CSE 190 Discussion 7

Final Project: Dual User VR

Agenda

- Final Project: Dual User VR
- Dual User Application - Networking
- Technical Requirements
- VR Experience
- Extra Credit

Final Project: Dual User VR

- The Final Project is up on the webpage
  - Due Tuesday of Finals week - June 11th at 3:00pm
- You will create a two user VR application for two Oculus Rifts with controllers running on two separate computers
- Some features you need to implement:
  - Users need to work together on something
  - Need to use collision detection, touch controllers
  - Incorporate audio
  - Create at least one 3D object on your own

Final Project: Dual User VR

- In addition to the application also need to create a website/blog with at least two posts and video
- Report/Blog Posts Due:
  - #1 - Monday of Week 10 (June 3rd at 11:59pm)
  - #2 - Monday of Finals Week (June 10th at 11:59pm)
- Video Due:
  - Tuesday of Finals Week (June 11th at 3:00pm)

Report #1

- The first post needs to contain (at a minimum)
  - The name of your project (You need to come up with one)
  - The names of your team members
  - A short description of the project
  - One or more screenshots of your application in its current state

Report #2

- The second post needs to contain (at a minimum)
  - Progress you made
  - Updates on any changes made to the team or team name
  - Post another screenshot
- Can use whatever site you wish
  - Just make it the same for both posts
- Feel free to add more entries beyond the required two

Video

- Each team needs to record a video (2-3 min long)
- This will be shown during the first hour grading during the final
- Need to use youtube
  - We will be creating a playlist for you to add your videos to
- Videos are due by June 11th at 3:00pm

Dual User Application - Networking

Networking

- Recommended networking library: RPC
  - Used to call functions and send information over the network between your two controllers
  - Documentation here
- You can use other, more extensive libraries for networking online, but it will be up to you to figure out how they work.

Sockets Sample

- All the code for the loop above is within the example listed previously
- How the sample works:
  - Sender:
    - Serialize data into one large array
    - Send it over the network
  - Receiver:
    - Parses the information based off the packet headers you create

Sockets Sample

- Quick Networking Crash Course:
  - 127.0.0.1 is the localhost, use it to test on the same computer
  - Make sure each user uses different ports,
    - If you don’t the second person who tries to use it will get an error
  - Ports numbers < 1024 are reserved

Sockets Sample

- To find your local IP:
  - open up command prompt
  - run ipconfig
  - look for the interface that looks correct
    - Looks correct if...
  - Then look for your IPv4 address

Network Architecture

- Important things to keep in mind:
  - The clients should run at 90 fps, but the server can run at 30 ticks per second
    - Keep your camera calculations on the client, only send over the scene graph
  - Having the server run at a constant rate might make some calculations easier

Network Architecture

- Important things to keep in mind:
  - Don’t try to do everything at once.
  - First make sure you can get communication over the network
  - Then integrate it with your code
- Server - Client Architecture:
  - Server = simulation of the virtual world
  - Client = window into the virtual world

Server-Client Architecture

- The server is a simulation of the virtual world
- The client is a window into the virtual world

Server

- Have the server run at a constant rate to simplify things
  - 30 ticks would be a good starting point
- The server will maintain data structures for all the objects in the simulation
  - A scene graph might be good here

Server

- The server will run a loop consisting of something similar to the following
  - Receive input from clients
  - Update the state of the simulation (collisions, physics, etc)
  - Send state to clients
  - Wait until next tick
- Having a constant tick rate simplifies how the server works and limits the work the server puts on the client

Client

- Client maintains local copy of the state
- Client loop runs as fast as possible:
  - Receive messages from server
  - Renders the virtual world
  - Collect input events to send to the server
- The client does not act on input directly
  - The server will do the update, and send the world state back to the client
- Having an authoritative server helps avoid conflicts

Events

- The server needs to send object data to the clients
- Send events between the Server and Client
  - Sending “fire” events from client to server
  - Sending “hit” events from server to client
- Adds in indirection between button presses and actions

Architecture Adjustments

- Since our case is very simple, with only two users, you can simplify the architecture a lot.
- You can have one of the computers run a combined server/client, and
  - Run the server on another thread OR
  - Only do the server update every so often
- If your game is simple enough, you could run the server along with the client in the same program
- In the end, do whatever makes sense to you and your game

Technical Requirements

3D Model Customization

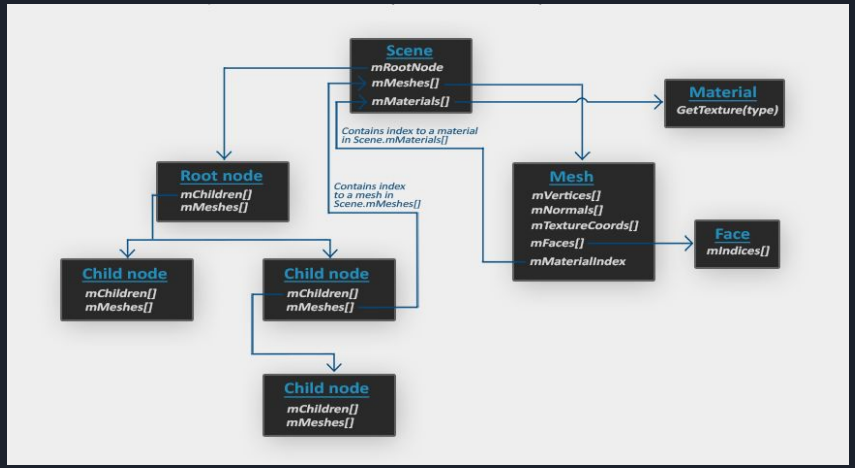
- You need to create at least one 3D object
- Scan model from pictures:
  - Can use tools like Agisoft Photoscan
  - You can checkout the workflow of Agisoft Photoscan, here is a tutorial for it
  - There is a 3D scanner in the VR lab that you can try out

3D Model Customization

- You can create or pre-process your 3D meshes using tools like Blender or MeshLab
- Make sure your model is optimized enough
  - Since we are using network communication, models that are too big will cause your framerate to drop
  - You can use MeshLab to reduce the polygon count of your model

Model Loading

- Make sure you export your model is exported to Wavefront .obj file or fbx file or files that you are familiar with
- If you are using Assimp, check out its supported file formats here
  - Assimp supports fbx and obj file formats
  - Regardless of the file format, the data structure of Assimp stays the same.



MTL files

- Creating models sometimes generates additional files to store color, material, and texture information
- One of these is the Material Library file (.mtl)
  - Contains material definitions
- To access loaded materials with Assimp:
  - `aiMaterial* material = scene->mMaterials[mesh->mMaterialIndex];`

MTL files

- To check if your obj file contains a mtl file definition:
  - Open the obj file as text file
  - Check the headers to find the name of the mtl file
    - It often has the same name as the obj file
  - You need to place the .mtl file in the same directory as the .obj to load it

```
1  # Blender v2.78 (sub 0) OBJ File:
   'Handgun_Game_Blender Gamer Engine.blend'
2  # www.blender.org
3  mtllib Handgun_obj.mtl
4  o bullet_Cube.005
5  v -1.692615 -0.021714 -1.219301
6  v 7.334266 -0.021714 -1.219302
```

Texture files

- Some materials might specify texture files that needs to be used for the material
- In obj/mtl files, a material can specify paths to image files
- Again you need to make sure you place those image files where the .mtl file needs them

```
2  # Material Count: 5
3
4  newmtl Fire.001
5  Ns 96.078431
6  Ka 1.000000 1.000000 1.000000
7  Kd 0.640000 0.640000 0.640000
8  Ks 0.500000 0.500000 0.500000
9  Ke 0.000000 0.000000 0.000000
10 Ni 1.000000
11 d 0.000000
12 illum 2
13 map_Kd textures\handgun_Fire.png
14 map_Ke textures\\handgun_Fire.png
15 map_d textures\\handgun_Fire.png
16
```

Texture files

- To load these texture images:
  - You can refer to the Model.cpp example in LearnOpenGL
- `vector<Texture> loadMaterialTextures()`
- `unsigned int TextureFromFile()`
- These two functions show how to load the texture images.
- To find texture coordinates of the mesh:
  - `mesh->mTextureCoords[0]`

OpenAL: Open Audio Library

- You need to incorporate audio into your application
  - Can be simply background music or triggered by an event
- We recommend using OpenAL
  - Link to OpenAL slides

VR Experience

VR Experience - Usability

- Usability determines a big part of the experience
- You shouldn’t need to spend a long time explaining your controls
  - Make controls simple
  - You don’t have to work on UI too much, just don’t make them overly complicated
  - Anyone with decent amount of VR experience should be able to easily figure it out

VR Experience - Usability

- Some question to ask to yourself:
  - How intuitive is my application to use?
  - Will my friends or other fellow classmate be able to figure it out?
  - Does the user have any guide throughout the experience?
  - How do my controls compare to real world actions?
  - Am I showing too much information at same time?
- Demoing the app to the grader in a right way also matters

VR Experience - Creativity, Aesthetics ...

- How original/unique is your application?
  - Try googling your ideas/searching the Oculus Store/Steam
  - See how other people implement ideas, and see if you can do something different?
  - Take advantages of the techniques used in past assignments
- Some aspects that you can innovate:
  - Topics and application styles
  - Interaction techniques
  - Audio, etc.

VR Experience - Creativity, Aesthetics ...

- Think about:
  - Color schemes
  - Models
  - Textures
  - Materials
  - etc.
- Resources of guidelines toward designing a good VR experience

Extra Credit

Extra Credit

- We created a list of hackathon-style awards for teams with outstanding apps
- They also serve as a good guideline for the features we encourage everyone to take care of in your design

References & Links

- Client-Server network using C++ and Winsock
- MTL file format explained
- Agisoft Photoscan tutorial
- Agisoft Photoscan Review article
- Assimp:
  - Library
  - LearnOpenGL Tutorial and sample codes

QUESTIONS?