Xuan (James) Zhai

CS 5382/7382 Final Exam Fall 2021

This is a take home, open-book, closed-friend exam.

You may use any informational resources available to you including the textbook, research articles, online tutorials. You **may NOT use other students** in the class or post questions about these questions anywhere. If you wish clarity on questions, or directions, you may ONLY ask the instructor.

The goal of the written and oral exam is for you to demonstrate you own understanding of the principles, which you may convey using words, graphics, examples and an ability to discuss and apply them.

It is worth 70% of your final exam grade. The rest of your final exam grade is comprised of oral discussion related to these in addition to demo your submitted programs.

Answers must be submitted to Canvas by midnight Thursday, May 5, in advance of your oral exam appointment.

The last part of this exam is related to an article you are asked to read and respond to. 5xxx students choose ONE of the papers. 7xxx students must do both.

I. Lighting

1 a) Please write the equation for the Phong illumination model here. It is the basis for the rest of this question

$$I = \underbrace{k_d I_d \, l \cdot n}_{\text{Diffuse}} + \underbrace{k_s I_s (v \cdot r)^{\alpha}}_{\text{Specular}} + \underbrace{k_a I_a}_{\text{Ambient}}$$

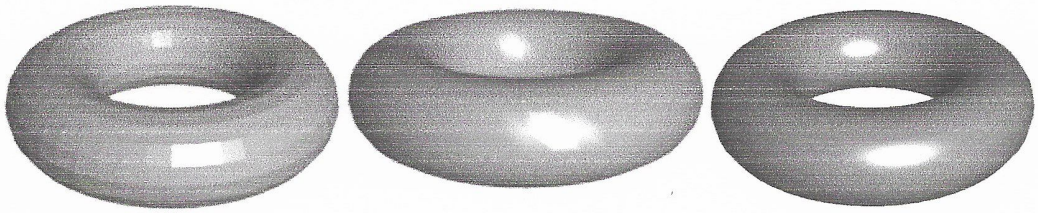
b) Identify the three components of this model and specify the dependencies of each component with respect to viewer position, light position or direction, surface normal.

The three components of this model is Diffuse light, Specular light, and Ambient light. The diffuse light relates to the diffuse property of light and material, light position and normals. Specula component has the relation between the viewer's position and the reflection. Ambient component is about the property of light and material.

c) Please identify the component related to specular highlights and describe the role of each component, describing, in particular what corresponds to "shininess."

The component relates to specular highlights is $k_s I_s (v \cdot r)^{\alpha}$ in the equation. In the model, the ambient light component is similar to a default light to the environment. It comes from all direction and lights up even shadowed portion. The diffuse light component relates to the light source. The lighting is brighter toward the light source. That's why in $k_d I_d l \cdot n$ we need light position and the normal relates to the light position. Specular light components depend on how the light reflects off the object, or the highlights on the objects. It needs the angle of the viewer and the reflection of the light to form the shiness $(v \cdot r)$. α is the shiness coefficient which sharpen the drop off between 0 and 1.

2) Consider the following images, created with the **same lighting model** applied to the SAME set of vertices specifying the geometry and topology of object. The normals, however, differ.



Render 1 Render 2 Render 3

Please discuss how each of these renders may have been created to achieve the different results. Include a sketch of normal. For each, what is the role of the vertex shader, fragment shader, application.

Render 1: This is Flat shading. For each polygon on the object, the color applied on each vertex is the same. That's because their normals are the same in one polygon. The normal we use is the one that normals to the polygon. Therefore, Flat shading computes the color of each polygon, and the color can be determined in the application

Gouraud shading

Render 2: For each vertex, it has its own color/normal attribute. This normal is computed by taking the average of normals of faces it's adjacent to. Thus, in one polygon, different vertices may have different normals/colors. The color is determined in vertex shader, and the rasterizer will figure out the colors of fragments between two vertices. (Interpolation for interior pixels)

Render 3:

In this Phong Shading, the interpolation is not with colors but with normals. Each pixel will have its own normal attribute, and then use that normal to calculate the color of each pixel. Therefore, in this shading method it will calculate the color in the fragment shader.

3) What is the different between bump mapping and reflection mapping?

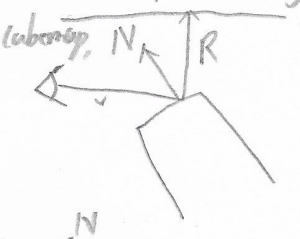
- Describe specifically in terms of techniques, type of texture maps, normals, visual effect.
- Specify WHERE (e.g. vertex or fragment shader or application) different techniques are applied.
- Include a sketch to demonstrate the use of normal in creating the effect.

1) Bump mapping can apply fake bumps to a smooth surface, which is just like cheating. When you want to make a surface rough. You can create a lot of vertices. Or, you can pertub the normal vectors without having to make any change to the shape. The graphs below can give you an idea about how we can fa

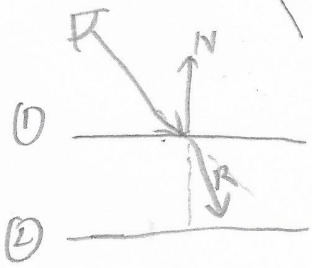
a rough surface on a flat surface. In this mapping technique, the normal of each pixel is not come from interpolation but from a normal map. In a word, normal are stored in the bumpmap texture. In the implementation, you need to have a norm map in the application to get the normal for each pixel. Later, you will create the texture based on those normals.

2) reflection mapping (environment mapping) is another technique which will simulate the reflection of environment on a shiny object. The environment is generated wit a map; normally sphere map or cube map (sky box). In the environment, you can create the object can give it reflection and refraction effect. For the reflection,

For example with cubemap,



You will need to get the reflection vector R based on the view position and the normal. In the fragment shade we then just use R to sample from cubemap. About the refraction, it's also in the fragment shade, but a bit different in getting the R. The R is calculated with a refractive index between two materials.



4) Consider the following two fragment shader, used to perform image processing computations in which an image is loaded into a texture. Briefly describe the visual effect of each

a)

```
uniform sampler2D uImgUnit;
in vec2 vST;
out vec4 fFragColor
void
main( )
{
     vec2 st = vST;
     vec3 rgb = vec3(1.0, 1.0, 1.0)- texture2D( uImgUnit, st ).rgb;
     fFragColor= vec4( rgb, 1. );
}
```

vec3 rgb = vec3(1.0, 1.0, 1.0) - texture2D (uImgUnit, st).rgb;

This line of code will reverse the rgb value of the texture

Reverse (Invert) the color means product the opposite color in the spectrum

b)

```
uniform sampler2D uImageUnit;
in vec2 vST
out vec4 fFragColor;
void main( )
{
  ivec2 ires = textureSize( uImageUnit, 0 );
  float ResS = float( ires.s );
  float ResT = float( ires.t );
  vec3 irgb = texture( uImageUnit, vST ).rgb;
  vec2 stp0 = vec2(1./ResS, 0. ); // texel offsets
  vec2 st0p = vec2(0. , 1./ResT);
  vec2 stpp = vec2(1./ResS, 1./ResT);
  vec2 stpm = vec2(1./ResS, -1./ResT);
  vec3 i00 = texture( uImageUnit, vST ).rgb;
  vec3 im1m1 = texture( uImageUnit, vST-stpp ).rgb;
  vec3 ip1p1 = texture( uImageUnit, vST+stpp ).rgb;
  vec3 im1p1 = texture( uImageUnit, vST-stpm ).rgb;
  vec3 ip1m1 = texture( uImageUnit, vST+stpm ).rgb;
  vec3 im10 = texture( uImageUnit, vST-stp0 ).rgb;
  vec3 ip10 = texture( uImageUnit, vST+stp0 ).rgb;
  vec3 i0m1 = texture( uImageUnit, vST-st0p ).rgb;
  vec3 i0p1 = texture( uImageUnit, vST+st0p ).rgb;
  vec3 target = vec3(0.,0.,0.);
  target += 1.*(im1m1+ip1m1+ip1p1+im1p1);
  target += 2.*(im10+ip10+i0m1+i0p1);
  target += 4.*(i00);
  target /= 16.;
  fFragColor = vec4( target, 1. );
}
```

This is the sharpening effect.

This nine rgb values are the values of current pixel and the eight pixels it's adjacent to.

Here you add some weights to those rgb values and normalizes them

$$B \leftarrow \frac{1.0}{16.0}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

You sample the texture value on the current pixels and the eight pixels, change the weight and do some average, and apply the result to the current pixel so that the edge can be sharpened.

PAPER REVIEW PORTION

If you are in 5xxxx choose **one** of the papers included online.
If you are in 7xxxx section, you must read both papers. Read, at a minimum, the sections noted. Then answer the questions on the next several pages.

PAPER A: Pixel-Art Magnification

a) What is the **title** of the paper?

MMPX Style-Preserving Pixel-Art Magnification.

b) **When** was the paper published? **Where** was the paper published?

Journal of Computer Graphics Technique in 2021.

c) **Who** are the authors?

Morgan McGuire and Mara Gagiu.

d) Summarize the paper, in your own works, in 3-4 sentences.

MMPX is a new filter for magnify pixel art.
It can reconstruct lines and curves while preserving the palette, transparency, and some features in arts.
Compared to other filters like nearest-neighbor filter, this filter can preserve the art style and feature.
This filter is compatible with many platform/application, and the processing time is fast.

Read Section 1.1, 1.2, 1.3 and comment on these things:

h)What problem does this paper address? (Say, in comparison to existing methods that can be improved?)

Many filtely algorithms like point sampling, their results do not preserve the original art style, but exhibit exaggerated sharpness and jagged edges on curves and lines. How to achieve "style-preserving" while doing magnification becomes the problem.

i)Why do the authors claim this an important problem to work on?

The authors of this paper are developers of quadplay fantasy console. They, and other console / game developers believe that a better pixel art magnification technique can help modern arts be more compatible on consoles.

j) What are the two differences between the Load-Time filter and the Run-Time filter approaches?

Load-Time 1) preprocess to generate additional image (size & orientation).
2) Target creation of new games with pixel-art style.

Run-Time 1) process individual images in each frame.
2) target modern or historical content that did not anticipate access to a high-resolution display

Section 2 is a lengthy discussion of existing and related work. Note Figure 5 and the portion of the text that relates to it.

k) How does this relate to what we have learned about texture parameters?

In class, we have seen the difference between GL_Nearest and GL_Linear in GL_TEXTURE_MAG_FILTER. One is slightly faster and the other has relatively smooth transition. Here, the Unbiased bilinear magnification filter is the same as the GL_LINEAR we have seen in class.

Sections 3 and 4 include a great deal of detail about the technique and evaluation of results. Look at Figures 15-19 showing the visual comparison of for techniques.

l) Which technique do you think is better and why?

I think MMPX is the best overall because it did well in most scenario. XBR is also pretty good, but it cannot handle 1:1 edge, or edge between 30°-60° really well. EPX and Nearest are bad in 1:1 edge, 2:1 edge, and rounds.

Look at Table 1, showing performance data.

m) Which language of implementation performs best?

GLSL

n) Which technique is most efficient?

Nearest in processing time

MMPX in processing time + result

o) Is the improvement of the technique you chose above (l) a good choice overall?

Yes, it's fast enough to create a result which is even better than some implementations already in production use.

works from

Read the Conclusion

p) (7xxx students only) What has the work achieved?

PAPER B: Art, Perception and Image Formation

a) What is the **title** of the paper?

b) **Who** are the authors?

Read Section 1: Introduction

c) Summarize what his paper is about and the primary goal of this paper?

Read Sections 2-4

d) What big ideas (2-3) did you find most interesting?

Read Sections 5.1 – 5.4. Although this discusses creation of images by artists, there are some parallels to image formation in graphics.

e) Explain at least 4 big ideas and choices that artists have that correspond to the concepts we have discussed in class. Clearly describe how these are implemented in a graphics system.

f) Identify elements of artistic style that can not be captured by the techniques we have discussed so far.