Hunter Gorczycki
hunterg@escalitygames.com

130 Unity Portion Simple Documentation

Unity Side:

There are two main scripts that are being used for this part of the project. Other scripts were used for the demo to show off the mapping function and gps data, but are not included in the git repository.

Socket.cs

- Functions:
- Awake()
    - Sets up the TCP Connection variable.

- Update()
    - Every frame, we call the functions here. We will be calling SocketResponse() to listen for any messages that have come through the socket.

- OnGUI()
    - A custom function to create a button in the top left of the screen that will serve as the input and interaction of our testing.
    - Creates “Connect” button that sets up the socket in the TCP class.
    - Once the socket is ready and connected, the connect button is changed to a “write to server” field and button. This lets you write whatever you want to the server (string).

- SocketResponse()
    - Checks the socket to see if any new message is available.
    - If a message is available, it sets a Unity text variable to the message - this shows up on the mian UI of the Unity program. If UI is not set up, it will output to the debug console in Unity.

- SendToServer()
    - Called when you click “Write to Server” on the Unity UI. Usually uses whatever message is written in the field above the button. However, for the demo we populated this with your current GPS coordinates.

TCPConnection.cs

- Variables:
    - conName and conHost: set a name and an IP address for where you wish to connect to.
    - conPort: what port number does your server have open to listen for TCP connections.

- Functions:

- SetupSocket()
    - Creates the socket using stream writers and readers
    - Essentially the initializer function.

- ReadSocket()
    - Called on every frame to see if new data has been sent through the socket.
    - If so, convert the data from a byte array stream into somethiung more readable in a buffer and return it.

- CloseSocket()
    - Called when the connection is terminated on either end. We want to close the socket and stop listening.
    - This is called when TCP can confirm there is no listener at the other end.
    - Closes the reader, writer, and socket.

- MaintainConnection()
    - Keep trying to connect if you are not connected.

Server Side:

This is the C++ project located in ServerC. This is a collection of cpp and .h files that represent the game server. This can be ran on a command line or executable prompt. For our purposes, we ran it inside Visual Studio 2015.

- GameServer.cpp
    - GameServer()
        - Initialize the server
        - Client_id is set to 0 as there are no clients yet. 0 will be the host.

    - Update()
        - Called every frame.
        - Check if there are any new clients attempting to connect and connect them if they are.
        - Increase client count
        - Call ReceiveFromClients to check if there is any message waiting in our buffer from a client to be processed.

    - ReceivedFromClients()
        - Go through each of the threads (clients) in your server and check if they have sent data to you\

    - SocketConnections.cpp
        - Two wrapper functions for Winsock receiving and sending data

- Server.cpp
    - Server()
        - Sets up winsock conditions and variables for opening sockets
        - Sets up the sockets to listen to incoming requests

    - acceptNewClient()
        - Checks if any TCP connection request has been made
        - If so, assigns it a session/thread and adds the client to the server

    - sendMessageToEachClient()
        - Used to send a string back to every client connected to the server
        - In the future, will probably want to make a function that sends only to the client in question. But for now, it iterates through all sessions and sends a string through the socket.

    - ReceiveData()
        - Check each session to go through every client
        - Check if any one of them has a message they are trying to send to you
        - If there is a new message, received it and process it
        - THIS IS WHERE THE C++ QUERY API WILL BE CALLED
        - Resets the buffer so that it can receive new messages

- Main.cpp
    - Runs two infinite while loops to keep the server on and updated at all times.