Game Engine architecture is essential to mapping out possible issues in the future.
- Game Architecture requires deep understanding of architectural styles and their advantages/disadvantages and its subcomponents

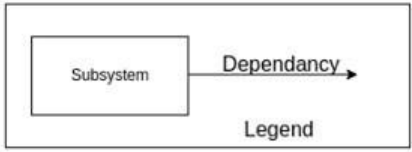
Conclusion

- Super Tux Kart may have suffered from lack of sufficient architectural planning (possibly due to the open source aspect of the project).
- Super Tux Kart runs a Layered style architecture with some Client/Server interactions.

Revised Architectural Style Hypothesized

A layered style objected oriented architecture





SOURCES

Gregory, J. (2009). *Game engine architecture*. Wellesley, Mass.: A K Peters.
Super Tux Kart Documentation: http://supertuxkart.sourceforge.net/doxygen/?title=doxygen
Super Tux Kart Source: https://github.com/supertuxkart/stk-code