CompSci 777 S2 C 2004

Assignment #3

Gaggle

Implementing a game in a multiplayer network gaming environment

Due 10:30pm Tuesday 28th of September 2004

Description

For this assignment you are asked to implement a game in a gaming environment called Gaggle. You are provided with code that implements the basic parts of the game environment, including networking, loading of textures and models, and the handling of network messages. What you are required to implement is the rendering, game management, and user interface.

Gaggle basics

Gaggle consists of Worlds and Players. A World is a game environment in which Players can play. What happens in a World is controlled by the World's network server program. A Player is controlled by a network client program. Worlds and Players communicate with each other through network messages.

A World consists of a static World Layout (used to define walls, floors etc), and dynamic geometric Objects. An Object consists of a Model (the geometric mesh) and a Texture (the texture applied to the mesh). Models and Textures may be shared by multiple Objects.

A Player is a type of Object. Like all objects, it has a Model and a Texture. Unlike other Objects, the Model and Texture for the Player are provided by the Player's client program. All other Objects in the World are provided by the World server. A Player can request what Objects are in the World, and any Models and Textures used in the World.

The World controls the creation, removal, and movement of all Objects in the World. Players may request to be moved, but the World decides how the Player is actually moved (if at all). Any changes in the World are communicated to the clients of the Players in the World.

A World may connect to other Worlds. How this connection is presented in the World's game environment is up to the World's server. A World can send a Player to a connected World. A Player can be in at most one World. Except for Players, no other Objects can be send to another World. When a Player joins a World, it is up to the World's server to decide where to place the Player.

The World Layout defines the basic environment of the game. It is specified as a regular 2D grid (although it does not need to be implemented in that way) of any size and scale. Walls, floor, and ceiling may be specified for each of the four sides of each grid square. Walls are single-sided and flat. The front side of a Wall faces inwards into the grid square. Each wall, floor, and ceiling has a Texture. Walls can be any height and are split into three sections (stacked vertically). Each section can be set to be open or closed, to form windows and doors for example. Players can ask a World about any rectangular region of the World Layout.

Assignment requirements

You need to implement and provide:

1. A Player client for a Player which can interact with any World
2. A World server which implements some game
3. A short report detailing your implementation

You may use existing libraries and game engines in implementing your solutions. When doing so, you must clearly indicate in your short report what code you wrote for this assignment, and what code was provided by other people's libraries and engines. Do remember that any libraries you use must be available freely to others for marking (so if you use a commercial engine, you must pay for us to get a full license :-)).

You may also modify any files provided for this assignment, with the proviso that the changes do not break compatibility. Player clients written by other people need to be able to play in your World, and your Player client needs to be able to play in other people's Worlds. Furthermore, if you modify the files, you will be on your own applying any bug fixes.

Player client (40% of assignment)

The Player client implements a 3D renderer to show the World the Player is currently in, typically from a first person point of view. Information about the World needs to be kept track of as it is received from the World server. Models, Textures, and Objects need to be stored in data structures efficiently for fast rendering.

The Player client also needs to get input from the user so that the Player can be controlled through, for example, the keyboard or mouse.

Marking: marking will be based on the correctness of rendering the World Layout and Objects; the efficiency of rendering; the quality of the output; the support of Gaggle features; the functionality of the user interaction; correctness of managing the Player based on the user input and World messages; and general "Wow!" factor.

World server (40% of assignment)

The World server should implement some game for Players to play. You need to manage Models, Textures, Objects, and Players in the World. You also need to create a World Layout.

The World server needs to control the movement of any Objects it creates in the World, as well as the movement of Players based on the requests sent by the Player clients. Collision detection must be implemented by the server to stop Players from going through walls and Objects.

Marking: marking will be based on the efficient management of Models, Textures, Objects, and Players; the correctness of Object and Player movement; the correctness of interaction with Player clients and connected World servers; the support for, and imaginative use of, Gaggle features; the innovation, complexity, and completeness of the game; and general "Wow!" factor. Bonus marks will be awarded for reusing the Player client renderer through good design, to allow viewing on the server of the happenings in the World without needing a Player client.

Short report (20% of assignment)

Provide a short report describing what data structures and algorithms you used in order to achieve fast rendering and efficient data management. Give the reasons behind your choices.

Also include instructions for the game implemented by your World server, and how to use your Player client.

If you used any libraries or game engines, indicate what code you wrote for this assignment, and what code was provided by other people's libraries and engines. Give the reasons for choosing those particular libraries or game engines.

Assignment submission and due date

The due date for the assignment is 10:30pm Tuesday 28th of September 2004

Submit all your files needed to compile and run your implementation through the Assignment Drop Box. Please include executables in case there is difficulty compiling your code. You do not need to submit any files provided for this assignment if they are unmodified. You also do not need to submit any libraries you used that can be obtained over the web for free (provide the URLs where the needed libraries can be downloaded from in your report). If your programs require anything special to be done in order to compile or run, write it in the report. You may also include VC++ solution and project files. The report should be submitted in some commonly readable format, such as RTF, PDF, DOC, or ASCII text.