Alpha framework

Change Log:

24 – 11 – 2015

Legit pushing and popping of states are working perfectly (Princeton). Added pause state and pause scene to test gameStateManagerOverride functions. Functions are working.

21 – 11 – 2015

Removed 3 gameStates templates (menu, game, option). Implemented third person camera (WL). Edited gameStateManager to do legit pushing and popping of states.

19 – 11 - 2015

Removed cpps of the 3 inherited scene managers (selection, info, gameplay) and moved the codes to header. Child classes (individual scenes) should still inherit from them.

Previous gameStates (MenuState, PlayState) are now used as template for individual gameStates. Inherit directly from GameState, do not inherit from MenuState/PlayState.

Aim: To allow fast development of game prototypes or projects.

# Main Classes

## Application

A direct handle to create an opengl window and using GLFW/GLEW keyboard and mouse controls.

Consists of:

* 1 keyboard handler
* 1 mouse handler
* 1 GLFWwindow handler
* 1 Game State Manager

## Game States

Used as building blocks for a game/ game scene.

Functionality:

* Updating itself
* Pausing and resuming of itself when necessary
* Handle events
* Draw itself

# Handlers

## Input Handler

A handle to take in all the inputs from keyboard, mouse or/and controller. Input Manager for each game state will manage the inputs separately.

Examples:

* Keyboard
* Mouse
* Controller

## Resource Handler

A handle to the resource pool that consists of all the resources used in the game/scene.

Consists of:

* 1 Mesh Container
* 1 Texture Container
* 1 Shader Container
* 1 Sound Container
* 1 Game Objects (2D & 3D) container

# Managers

## Game State Manager

Manage flow of the game and the transitions between game states (if any).

Consists of:

* Vector of Game State
* 1 Resource Handler

## Scene Manager (Abstract)

Manage everything of a scene, from inputs to output.

Consists of:

* 1 Resource Manager
* 1 Input Manager
* Rendering functions
* Update functions

### Scene Manager Selection

Inherits from Scene Manager, specialize to take in input and choices from the user. Should only consists of buttons (2D/3D).

Specialize Functions(s):

* UpdateSelection(void) – Handle the selection made by the user

Classes that should derive from this class

* Menu Scene
* Level selection Scene
* Option Scene
* Pause Scene

### Scene Manager Info

Inherits from Scene Manager, specialize to display information and message to the user. Consist of a few buttons for the player to interact and changing of scene.

Specialize Function(s):

* RenderInfo(void) – Display info that are already predefined
* RenderInfo(string) – Display info outside of the class

Classes that should derive from this class

* Instruction Scene
* High Score Scene

### Scene Manager Gameplay

Inherits from Scene Manager, everything related to the game/a level of the game should be contain within this class. Consists of a wide variety of game objects that are required by the game/scene. Primary child class should be Game Scene.

## Input Manager

Manage the inputs that are defined for that particular scene.

## Resource Manager

Manage the resource required for that particular scene.

Loaders: Loading of OBJ, textures and shader into the program.

Consists of:

* LoadOBJ
* LoadTGA
* shader

# GameObjects

## Parent GameObject

### GameObject

### GameObject2D

### GameObject3D

### GameObjectSprite2D

### Mobile GameObject

### Static GameObject

## Child GameObject

### Button2D

### Button3D

Resource Handler

Collision Detection

Scene Graph

Scene

Display Handler / Renderer

Input Handler

Scene Manager

Scene2D Manager

Scene3D Manager

Resource Manager

Display Manager

Input Manager

Game States

GameScene Manager

Game State Manager

Application