Technical Design Overview

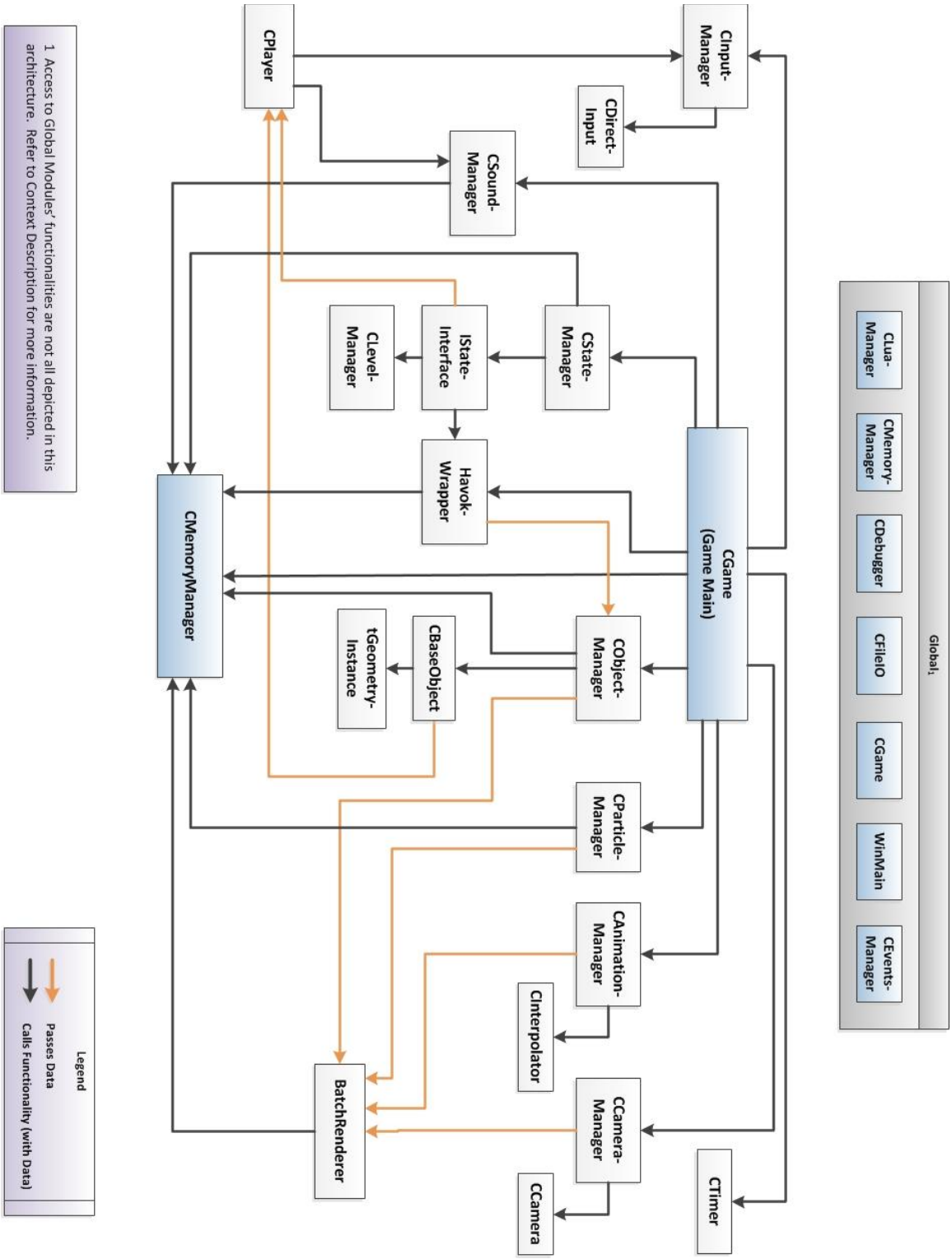
The purpose of the Technical Design is to be a reference guide for the programmers throughout the development process of the game. The programmers will use the Technical Design to:

- Realize the scope of the game
- Reduce any design errors that may arise
- Reduce any confusion about module interactions
- Understand the coding standards
- Know how to manage development problems

The Technical Design is comprised of:

- System Architecture flow chart – provides layout of game modules
- Context Model Description – describes each module and their interactions
- Module Breakdowns – describes purpose and interface of every module
- Memory Map – provides a breakdown of how much memory out game will be using
- Code Review Plan – will be used to keep code clean and well-commented
- Integration Plan – will be used to keep integration problems to a minimum
- Testing Plans – will be used to find, log, and fix all quality assurance issues
- Game Folder Hierarchy – shows layout of project and source folders

System Architecture



CGame

Description:

This module is responsible for initializing all the game settings, as well as initializing all global and system modules. When CGame is first initialized through WinMain, the game will be set to a 'running' state. The main menu state (CMainMenuState) will be initialized, making it the first state that the player enters. While CGame is running, the main game loop (in WinMain) will call the Input, Update, and Render methods. The main game loop will stop based on user's input in the game. Once out of the game loop, CGame will shutdown all the modules, which will deallocate any dynamic memory.

Dependencies

Access to the following:

- CObjectManager
- CStateManager
- CInputManager
- BatchRenderer
- CParticleManager
- CCameraManager
- CAnimationManager
- CMemoryManager
- CDebugger
- CTimer
- CSoundManager
- CEventsManager
- CHavokWrapper

Accessed by the following:

- WinMain

Members:

Type	Name	Description
Idirect3D9*	m_pDirect3D	Pointer to Direct3D object to initialize and access DirectX's functionality.
Idirect3Ddevice9*	m_pDirectDevice	Pointer to Direct Device to initialize and access DirectX's functionality.
HWND	m_hWnd	The handle to the window, passed from WinMain
HINSTANCE	m_hinstance	A handle passed from WinMain
CGame	m_cInstance	A static instance of CGame
CStateManager *	m_pStateManager	A pointer to the CStateManager instance, to access its functions and initialize the module.
float	m_fStartTime	The start time of the game.
float	m_fDeltaTime	The time that has elapsed from previous frame to current frame.
float	m_fPreviousTime	The time logged for the previous frame.

Bool	m_bRunning	A Boolean that is true if the game is still running, and false otherwise.

Methods:

Return	Name	Parameters	Description
Bool	IsGameRunning	void	Returns a Boolean that represents the running state of the game. If true is returned, the game is still running.
Float	GetDeltaTime	void	Returns the elapsed time from current frame to previous frame.
IDirect3D9*	GetDirect3D	void	Returns the pointer to the Direct3D object
IDirect3D-device9*	GetDirect3D-device9	void	Returns the pointer to the Direct3D device.
CGame*	GetInstance	void	A static method that returns the pointer of the instance to our class.
HINSTANCE	GetHInstance	void	Returns the HINSTANCE. (Mainly for the modules that needs access to this information during their initialization)
HWND	GetHwnd	void	Returns the HWND. (Mainly for the modules that needs access to this information during their initialization)
Void	Init	void	Initializes the settings of the game, such as window's size and the running state. Here, the game will also initialize all system modules, including the CStateManager, to enter the Main Menu state.
Void	Input	void	Determines whether to exit the game loop, based on the user input received from the game states.
Void	Update	void	Calculates the elapsed time and sends it down to the CStateManager to update all states of the game.
Void	Render	void	This method will clear the screen and render the game objects, by going through the render methods in the game states.
Void	Shutdown	void	Calls all major modules' Shutdown methods, which deallocates any dynamic memory.

Time to Complete Estimate

- CGame Total – 2 day
    - Implementation – 1 day
    - Testing – 1 day (A)
    - Integration – 1 day (A)

(A) Same day.

Module Author(s)

Hector Llanos, Elizabeth Kim