Technical Design Document

csd2400f24-csd2401f24-uxg2400f24-daa2402f24

Fall 2024

-**Group Members**-

*Amanda LEOW*

*Product Manager*

SAW Hui Shan

Programmer

Lilianna HANAWARDANI

Wai Lwin THIT

Table of Contents

[Overview 3](#_Toc177413241)

[Manager Base Class 4](#_Toc177413242)

[Game Management Manager 4](#_Toc177413243)

[Entity Component Manager (ECS) 4](#_Toc177413244)

[Log Manager 5](#_Toc177413245)

[ECS Components 6](#_Toc177413246)

[Entity 6](#_Toc177413247)

[Component Base Class 6](#_Toc177413248)

[System Base Class 6](#_Toc177413249)

[Naming Conventions 7](#_Toc177413250)

[Coding Documentation 7](#_Toc177413251)

[GitHub tips and rules to remember: 7](#_Toc177413252)

# Overview

The Lack of Oxygen (LOF) game engine is a C++ based game development framework that utilizes an Entity Component System (ECS) architecture. The project is designed with modularity and efficiency in mind, employing various managers to handle different aspects of the game engine.

LOF Classes

Managers

* Manager Base Class
  + Base class for engine managers
  + Managers implements the Singleton pattern for global access
* Game Management Manager
  + Manager of the game loop
* Entity Component Manager (ECS)
  + Manager of ECS components and systems
* Log Manager
  + Manager of the logfile

ECS

* Entity
  + Each entity has a unique identifier and a component mask
  + Represents a single game object
* Component Base Class
  + Base class for components
* System Base Class
  + Base class for systems
* Movement System
  + Class for Movement System

Utility

* Clock
  + Provides precise time measurements and sleep functionality

### Manager Base Class

The Manager class serves as the foundation for all specialized managers in the LOF engine. It provides common functionality such as startup, shutdown, and status tracking. All other managers in the system should inherit from this base class.

Key features:

* Abstract base class for all engine managers
* Provides virtual methods for startup and shutdown
* Tracks the started/stopped state of the manager
* Allows type identification for derived manager classes

### Game Management Manager

The Game\_Manager class is responsible for coordinating the high-level game flow. It manages the game loop, tracks game state (such as whether the game is over), and coordinates updates to other systems like the ECS.

Key features:

* Manages the main game loop and step count
* Controls the game over state
* Coordinates with other managers (e.g., ECS\_Manager, Log\_Manager)

### Entity Component Manager (ECS)

The ECS\_Manager class is the core of the LOF engine's ECS implementation. It handles the creation and management of entities, components, and systems, providing a flexible and efficient framework for game object composition and behavior.

Key features:

* Manages entity creation and destruction
* Handles component registration, addition, and removal
* Coordinates system updates
* Provides efficient entity querying based on component composition

### Log Manager

The Log\_Manager class provides a centralized logging system for the LOF engine. It allows for writing formatted log messages to a file, which is crucial for debugging, error tracking, and monitoring the engine's behaviour.

Key features:

* Implements the Singleton pattern for global access
* Manages log file creation and writing
* Supports formatted log messages
* Allows for customizable log file names
* Provides options for immediate flushing of log messages

## ECS Components

### Entity

The Entity class is a fundamental part of the ECS, representing any object in the game world. Each Entity is essentially just an ID with an associated mask indicating which components it possesses.

Key features:

* Unique identifier for each game object
* Component mask for efficient component possession checking
* Methods for adding, removing, and checking components

### Component Base Class

The Component class serves as the base for all specific component types in the LOF engine. Components represent the data and attributes of entities.

Key features:

* Abstract base class for all component types
* Allows for type-safe component management in the ECS

### System Base Class

The System class is the base for all system types in the LOF engine. Systems are responsible for implementing game logic that operates on entities with specific component compositions.

Key features:

* Abstract base class for all system types
* Defines the interface for system updates

# Naming Conventions

1. Classes

* Classes always begin with an upper-case letter (e.g., Manager).
* Classes with more then one word (e.g., Log\_Manager).

1. Functions

* Functions use lower case (e.g., distance()).
* Functions with more then one word (e.g., is\_started()).

1. Variables

* Functions use lower case (e.g., length).
* Functions with more then one word (e.g., square\_length).

1. Files

* Files always begin with an upper-case letter (e.g., Main.cpp).
* Files with more than one word (e.g., Shader\_Object.cpp).

# Coding Documentation

1. Doxygen-style comments for classes, methods, and files
2. Inline comments for complex logic

# GitHub tips and rules to remember:

1. Pull before pushing anything to avoid conflicts.
2. Create your own branch to perform your code separate from the MAIN branch.
3. Merge MAIN branch regularly into your branch to avoid having to deal with a lot of conflicts upon merging your branch into MAIN branch.
4. Be mindful of pushing any code to MAIN branch, you should inform teammates prior to pushing.