Durham University

Assignment
Virtual & Augmented Reality

Important, please read first!

- The assignment should be submitted via DUO; for deadlines, please consult DUO/level handbook.
- The Unity project you submit will be tested on Unity Version **2020.2.2** - make sure to use this version. If you have more than one Unity version installed, use the Unity hub to manage them.
- The methods asked in problems 1, 2, 3 should be implemented by you, as evidenced in the source code that you will submit, not via existing functionality available in Unity - no marks will be awarded if you use Unity's built in distortion code.
- Submit a compressed archive (.zip) with (a) your Unity project containing **only** the requested scene, meshes, scripts & shaders, and (b) a .pdf report no longer than 6 pages including images (max ∼2000 words).
- The marks available for correct implementations/answers to each question are indicated. Partial credit will be given for good attempts.
- The Virtual & Augmented Reality module is only assessed by this coursework.
- A FAQ section in DUO will be updated with questions as they arise.

You have recently been hired by a VR headset manufacturer. You are part of the team that is developing the headset's lens undistortion/pre-correction subsystem. Your team's mission is to provide an efficient implementation of the undistortion mechanism to minimise the computational penalty on the rest of the rendering pipeline. You have been tasked to implement a few variations of the pre-correction system, comment on their performance & accuracy and discover remaining limitations. If you succeed in this, your implementation will be embedded in the VR headset device driver and you will get a promotion!

As discussed during class, almost every VR headset released to date has had some level of visual corrections implemented in software to correct for artifacts caused by some aspect of the hardware. Correcting for lens distortions is a great example of this. Chromatic aberration and pincushion distortion are common for an optical system with a large field of view. When looking through the lenses of an uncorrected VR headset, pincushion distortion and color fringing are always very apparent. When images are drawn on the screen without any correction, then the virtual world appears to incorrectly warp as the head moves: if the users yaw their head back and forth, then fixed lines in the world, such as walls, appear to dynamically change curvature because the distortion in the periphery is much stronger than in the center. If uncorrected, then the perception of stationarity diminishes because static objects appear to be warping dynamically. Furthermore, VR sickness usually worsens because incorrect accelerations are being perceived (visually) near the periphery.

Performing efficient lens corrections is a vital part of the VR pipeline. By using the right techniques we are able to reduce the GPU requirements for this specific part of the pipeline while keeping the image quality high. This enables the application to create richer content or the device to save power by shortening the time the GPU has to be awake.

Before proceeding, please read LaValle's chapters 4 & 7.3 (free book available in DUO) and attend relevant lectures. Helpful but not necessary reading: [1, 2] (also provided in DUO). The distortion correction formulas that you will use for all problems are based on a heavily simplified version of Brown's model, that only uses two coefficients, $c1$ and $c2$ for radial distortion, omitting the p-coefficients required for tangential distortion correction. In the formulas below, world points that are inside the canonical viewing frustum have x and y coordinates ranging from -1 to 1. We will refer to these points using polar coordinates (for notation please consult LaValle's chapter 7.3) :

$$r = \sqrt{x^2 + y^2}$$
$$\theta = \text{atan2}(y, x)$$

Brown's simplified Forward radial transform:

$$r_d = f(r_u) = r_u + c_1 r_u^3 + c_2 r_u^5$$

Brown's simplified Inverse radial transform:

$$f^{-1}(r_d) \approx \frac{c_1 r_d^2 + c_2 r_d^4 + c_1^2 r_d^4 + c_2^2 r_d^8 + 2 c_1 c_2 r_d^6}{1 + 4 c_1 r_d^2 + 6 c_2 r_d^4}$$

PROBLEM 0: SETTING UP THE SCENE - 5 MARKS:

For all problems, you will use the same simple scene. The scene should contain 9 white rotating cubes, arranged on a 3x3 grid on the default Unity skybox, covering the entire field-of-view. The frustum and view-port should be **square**. The cubes should be rotating, completing a revolution every 2 seconds. For marking, I should be able to test each effect by changing parameters in an *Inspector* object to select a different problem/parameters.
Important note: The output for the next 3 problems will be simulating a pre-corrected image that would supposedly be sent to a headset. No headset is required to complete the coursework. Sending the images you generate to an actual headset will not look right (actually will look identical as on Unity's output!), as the headset drivers are already implementing the correct pre-distortions for that particular headset model.

PROBLEM 1: PIXEL-BASED PRE-DISTORTION IN THE FRAGMENT SHADER - 25 MARKS:

For this problem you will implement pincushion distortion pre-correction in the fragment shader by warping the frame buffer appropriately. Assuming that the lens is radially symmetric, the pincushion distortion can be described as a stretching of the image that becomes increasingly severe away from the optical axis.

1. To correct the pincushion distortion effect, you will need to apply an appropriate transformation so that the light emitted by the screen reaches the eye undistorted. Use the simplified radial distortion model mentioned above. The actual parameters you use do not matter but the inverse distortion generated should be clearly visible (not subpixel...). [10 marks]
2. You will perform (as in most VR software) this reverse distortion by rendering the scene camera into a separate render target (render texture) and then projecting that texture on a full-screen quad while warping the texture in the fragment shader. You will then display the full-screen quad on the output. [15 marks]

PROBLEM 2: CORRECTING LATERAL CHROMATIC ABERRATION - 10 MARKS:

For this problem you will implement lateral chromatic aberration (LCA) pre-correction. LCA is an artifact inherent to VR lenses. LCA is caused by red, green, and blue light refracting through lenses differently (wavelength-dependence). A white pixel on the panels will refract through the lenses and separate into red, green, and blue pixels visible to the viewer.

LCA correction aims to adjust for this by pre-distorting the rendered image in the opposite way so that the image viewed by the user after lens refraction appears as a single white pixel as intended. LCA artifacts usually look like opposing red and blue color fringes emanating from the center of the optics.

To correct for LCA you will implement the inverse of the expected colour-dependent magnification of the image in a shader and show the result on screen. Pick parameters that make the inverse colour fringing visible (not subpixel...). Actual values do not matter as this is for demonstration purposes only.

PROBLEM 3: MESH-BASED PRE-DISTORTION IN THE VERTEX SHADER - 30 MARKS:

For this problem you will implement pre-distortion in the vertex shader to take advantage of optimisations in the rendering pipeline potentially leading to high performance gains.

1. You will use a square mesh (plane made of triangles) which you will pre-distort in the vertex shader (displace its vertices in world space) and then the scene camera's output will be projected onto the warped mesh (applied as a texture) and the mesh rendered to the screen using another camera. [15 marks]
2. Use Blender to create three different meshes of different geometric complexity that will be used for full screen rendering. Comment on the effect the different mesh geometric complexities have on pre-distortion quality. Use the Unity Profiler to actually measure performance impact and describe what you found in the report. [10 marks]
3. Finally invert the output (unwarp the pre-distorted image as if it had gone through the lens), again using the mesh-based method, and compare it to the original image. What do you see? Answer the relevant questions asked in problem 4. [5 marks]

PROBLEM 4: RESEARCH REPORT - 30 MARKS:

The main purposes of the research report is to include evidence of testing, provide images/answers to the questions below, and comment on anything else that you consider important.

1. Produce static renders for the report, where all 9 cubes are rotated 45 degrees around the y-axis, showing the inverse chromatic aberration pre-correction, the barrel distortion in the fragment shader and the barrel distortion using the pre-calculated mesh. [6 marks]
2. What are the advantages and disadvantages of pixel-based and mesh-based undistortion methods? Comment on the accuracy of pixel-based vs mesh-based approaches and their effect on visual quality. Comment on the performance of both methods, i.e., comment on the expected number of calculations and texture lookups for each method based on the frame buffer resolution that you have used. Comment on the effect that the number of vertices in the mesh has on rendering quality. [8 marks]
3. Would eye tracking help for any of these distortions? If so, how? If not, why not? [6 marks]

4. When performing barrel distortion, some parts of the image are compressed in a smaller real estate in the image, other parts are expanded. What would this cause? Suggest potential fixes. For a given display panel resolution, would you suggest the rendering resolution to be equal to the hardware resolution, less or more and why? Further comment on apparent resolution, how detailed a scene 'appears' to be or is expected to be following all these transformations. Discuss the ramifications that this could have in modern headsets. [10 marks]

References

[1] BROWN, D. C. Decentering distortion of lenses. *Photogrammetric Engineering and Remote Sensing* (1966).

[2] DE VILLIERS, J. P., LEUSCHNER, F. W., AND GELDENHUYS, R. Centi-pixel accurate real-time inverse distortion correction. In *Optomechatronic Technologies 2008* (2008), vol. 7266, International Society for Optics and Photonics, p. 726611.