



AURA Game Engine

By Robert Esposito and Anton Zeitger

Architecture

AURA Game Engine

Discussion Points:

- 2D game engine, in C++ and using SFML Library
- Component Architecture implementation
- SceneGraph, GameObjManager, GameObj, cBaseComponent
- Aura:: Static Library implementation
- XML example

SCENE GRAPH

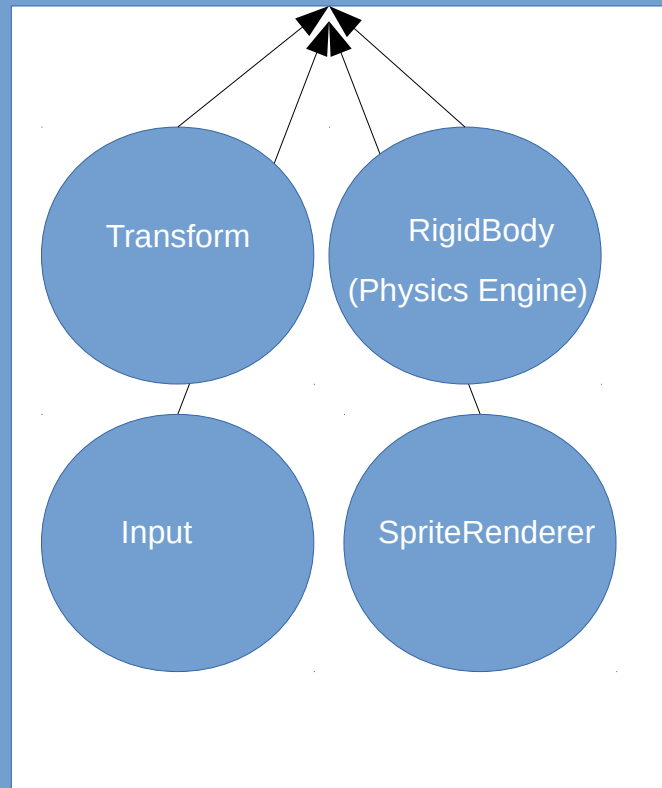
GAME OBJECT MANAGER

GAME OBJECT

Base Class

cBaseComponent

Sub Classes



Game Engine

The logo for Aurora Game Engine features the word "Aurora" in a large, white, stylized font. To the left of the text is a vertical plume of colorful smoke or nebula in shades of purple, blue, and green. To the right of "Aurora" is the text "Game Engine" in a smaller, white, sans-serif font.

Aurora Game Engine

A screenshot from the game "Lost Galaxies" showing a night sky with the Milky Way galaxy visible through the silhouettes of evergreen trees. The scene is framed by a thick red border.

Lost Galaxies

Aurora Game Engine

Our Game
Project



AURA

Game Engine

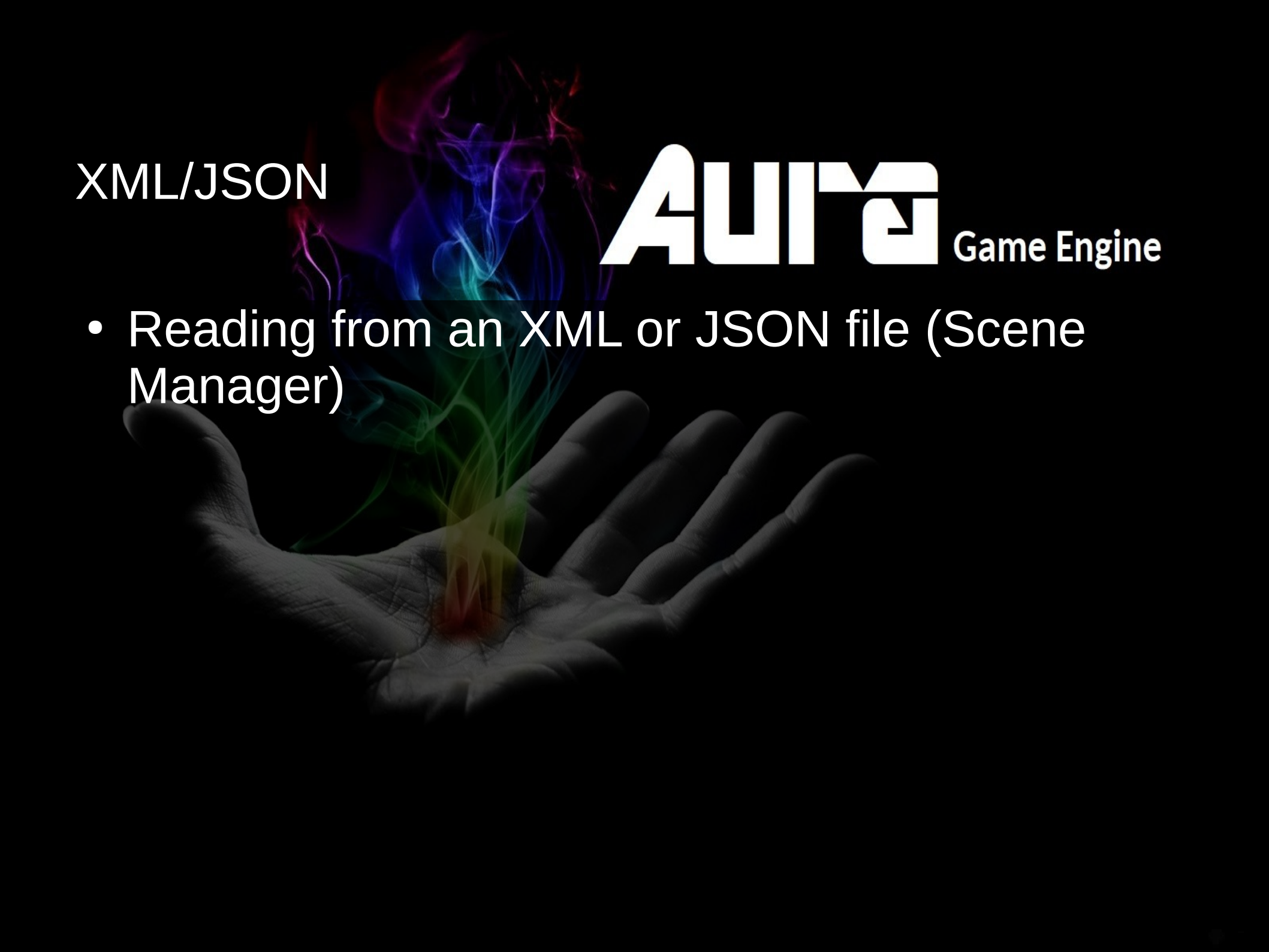
Lost Galaxies

- * 2D Single player game
- * Asteroid type game, avoid and shoot asteroids to win
- * Mechanics: 2D physics (collisions, transform, input controls, sprite renderer)

LIBRARY SETUP



- Aura::
- Static Library that separates the engine from the game, allowing for re-use of code and multiple games in a VS solution
- Setting up the Static Library via the project settings
- Creating a namespace that encompasses all functions and classes in Aura:: library
- Linker settings to connect projects to the library

A hand is shown from the palm side, holding a vibrant, multi-colored flame. The flame has a base of red and orange, transitioning through yellow, green, and blue to a top of purple and pink. The background is solid black.

XML/JSON

AURA Game Engine

- Reading from an XML or JSON file (Scene Manager)



AURA

Game Engine

Conclusion:

- **Aura Engine** exemplifies the usefulness of the component architecture design pattern and the power of static libraries to reuse code and make your projects organized and expandable.
- Future goals: new components, physics engine, 3D environment, optimization.