Computer Science 164: Game Engines

Authors:
Leif Myer
Jonah Nobelza

Directory Navigation Instructions:

The 'Release' folder contains 'CS 164 Golf Game Engine.exe' that can be launched to play golf. The 'Debug' folder contains this as well. The 'Diagnostics' folder contains testing that is only visible in visual studio. This file contains performance statistics while the game is running. Inside the 'CS 164 Golf Game Engine' folder contains source code and dependencies. This directory will contain all game specific files and a README file that contains how to build the project and the change log. All include files and library files should be included in their respective folders. The 'engine' directory contains engine specific code.

Features:

Our engine was originally based off of the component model. Through out the course of the quarter it seemed to have developed into ta combination of the component model and the hierarchical model. Much like the unreal engine, there is a central actor object that all game objects extend from. This class has overridable functions like 'draw' and 'tick' that will be used by the separate systems like the Graphics system or the Ticker system respectively. The Actor3D class contains member variables that will be used by all game objects like position, scale and rotation. To create a game object, extend from the Actor3D class then add the address of that instance to the graphics, physics and ticker object.

Adding Levels:

To add additional levels, you must place a separate level file in the root directory. This directory currently contains 'hole.00.db, hole.02.db, hole.03.db'. Once the new files are placed, add them as command line arguments in visual studio. To do this, right click the project and go to properties. Under configuration properties, select debugging and you should see an option for command line arguments. There should already be 'hole.00.db, hole.02.db, hole.03.db' listed.

Additional Comments:

We must admit, we do not have a completed game. After finishing the rendering component of our engine, we started to fall behind. Admittedly, we should have tried to attend more sections, but schedules were tight, and both of us had commitments for other classes we needed to fulfill. Despite these obstacles, we still poured a lot of time into creating a game engine. We took pride in trying to solve issues ourselves, despite our minimal knowledge in C++. Although we did not accomplish constructing a complete game, the experience of trying to build a game engine from scratch provided a wealth of knowledge. Much of this experience involves teamwork, code structure and organization, design philosophy and implementation. After this class, we aim to construct a new engine utilizing the experiences has in this class.

Thank you for opening the door for us,

Leif Myer

Jonah Nobelza