ClayEngine

An MMO Engine by Elideus

Welcome to the Clay Engine, a next generation game engine built from the ground up to be the perfect tool for the development of massively multiplayer online games with support for high end transactional database back end services and thousands of simultaneous users on the same server.

# Concepts

The ClayEngine uses some innovative concepts so it is important to understand the manner in which different services and objects are instantiated. The engine supports running multiple clients and server objects within one single process. This is designed to facilitate debugging network scenarios more easily. When you start up the engine each client or server will start in its own thread which is identified by an Affinity (or thread) ID. There is one core Services class that is used as a kind of look up registry for services that are used by each client. The services are divided by an Affinity of the parent object for that client or server.

## Startup Sequence

#### wWinMain() -> ClayEngineEntryPoint()

* This class should load the JSON clayengine.json file to read process configuration.

#### ForEach(startup) -> ClayEngineClient

* This class will create a client thread using the ClayEngineClientEntryPoint which will determine the thread affinity for all threads associated with that client.
* ClayEngineClientEntryPoint will load the client configuration file, which defines the services that client will run such as RenderSystem, NetworkSystem, etc.

#### ForEach(startup) -> ClayEngineServer

* This class will create a GUI server thread using the ClayEngineServerEntryPoint which will determine the thread affinity for all threads associated with that server.
* ClayEngineServerEntryPoint will load the server configuration file, which defines the services that the server will run such as RenderSystem, GameSystem, NetworkSystem, etc.

#### ForEach(startup) -> ClayEngineHeadless

* This class will create a server thread using the ClayEngineHeadlessEntryPoint which will determine the thread affinity for all threads associated with that server.
* ClayEngineHeadlessEntryPoint will load the headless configuration file, which defines the services that the server will run such as NetworkSystem, BackendSystem, etc.

## System Definitions

### WindowSystem

This provides a window context object with callback hooks for various window status events from the OS. Each window is thread safe and requires you provide an Affinity ID to instantiate.

### InputSystem

This provides an input handling object with string buffer objects to be used by a basic command console.

### RenderSystem

This builds up a DirectX 11 compatible device and device context to which DirectX 11 API calls can be made using the DirectX Toolkit. That is to say, it provides DX11 devices compatible with DirectXTK 11.

### ContentSystem

This object is used to load up spritefonts, textures, models, and any other binary data that can be accessed for use in the rendering pipeline.

### NetworkSystem

This system will create a basic listen server and a simple NBIO client to connect to it.