Tuxedo Amigos

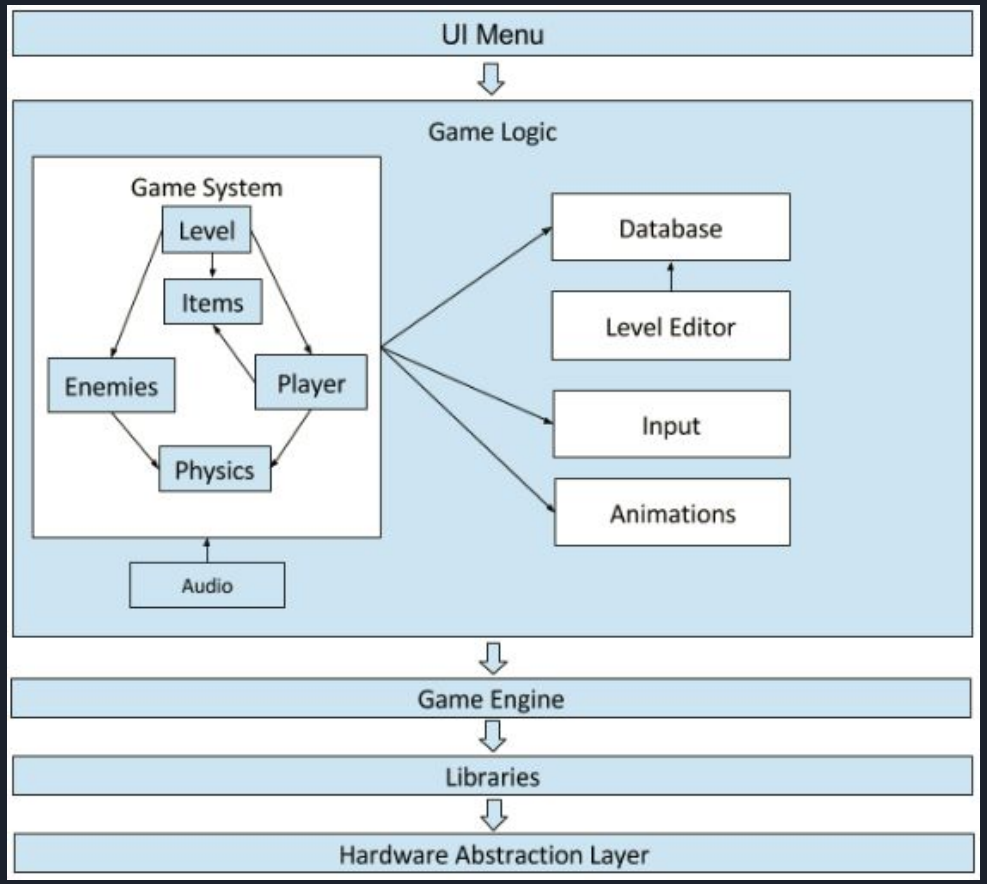
SuperTux Conceptual Architecture

Diana Balant, Matt Dixon, Ashley Drouillard,
Chris Gray, Sebastian Hoefert, Cesur Kavaslar

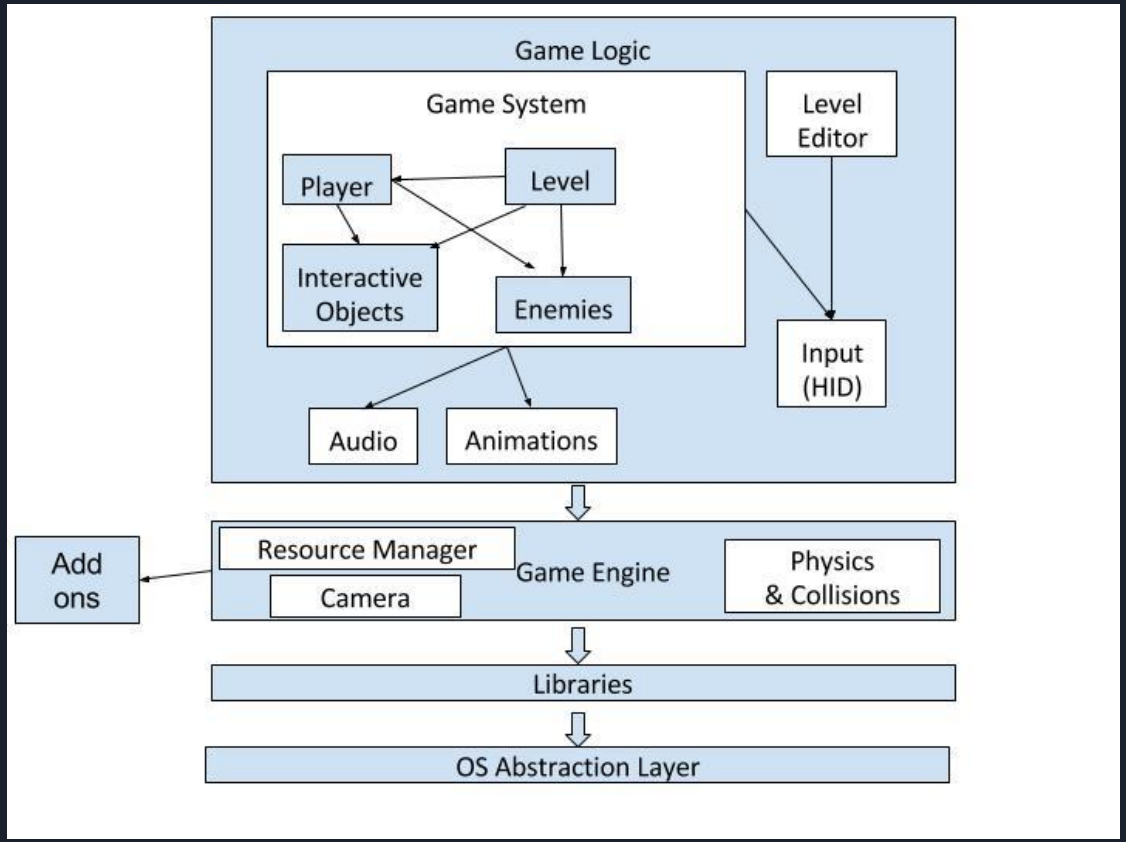
Overview

- Basic Architecture
- Component Breakdown
- Full Architecture
- Architecture Discussion (Pros & Cons)
- Use Case Diagram
- Sequence Diagram
- Division of Responsibilities
- Lessons Learned

Original Conceptual Architecture



Revised Conceptual Architecture



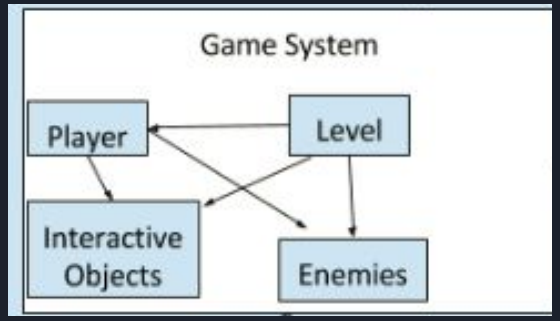
Component Break Down

1. Input (HID)
2. Game System
3. Level Editor
4. Audio
5. Animations
6. Add ons
7. Game Engine
8. Libraries
9. OS Abstraction Layer

Human Interface Device

- Keyboard, controller, etc.
- Tells the game what the player is inputting

Game System

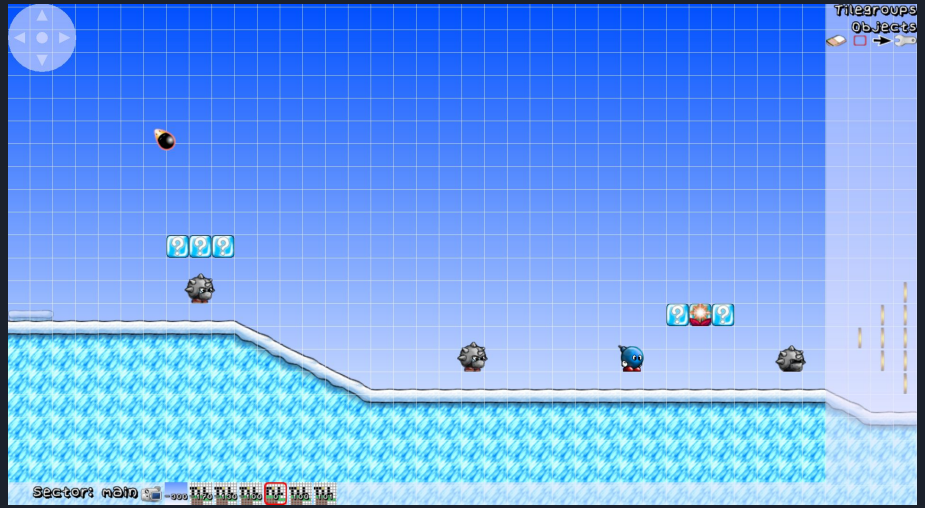


Broken up into separate components

- **Level**
  - What and where everything should be loaded
- **Player**
  - What the user controls
- **Interactables**
  - Coins, blocks, powerups, etc
- **Enemies**
  - Bad guys (Snowballs, Mr. Bomb, etc.)

Level Editor

- Allows the player to make or modify levels
- Uses in game assets



Audio

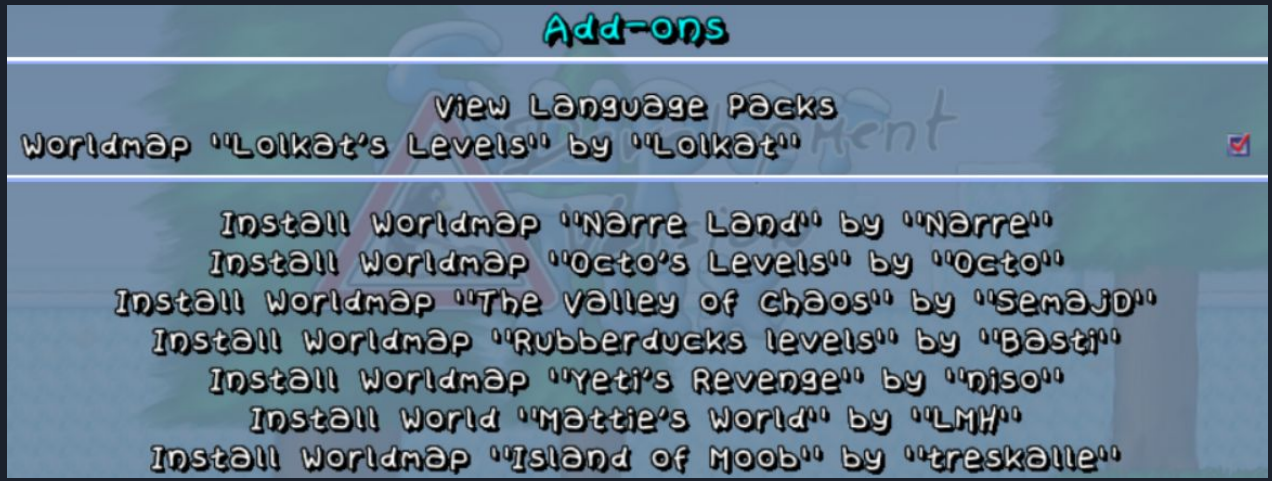
- Controller for all the audio in the game
- Invoked when needed

Animations

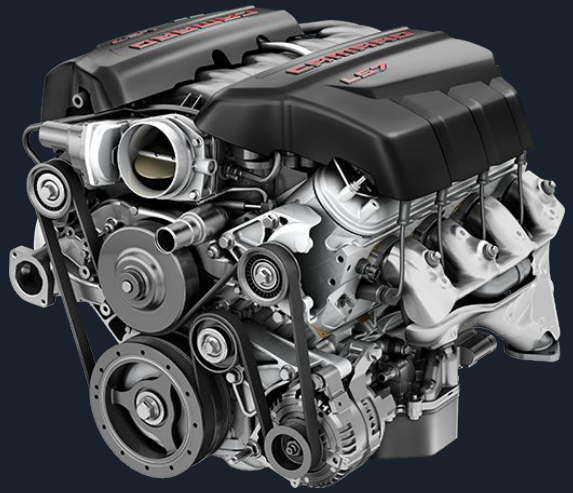
- Controls the animations and sprites of the game
- Only invoked when needed

Add-ons

- From online repository
- Additional user created levels
- Stored by resource manager



Game Engine

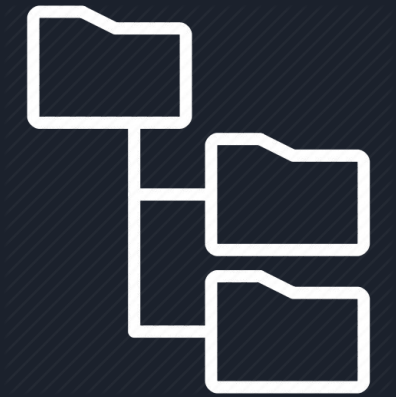


Broken down into a couple notable components

- Physics / Collisions
  - Collisions done by implicit invocation
  - Player registered for collision events
- Camera
- Resource manager
  - Has access to the game’s resources and assets

Libraries

- 3rd party code libraries
- Called from the game engine



Operating System Abstraction Layer

- Allows the game to be played on different platforms
  - Windows
  - Linux
  - Mac
  - Android

Architecture Discussions
(Pros and Cons)

Pros

- Able to change implementation without affecting clients.
- Systems designed as collections of autonomous interacting components
- Reuse on different implementations
- Changes to the function of one layer affects at most two other layers.

Cons

- Objects must know the identity of other objects in order to interact with other objects (method invocation).
- Not all systems can be designed with Layered Architecture Style

Use Case Diagram

Primary Actors

Player

Move (forward, backwards, slide)

Jump (up, back, smash)

Pause/Unpause

Change Settings in Options

Interact with Items

Download add-ons

Interact with Enemies

Select Level

Load Level

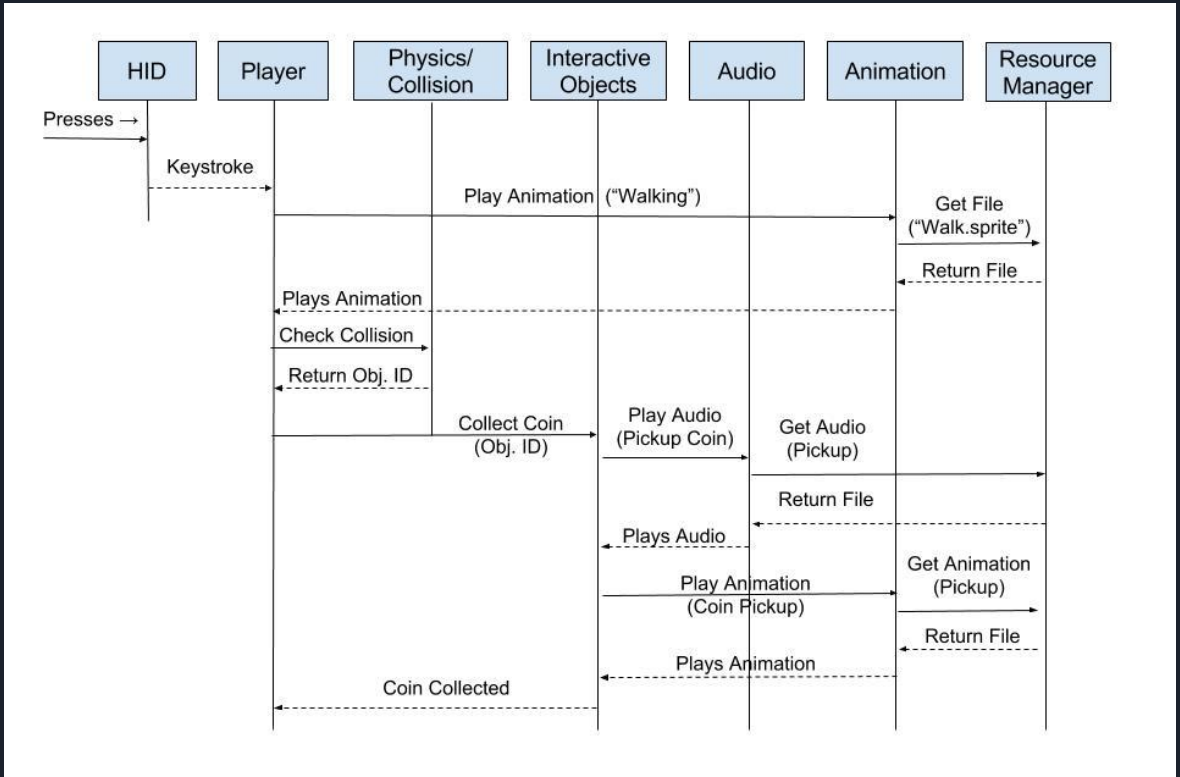
Create Level

Secondary Actors

Online Level Repository

Sequence Diagram

"Tux moves right and picks up a coin"



Division of Responsibilities

- Level editor/designer
- Lead Developer
- Program developers
- Artist
- Animators
- Audio

- Team of 6
- Open Development

Lessons Learned

- Don’t take too much time figuring out the architecture
- Use sequence diagrams to confirm dependencies
- No ‘perfect’ architecture

Thank you!

Any Questions?