CSE 190 Discussion 4

HW2: Level of Immersion

Agenda

- Homework 2 Minor Updated
- Render Panorama Image
- 3D Stereo, Mono, Single Eye Only, Inverted
- Head Tracking Filter
- Interocular Distance Adjustment
- Tips for the Extra Credit

Homework 2 Recap

- Assignment is updated with rubric details:
  - http://ivl.calit2.net/wiki/index.php/Project2S18
- Due Date: May 4th 2pm (This Friday!)
  - If you have scheduling conflicts, let us know
- Some minor changes:
  - IOD range now should be from -0.1m to +0.3m
  - Rendering mode cycle should start with 3D stereo, and then mono, left, right.
  - You should varying the cube size while maintaining it to be a cube (not just width)
  - Refactored extra credit options

Note: no teammates for this homework.

Render Panorama Images

Render Panorama Images

- Rendering the skybox should be a review from CSE 167
- You will need to differentiate which eye you are rendering to.
  - Check the eye type, it it is ovrEye_left, then render the left skybox,
  - Render right skybox for ovrEye_right
  - Make sure the texture coordinate is correct
- If you render it as a giant Cube surrounding user, the physical placement needs to have a horizontal offset.

3D Stereo and Mono

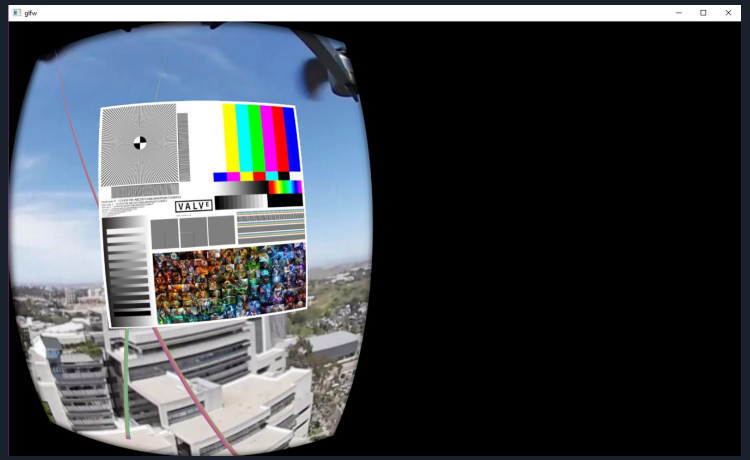
- If you look at the source code posted on this assignment page, you will find comments on areas of code which will give you hints on how to modify the code to change the rendering settings

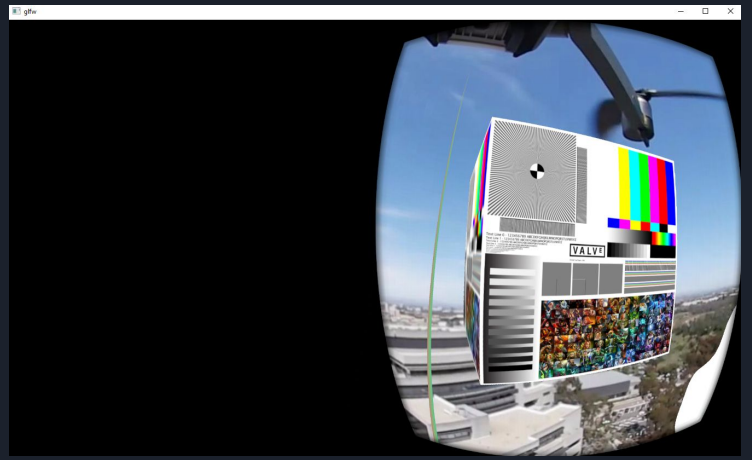
- These comments are only HINTS and it is up to you to figure out what to do with the information.

- To render mono, you can just render both eyes from the left side.

Left eye Only and Right eye Only

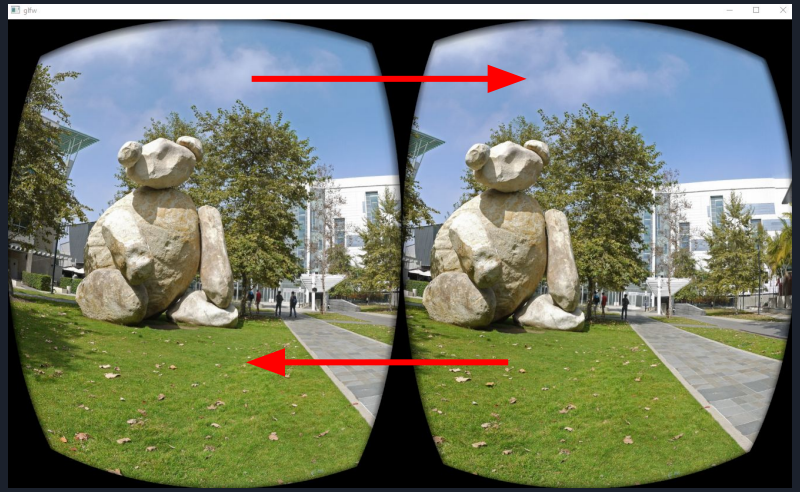
- Only render if it is the correct eye, otherwise just skip the rendering step





Inverted Stereo

- Simply swap which eye pose each eye uses to render
  - Still check ovrEye_left and ovrEye_right



Filtering Head Tracking Data

- When you change tracking modes, you will need to save the previous eye poses. Then when you render, you will reuse either the rotation or position part of the data

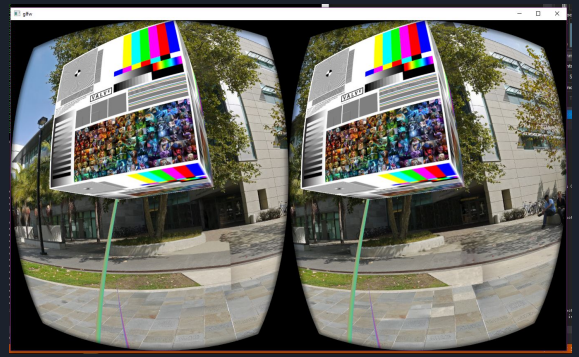
- Remember that the rotational data is stored in the top left 3x3 part of the matrix and the positional data is stored in the rightmost column

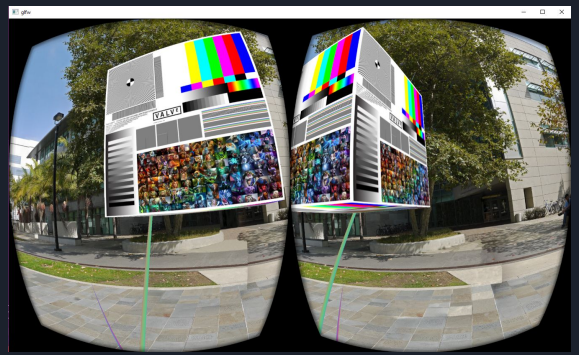
Interocular Distance Adjustment

- You can see an example of setting the HmdToEyePose (or HmdToEyeOffset depending on the SDK version) in the RiftApp constructor.

```
//Setup initial IOD
IOD = std::abs(_viewScaleDesc.HmdToEyeOffset[0].x - _viewScaleDesc.HmdToEyeOffset[1].x);
```

- You will need to modify it to change the IOD
  - When assigning back, you might need to divide it by 2





Some Technical Tips

- ovrPosef has a Position(Vector3) and Rotation(Quaternion) you might want to look into

- There are various toGlm function to help translate the various ovr data types to glm data types

Testing Tips

- When testing locking the position/rotation, try either only rotating or moving your head to see if it works.

- To test stereo/mono/inverted, try looking directly at the edge of a cube

Extra Credit
- Stereo Image Viewer

- It should just be two “regular, non-panoramic” pictures
  - Set camera FOV to as close to 90 degrees as possible
  - It’s like using the Oculus HMD as the ViewMaster Image viewer on it.
- Similar to the skybox code, you will need to render two different textures to your cube to create the stereoscopic effect

Extra Credit
- Custom Sky Box

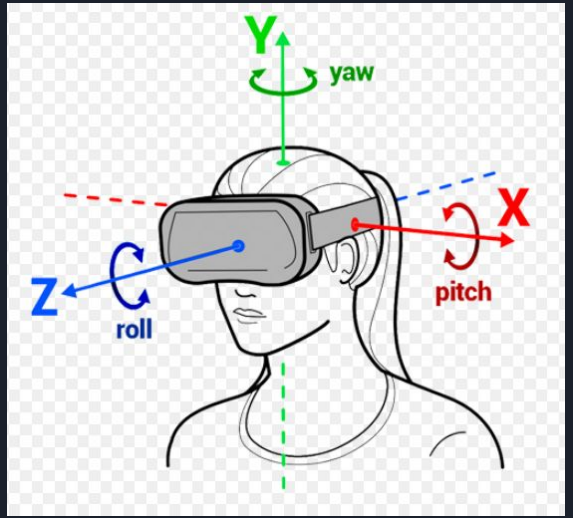
- Create your own monoscopic skybox.
- Convert your 360 degree panorama picture into a cubemap:
  - https://jaxry.github.io/panorama-to-cubemap/

- Make it an alternative option when “X” button is pressed
  - It will just be a skybox in mono
  - The device can be borrowed at the Media Lab
  - Can also use the Google StreetView App

Extra Credit
- Super Rotation

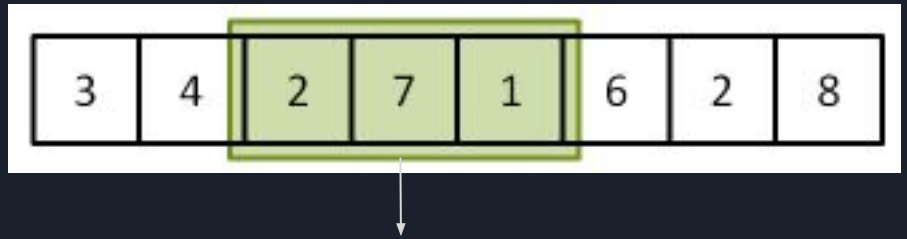
- Doubles the movement of your yaw rotation (left and right)
  - If you rotate your head to the right 90 degrees, you are looking to the behind

- No magnification on roll/pitch
  - When looking up, you are still looking to the above.
  - If the head is only tilting, it should be no effect of super rotation as well.



Extra Credit
- Smooth Controller Tracking

- Averaging the tracking data and beforing rendering the controller
  - Moving Average Explained
- Example:
  - Choose size as 3, average the value (position only!)
  - Then move the window to right by 1 cell, average again
  - Use the averaged data as the actual controller position



Ideally, the larger the window size, the smoother the controller movement should be, and more lag will happen when moving the controllers.

QUESTIONS?