Part I

Question One

0.1 Question

We want to combine three transformations into one by multiplying them. The transformation S should be applied second; the transformation T should be applied first; the transformation R should be applied third. Write the product of the three transformations showing the order in which they should be multiplied.

0.2 Answer

1. Translation
2. Scaling
3. Rotation

$$\begin{bmatrix} x^1 \\ y^1 \\ z^1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{Translation} \underbrace{\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{Scaling} \underbrace{\begin{bmatrix} \cos\theta & -a\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{Rotation} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a & 0 & 0 & a \\ 0 & b & 0 & b \\ 0 & 0 & c & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -a\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} \cos\theta & -a\sin\theta & 0 & a \\ \sin\theta & \cos\theta & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{ResultingMatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

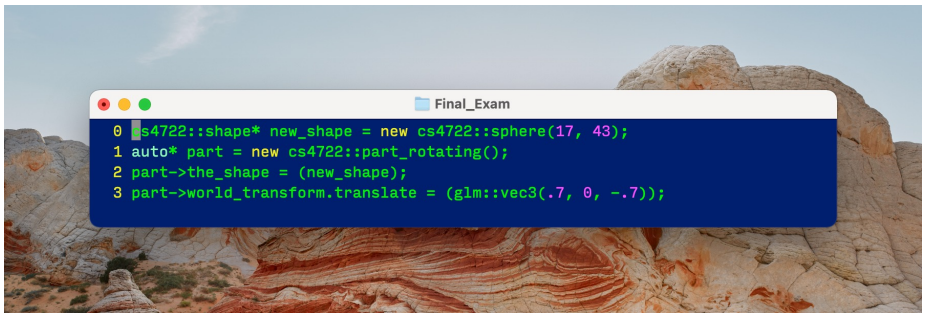
Part II

Question Two

0.3 Question

Write the code that carries out the following steps. You may assume that the appropriate imports have been provided.

0.4 Answer



Part III

Question Three: Data Memory

0.5 Question

We want to render a geometric figure which is modeled using 38 triangles. As with almost all of our examples, he figure will be drawn using gl triangles mode. This question is about measuring the memory used to store the vertex data.

0.5.1 How many vertices will be used to represent each triangle?

0.5.2 Answer

3 Vertices will be used to represent each of the triangles using the gl triangles mode.

0.5.3 Using the standard way we have represented vertex positions in our code examples, how many float values will be used to represent each vertex position?

0.5.4 Answer

There will be four float values used with each of the vertex positions.

0.5.5 How many float values in all will it take to represent all the vertex positions for the figure?

0.5.6 Answer

It will take 456 float values to represent the 38 triangles.

0.5.7 How many bytes will that take to store all this position data?

0.5.8 Answer

It will take 1,824 Bytes to store all the position data in this scene.

0.5.9 In the examples where colors were assigned to vertices, each color was represented by a cs4722::color object. How many bytes does it take to store the data (r, g, b, and a components) of a cs4722::color object?

0.5.10 Answer

There is one byte for each color coordinate, Given the R, G, B, and Components that will amount to four bytes.

0.5.11 How many bytes would it take to store all of the color data assigned to the vertices of this figure?

0.5.12 Answer

Given that each color component is four bytes it will take 456 bytes of data for color storage in this scene.

Part IV

Question Four: Pipeline

0.6 Question

Each part below describes a stage or a process in the standard OpenGL rendering pipeline. Your answer is to name the pipeline stage or process.

0.6.1 For a single geometric primitive, determines the pixels that are covered by the primitive.

0.6.2 Answer

Primitive Setup

0.6.3 Subdivides shapes more complex than primitives into standard primitive shapes

0.6.4 Answer

Culling and clipping

0.6.5 For a single vertex, does some processing and outputs a vertex to the later stages

0.6.6 Answer

Vertex Shader

0.6.7 Takes one or more vertices and creates a geometric primitive from them

0.6.8 Answer

Geometry Shader

0.6.9 For a single pixel, does some processing and outputs a color for that pixel

0.6.10 Answer

Fragment Processing

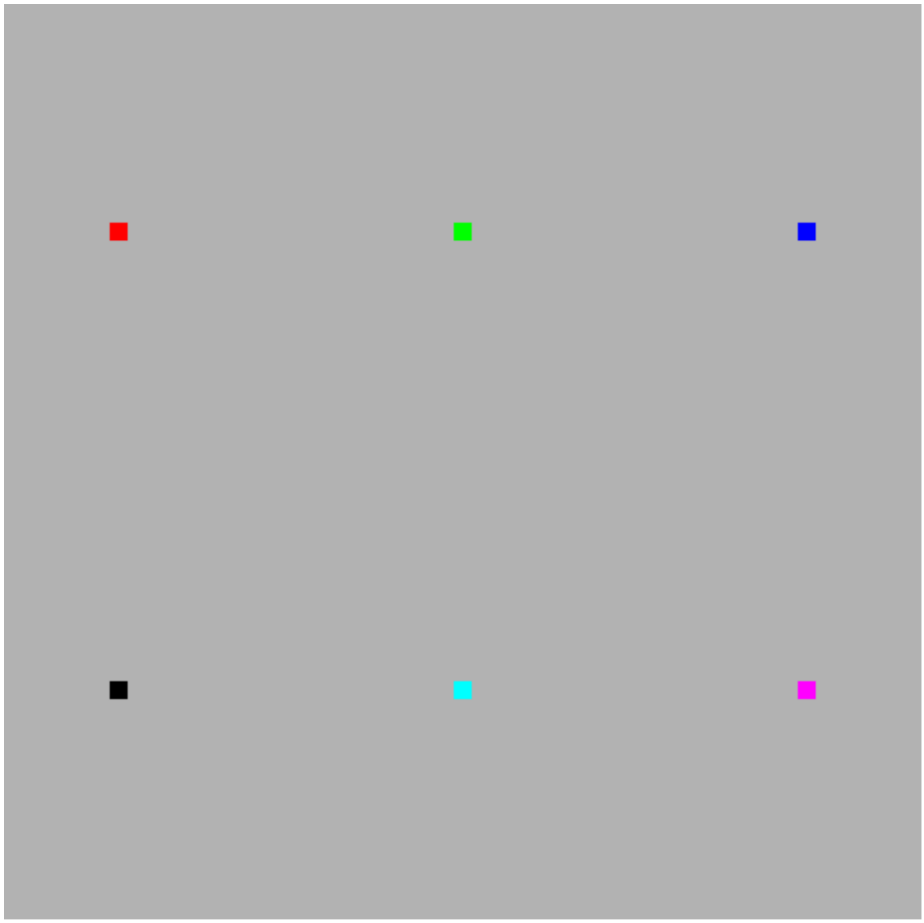
Part V

Question Five: Drawing Modes

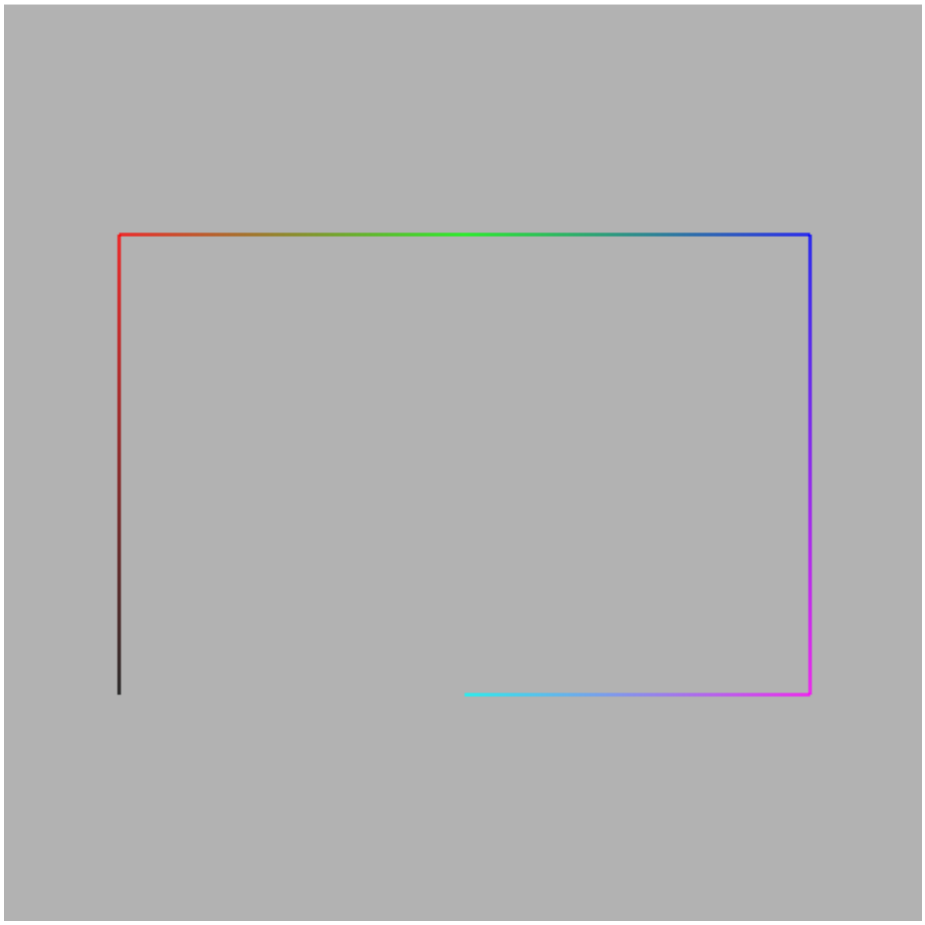
0.7 Question

The image to the below shows six points stored in the GPU buffer. The list of points starts with the black point, lower left, and continue around clockwise: black, red, green, blue, magenta, cyan. These points can be used to draw triangles or lines or points' in various ways, depending on the drawing mode. TRIANGLES or LINES for example.- -¿ For each picture below, tell which drawing mode was used to produce that picture. Each of the seven drawing modes available is used in one of the parts.

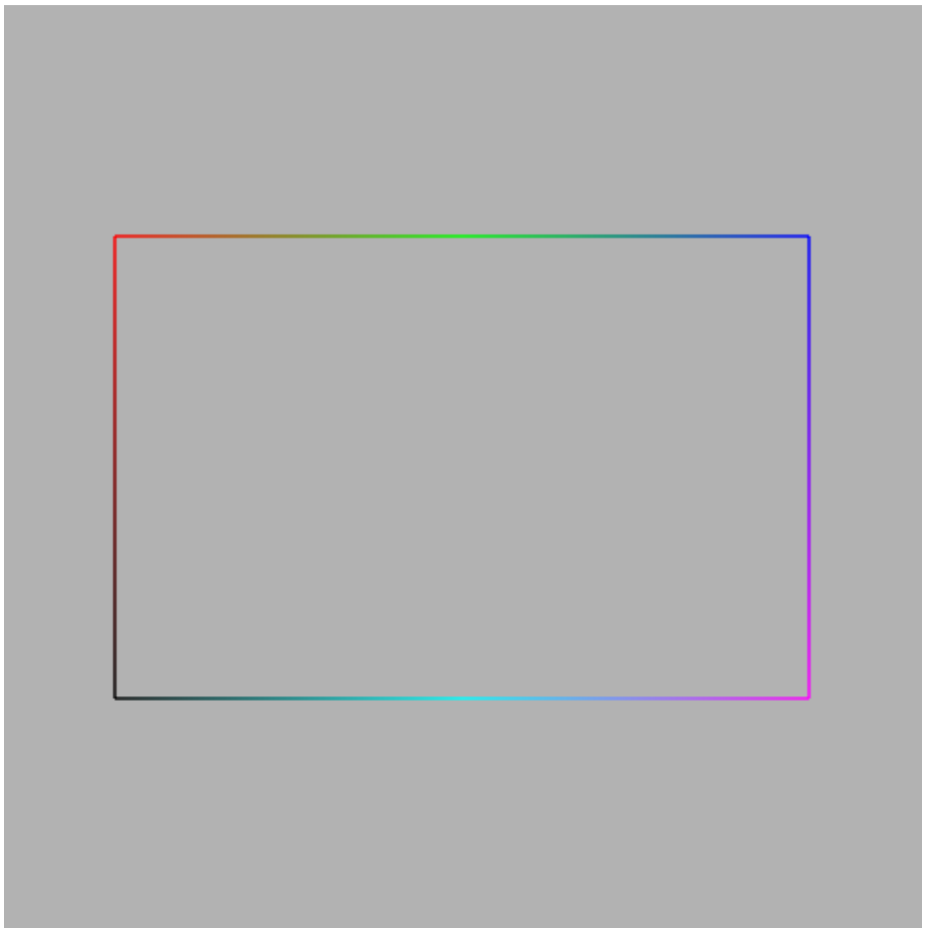
0.7.1 POINTS



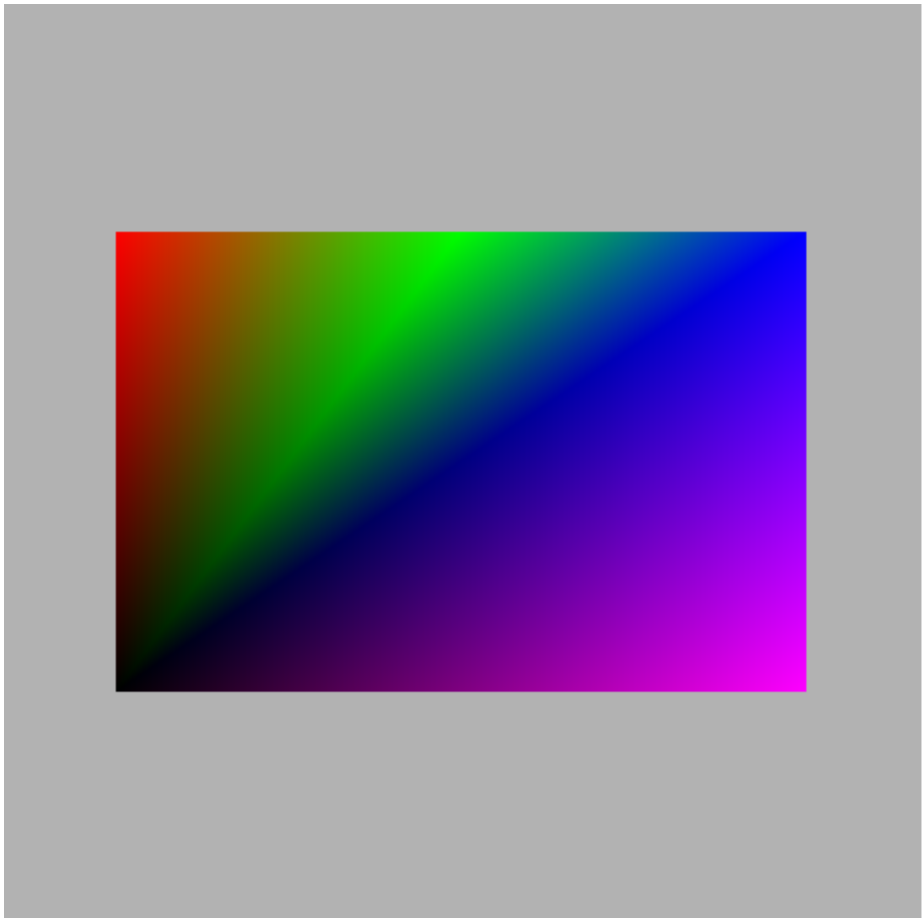
0.7.2 LINE STRIP



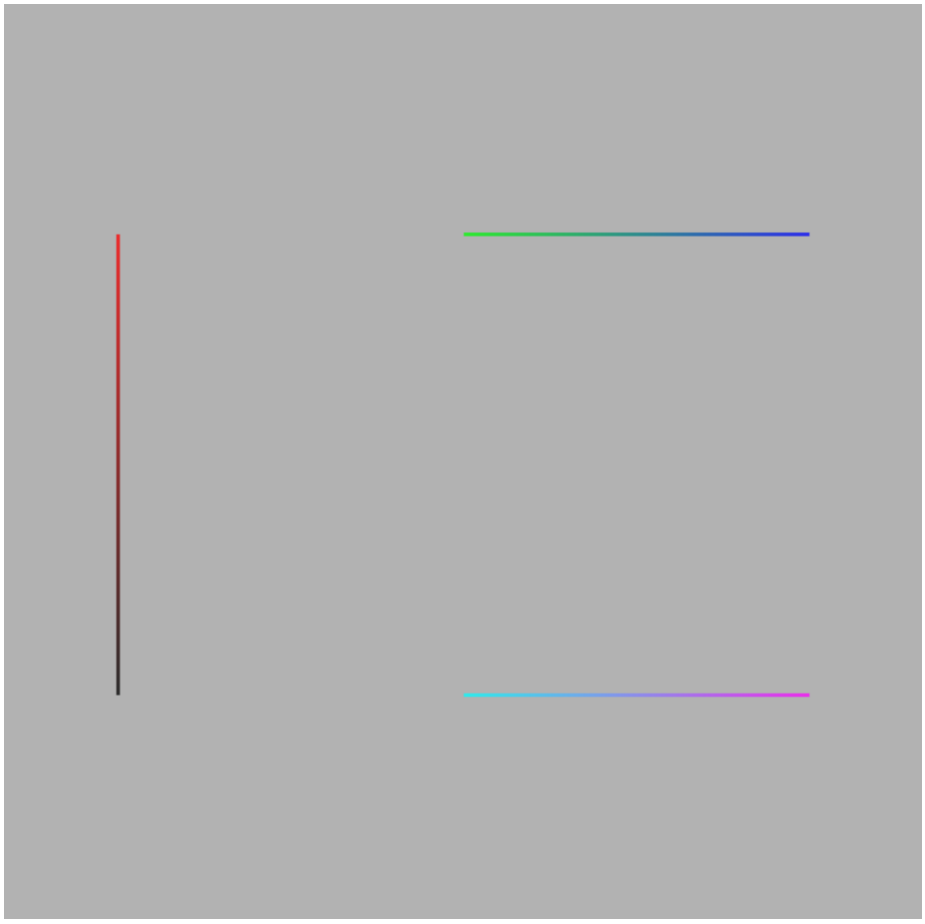
0.7.3 LINE LOOP



0.7.4 TRIANGLE STRIP

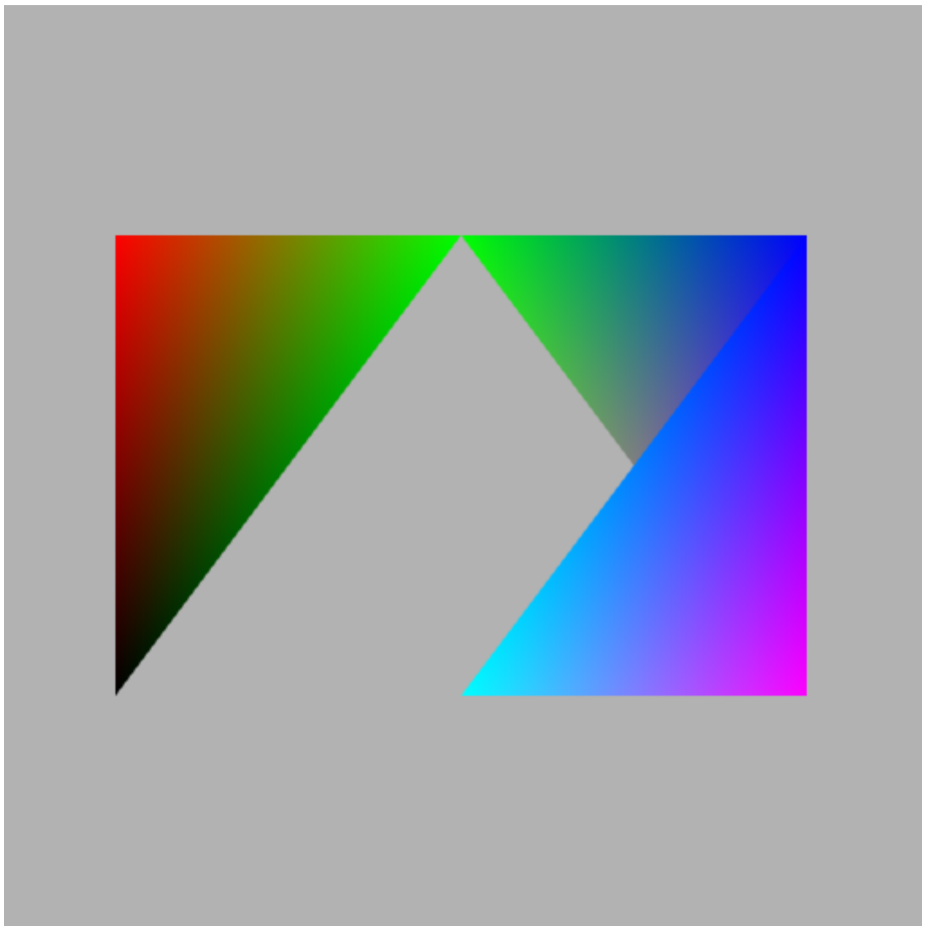


0.7.5 LINES



0.7.6 TRIANGLES

0.7.7 TRIANGLE FAN



Part VI

Question Six: Shading Language

0.8 Question

In each case, name the built-in variable in the shading language.

0.8.1 A vertex must be assigned to this variable

0.8.2 Answer

gl Position

0.8.3 A value can optionally be assigned to this variable to set the size of graphic that is used to display a point.

gl PointSize

0.9 Answer

Part VII

Question Seven: Projection Transformations

0.10 Question

What we call a projection transformation and have used in transforming vertices in our scenes is not actually a projection: it does not flatten the scene into two dimensions. Why is the scene left in three dimensions to be passed on to the fragment shader?

0.11 Answer

Because objects are not visible if they are farther away from the viewer then objects which are behind, or Depth Checking.

Part VIII

Question Eight: Lighting Geometry

0.12 Question

This question is about the geometry used to help compute the three kinds of shading: ambient, diffuse, and specular. When we computed shading, there were three main geometric components we used: vector from a point on a surface to the camera; vector from a point on a surface to the light source; vector normal to the surface. However, some of the shading types might use only some of these geometric components. Some might use none. This question asks you to tell for each geometric component, which kind of shading uses that component when being computed.

0.12.1 Notes

This part confuses me because there are only three shading methods to choose from I would want to just put all three for each one but I don't think I will.

0.12.2 Which of the three kinds of shading depend on the direction to the camera?

0.12.3 Answer

Specular depends on the direction of the camera.

0.12.4 Which of the three kinds of shading depend on the direction to the light source?

0.12.5 Answer

Diffuse depends on the direction of the light source.

0.12.6 Which of the three kinds of shading depend on the normal to the surface?

0.12.7 Answer

This will be the Ambient Lighting

Part IX

Question Nine: Texture Sampling Modes

There are various ways that texture samples are actually computed. This question asks about two different sampling modes.

0.12.8 The constant GL NEAREST specifies a certain way of determining a sample value. Explain how a sample would be determined using GL NEAREST mode.

0.13 Answer

A Sample will be determined by choosing the value that is nearest distance to the center of the pixel that is being textured.

0.13.1 The constant GL LINEAR specifies a certain way of determining a sample value. Explain how a sample would be determined using GL LINEAR mode.

0.14 Answer

The sample will be determined by getting the weighted average of the four texture elements that are closest to the center of the pixel being textured.

0.14.1 Compare the GL NEAREST sampling mode with the GL LINEAR in two ways: which gives better visual results; which takes more time to compute.

0.15 Answer

GL LINEAR is the slowest of the two but gives the better visual performance depending on the texture being sampled.

Part X

Question Ten

0.16 Question

What feature when using images for textures in OpenGL makes use of reduced size images in areas where the texture image is larger than the area being covered?

0.17 Answer

You can use the mipmap feature to use a reduced size image texture.