INDIVIDUAL DESIGN DOCUMENT

Harsh Gupta
2009CS10191

September 19, 2010

Abstract

This document contains all the high level implementation details of implementation of the multiplayer ping pong game

1 High Level Design

1.1 Various interacting objects

The three high level entities (described below shall run as separate threads)

1. A thread shall handle all the display activities for a server or client alike.This function shall be common to both the client and the server.

2. One thread shall take in user input from the player/players playing on the concerned computer. This thread would then pack and send the information to the server also.

3. One thread shall recieve data from the server and update the data as received from the server.The effectiveness of this thread shall contribute largely to the responsiveness of the system to the input received from the user. [1]

1.2 Communication with the "world"

1.2.1 Receiving data from the rest of the "world"

1. Client computer:The thread that receives from the server shall receive all the data via a single socket.The data shall be identified with markers as to which players it relates to. As described in the common design document only a diff of all the data shall be received by the application.

2. Server computer:Say there are N client computers interacting with the server at any time.Then the server shall receive data from N sockets ,one per client computer. [2]

1.2.2 Sending data to the rest of the "world"

1. Client computer:Any user input,ie any change in the position of the centre of the paddle of the player will be sent to the server via the socket reserved for sending information to the server initially. [3]

2. Server computer:The server shall send data to each client via its respective socket.

[1]The reason for making a separate thread for receiving data from the server is that we do not want too much data to be held up in the buffer.If there was a large amount of data stored in the buffer,then we would be displaying outdated values to the user,thus reducing the response of the game.

[2]Reasons for choosing N sockets over select() system call are that this would reduce the overhead of slecting one thread to a process then another.The tradeoff is between overhead of multiple sockets and better responsiveness of the game.

[3]Note: For a client setup, i shall have one socket handling both the sending and receiving of the data from the server.

2 Display

2.1 Basic Algorithm

A thread shall be managing the user output. This thread shall read the variable values and output to the screen in an appropriate manner.

2.2 User input

The user can give input via both the keyboard and the mouse.The Paddle can be controlled with the help of the arrow keys and the camera by the mouse.The glutKeyboardFunc will be used to handle user input.

2.3 Display

1. 3d graphics shall be used for display. The player shall have the option of viewing a bird eye view ie 2d view of the field or use the mouse to set the location and direction of the camera .The gluLookat() function shall be used for this purpose.

2. The player shall have an option whether to play in a full screen mode or in a window mode.This shall be implemented with use of GlutEnterGameMode and GlutLeaveGameMode functions.

3 Object oriented features

3.1 Reusing code

The code will be highly reusable with the functions being overloaded as required in the case of a client/server. [4]

3.2 Classes used

1. Player:This will have the functions required for a player, both computer and manual. [5]
2. Ball: This stores all the data which must be stored for a ball(radius,present coordinates,velocity).
3. Board: This stores the information which will be used for the board specifications (the dimensions).

3.3 Exceptions and Exception Handling

1. When A Host Connection is suddenly lost a custom exception , HostConnectionLost thrown and appropriately handled by catch block.
2. When host goes down then a custom exception ,HostWentDown shall be thrown and handled as mentioned above. [6]

4 Optimisation

4.1 Profiling code

Gprof(the gnu profiler) will be used to profile the code,search for bottleneck functions which are making the code run slower and then optimising them.

1. Detect memory leaks with the help of mtrace() call
2. Check for memory corruption using electric fence [7] [8]

[4]For example A send request shall be perceived as a broadcast to all clients in the case of a server and a simple send to a single server in the case of a client.This function could be overloaded.Other similar opportunities for overloading shall be exploited.

[5]A separate class for Computer and Manual Player using inheritance is not considered as it might be necessary to switch from a manual to a computerised player in the middle of the gameplay

[6]Custom expressions will be defined by extending the class for myexception

[7]In a multithreaded environment if any of the threads gets a segmentation fault, then the whole process comes down. Such detection of memory leaks will detect possible crashes of the game.

[8]Electric fence overwrites the default malloc,new and free to check for freeing memory twice,using a null pointer,corrupt memory etc conditions