Tutorial Project (Understanding the Basic Game Engine, Server, Network Protocol and beyond)

Client Part Due 2/18 (Sunday) 11:55 PM (7 points)
Server Part Due 3/4 (Sunday) 11:55 PM (10 points)
Documentation (2 points for Client and 2 points for Server)
Total - 21 points (& the part I is 3 points)

Overview

The importance of this project is to introduce you to developing for both the client and server by going through a series of basic to slightly more advanced programming assignments.

Every assignment found in this project will be discussed in lecture, but it is recommended for you to approach these assignments yourself ahead of time, so you can ask questions.

This is an individual project. You are encouraged to discuss between your fellow classmates, but you are expected to do all the programming yourself. You will be required to submit both your code and a write-up of your results in the end.

A. Client Development

Creating a Scene

1. Load the provided client codes (WoB_Client).
2. Create a new scene named "TestScene" and place it under the Scenes directory.
3. Using the editor, insert a terrain prefab named "MaasaiMara", found under the Prefabs directory, into the scene.
4. Attach a new C# script named "TestSceneScript" into the scene and place it under the Scripts directory.
5. Upon execution, the script needs to perform the following:
    a. Store the existing terrain position, then remove the terrain from the scene.
    b. Instantiate a new, but same terrain at the stored position.

Populating Scene with Species

1. Attach a new C# script named "TestSceneScript2" into the same scene.
2. Upon execution, the script needs to perform the following:
    a. Immediately instantiate 5 different species of your choosing at random positions on the terrain.
    b. Instantiate species using keyboard inputs while the game is running. Create 3 different species using 3 different keys.

Creating a Simple GUI Element

1. Modify the C# script named "Login" to create a new button at the center bottom of the screen. This script currently belongs to the Login scene.
2. Program this button to switch from the current scene to your "TestScene".
3. Create a similar button in your "TestScene" to switch back to the Login scene. You should implement this in "TestSceneScript".

Using Collision Rays to activate mouse click on the scene

Using collision rays, create a script to allow users to spawn a species directly on the terrain by using a single mouse click. The species should appear exactly on the location it was clicked.

1. Create a new C# script and attach it to “TestScene”.
2. Script should respond to mouse inputs.
3. Using collision rays, perform collision detection with terrain whenever mouse is clicked.
4. Instantiate a species at that location.

Port the client code to mobile platform (Optional)
Port the client code to mobile platform. This semester, we will be porting the game to Mobile phone. As a starting point, you need to port the project 1 client part to your mobile phone. You need to read the instruction from Unity to support mobile platform. First you will have to install Android Studio (if you want to port to Android) and link SDK to Unity. You also need to download Unity 5 that contains the support for Android. Also you need to test with Unity Remote for interactive development.

B. Server (& Client) Development

Server Login Verification

The client already has the capabilities to send login information as well as receiving verification status from the server, but the server portion is incomplete.

Data being passed around should be hardcoded.

1. Modify the existing login protocol (RequestLogin and ResponseLogin) on server-side to verify a user. Implementation should be compatible with the existing code on client-side.
2. Extend ResponseLogin on both server-side and client-side by send and received the following data:
    a. User ID (String Data Type)
    b. Player Level (Short Data Type)
    c. Money (Integer Data Type)
3. Output the information in Unity3D’s console once the client receives it.

Retrieve Number of Connected Players

Tip: Connected players are stored in a container in the GameServer instance.

1. Create a new protocol to retrieve the total number of players connected. Label each as follows: “RequestPlayers” and “ResponsePlayers”.
2. Trigger the request once login is verified.
3. Output the information in Unity3D’s console once the client receives it.

Create a new protocol

Create a new protocol to manage and manipulate persistent data stored in the server. To test this, you will need to send a request from the client to the server to trigger these actions.

1. Start off by creating a request named “RequestTest” and a response named “ResponseTest”. Both should follow the same structure as other protocols.
2. Create a new integer variable within “GameClient” called “newTestVar” with an initial value of 1000.
3. The client should be able to communicate with the server to trigger the following operations that modifies “newTestVar”:
    a. Addition
    b. Multiplication
4. Output the results to the server-side console.

Database (Optional)

Using the login protocol you’ve modified previously, remove the hardcoded values and make use of a database instead. This involves creating a single table to store all the values and reading from it.

1. Create a table in the database called “users”.
2. Create fields that store all the necessary user information needed for the login process. This includes the additional information you had to add.
3. Modify the login protocol by making calls to the database to retrieve this information.