

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

$$\begin{aligned}
 \begin{bmatrix} x^1 \\ y^1 \\ z^1 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -a \sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\
 &\quad \text{Translation} \quad \quad \quad \text{Scaling} \quad \quad \quad \text{Rotation} \\
 &\begin{bmatrix} a & 0 & 0 & a \\ 0 & b & 0 & b \\ 0 & 0 & c & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -a \sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\
 &\begin{bmatrix} \cos \theta & -a \sin \theta & 0 & a \\ \sin \theta & \cos \theta & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\
 &\quad \text{Resulting Matrix}
 \end{aligned}$$

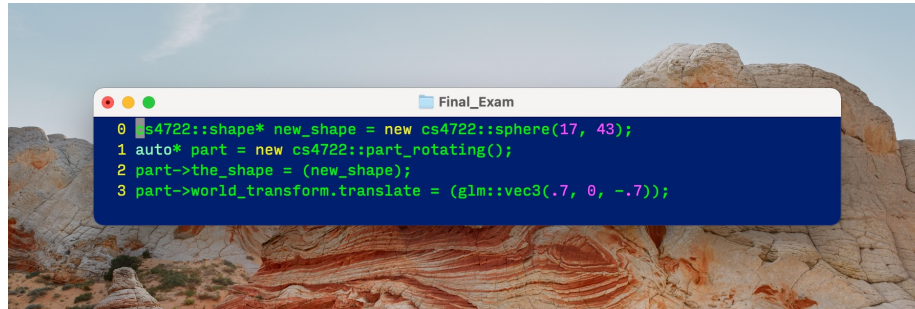
Part II

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, the 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

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

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

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

0.5.8 Answer

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

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

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

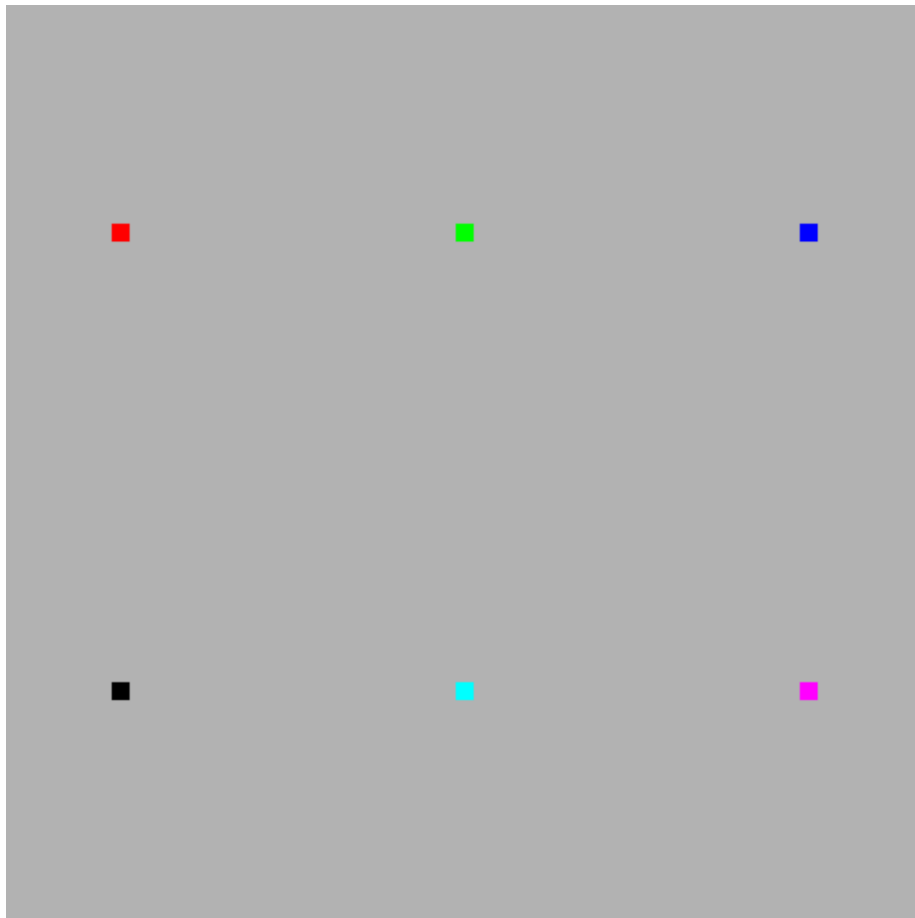
Part V

Question Five: Drawing Modes

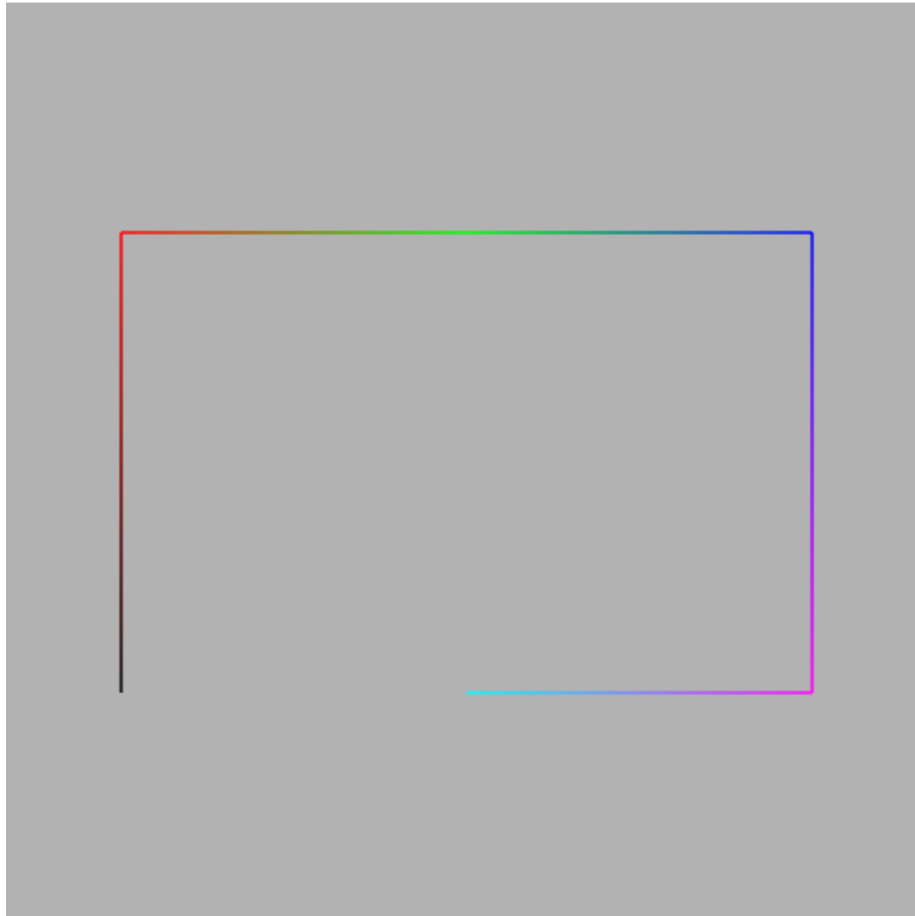
0.7 Question

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.

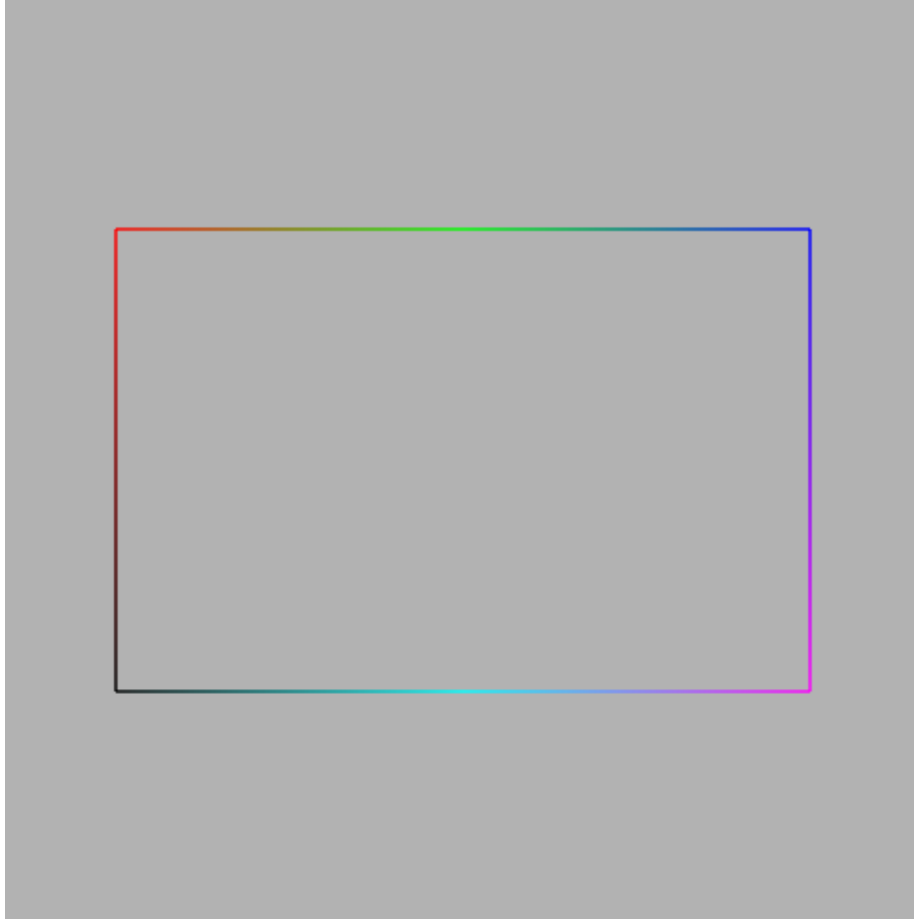
0.7.1 POINTS



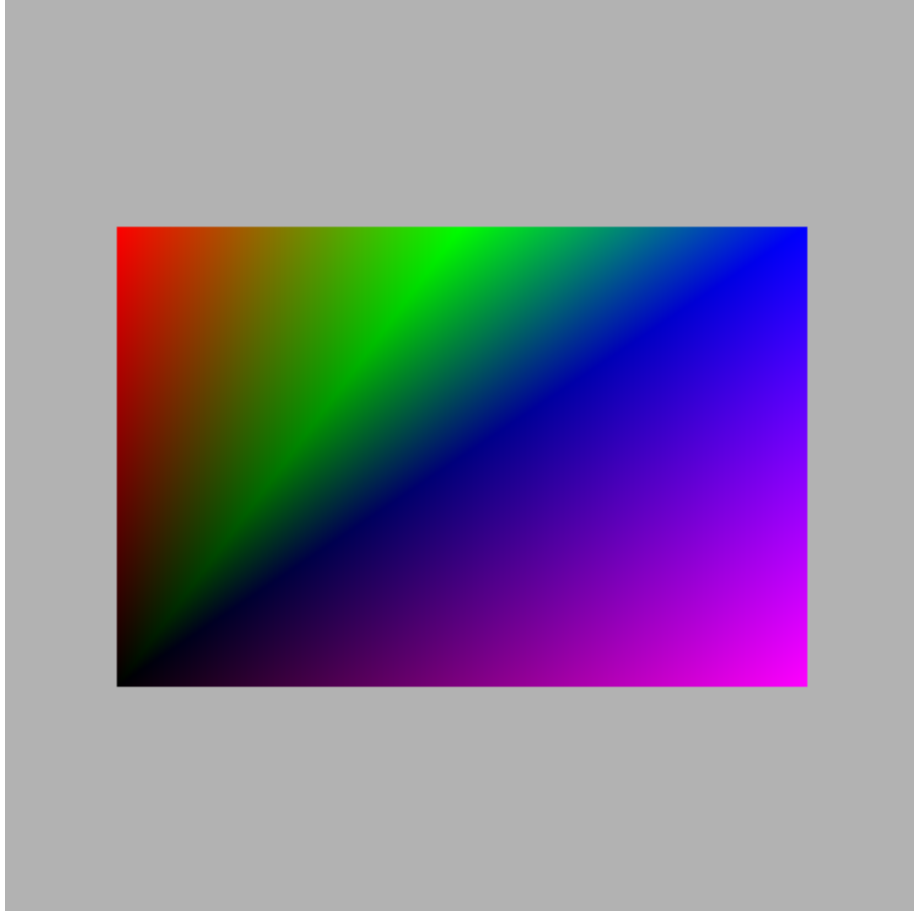
0.7.2 LINE STRIP



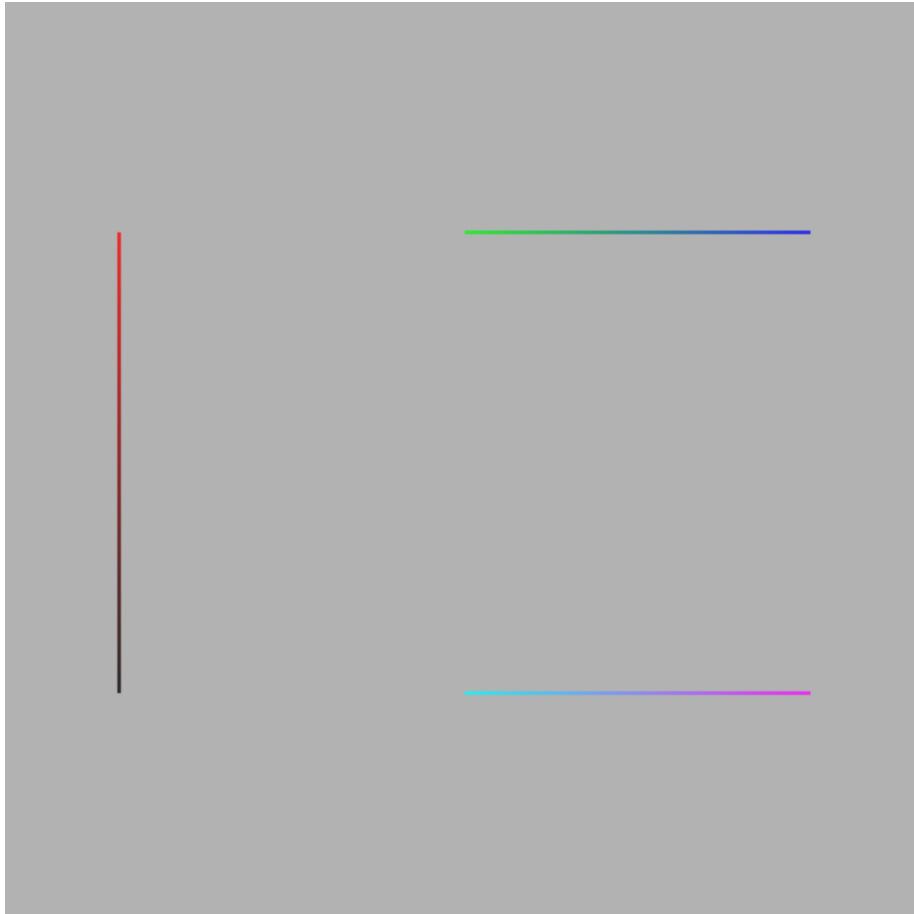
0.7.3 LINE LOOP



0.7.4 TRIANGLE STRIP



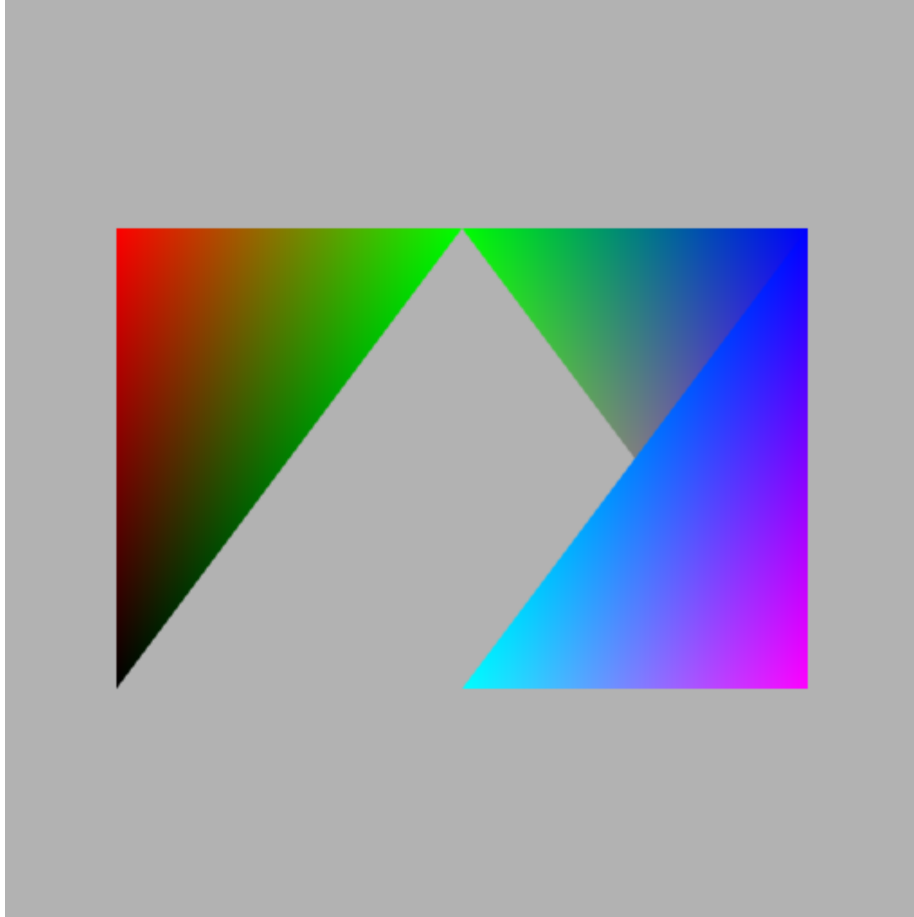
0.7.5 LINES



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 than objects which are behind, or Depth Checking.

Part VIII

Question Eight: Lighting Geometry

0.12 Question

0.13 Answer

Part IX

Question Nine: Texture Sampling Modes

0.14 Question

0.15 Answer

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.