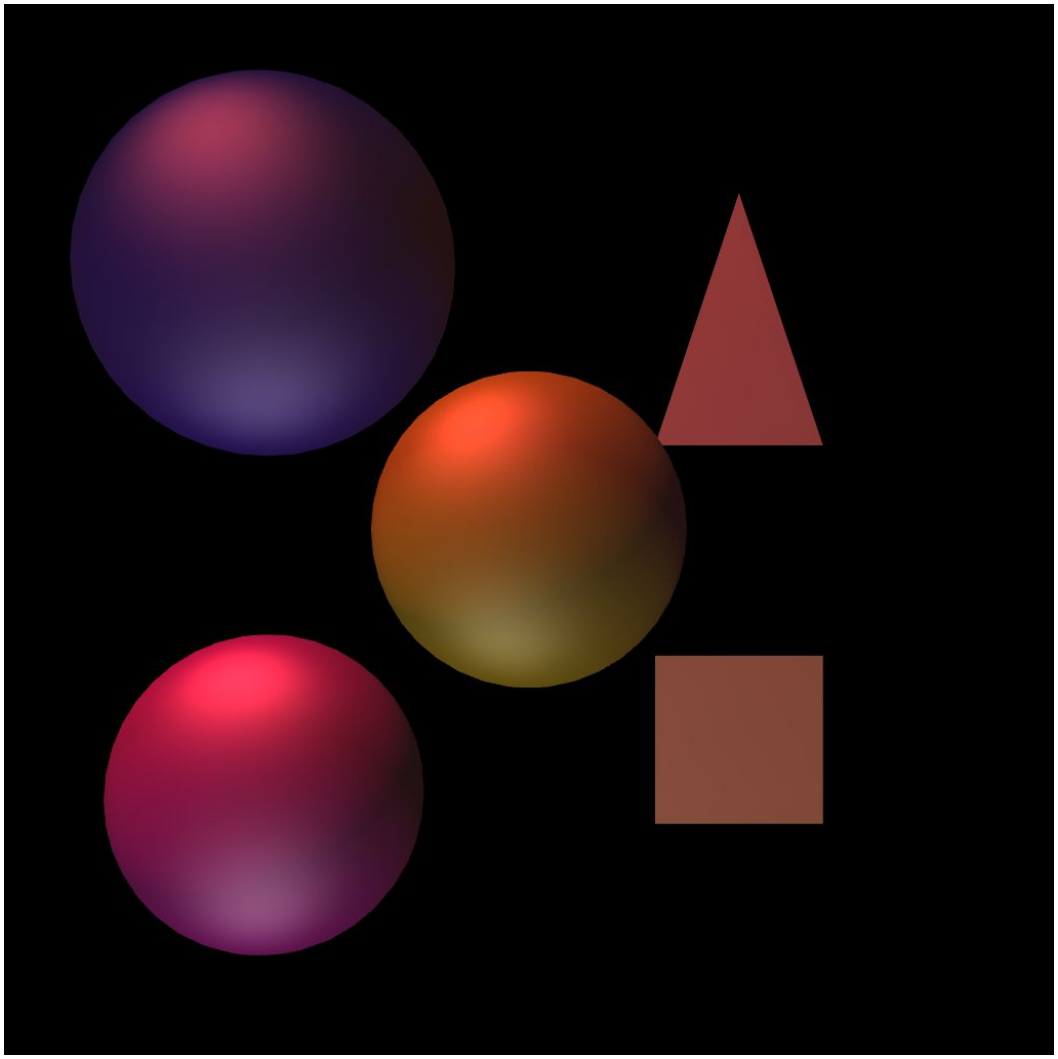
Read me

Program 2, 2017: rasterization

https://kunmiaoyang.github.io/prog2/



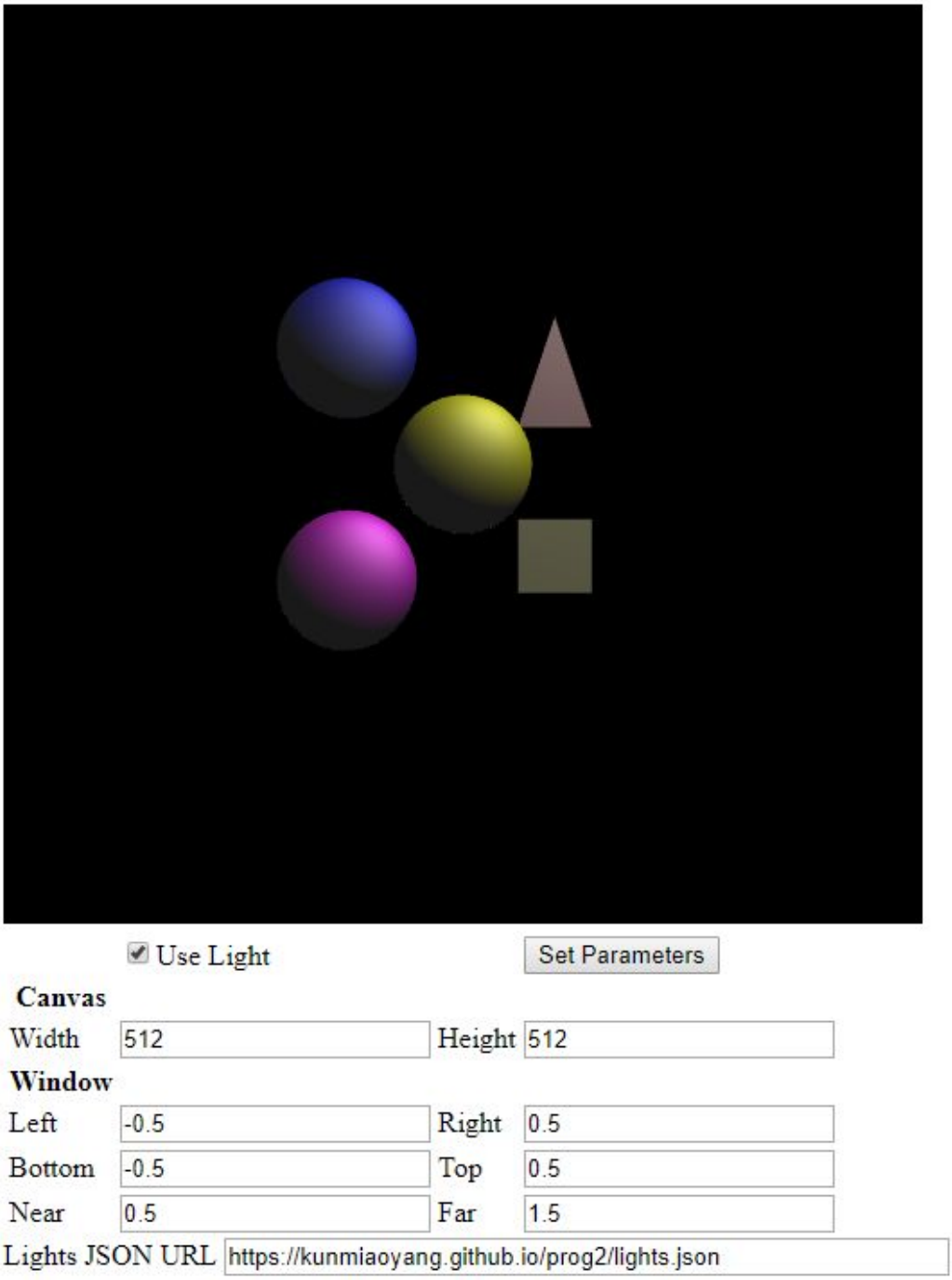
Kunmiao Yang

10.12.2017

CSC 561

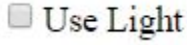
USER INTERFACE

The page UI looks like this:

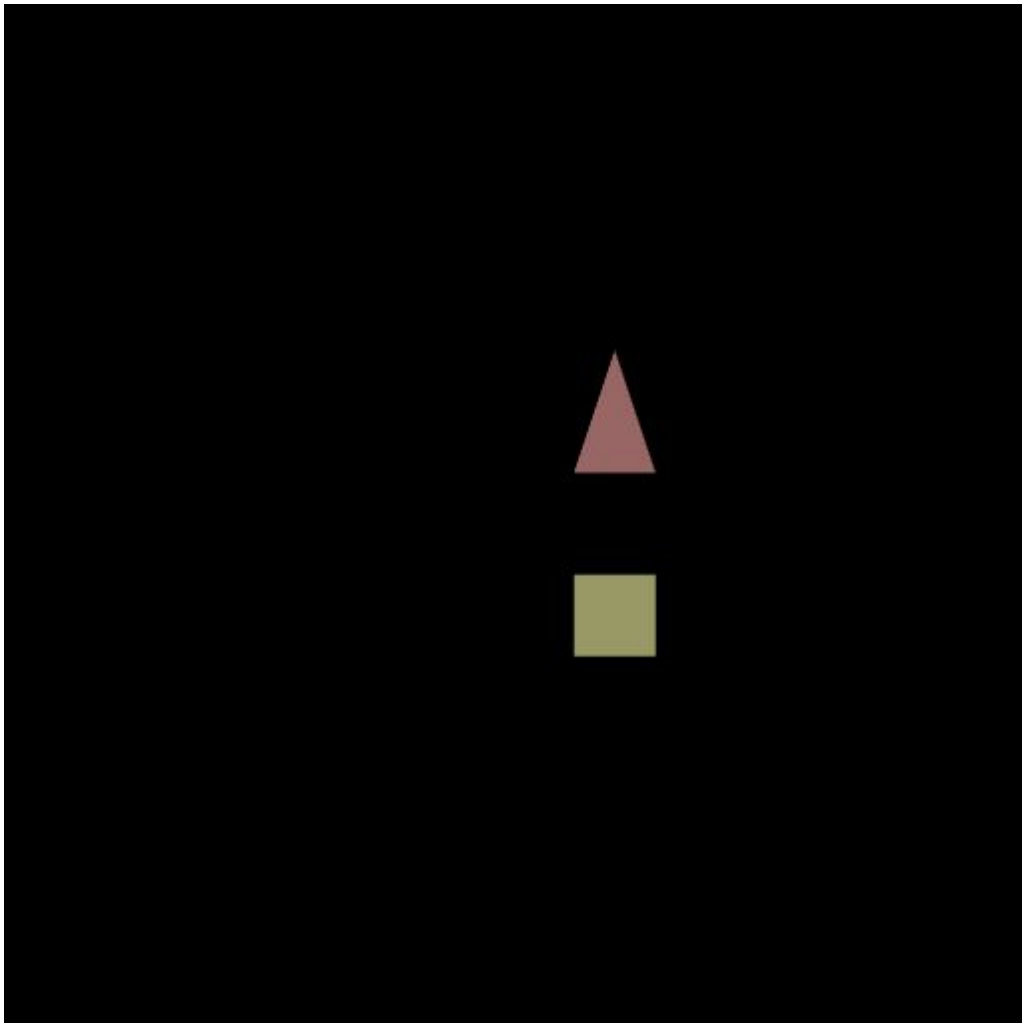


After setting all the parameters, click the **Set Parameters** button to refresh the image.

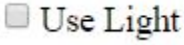
Part 1: Render the input triangles, without lighting



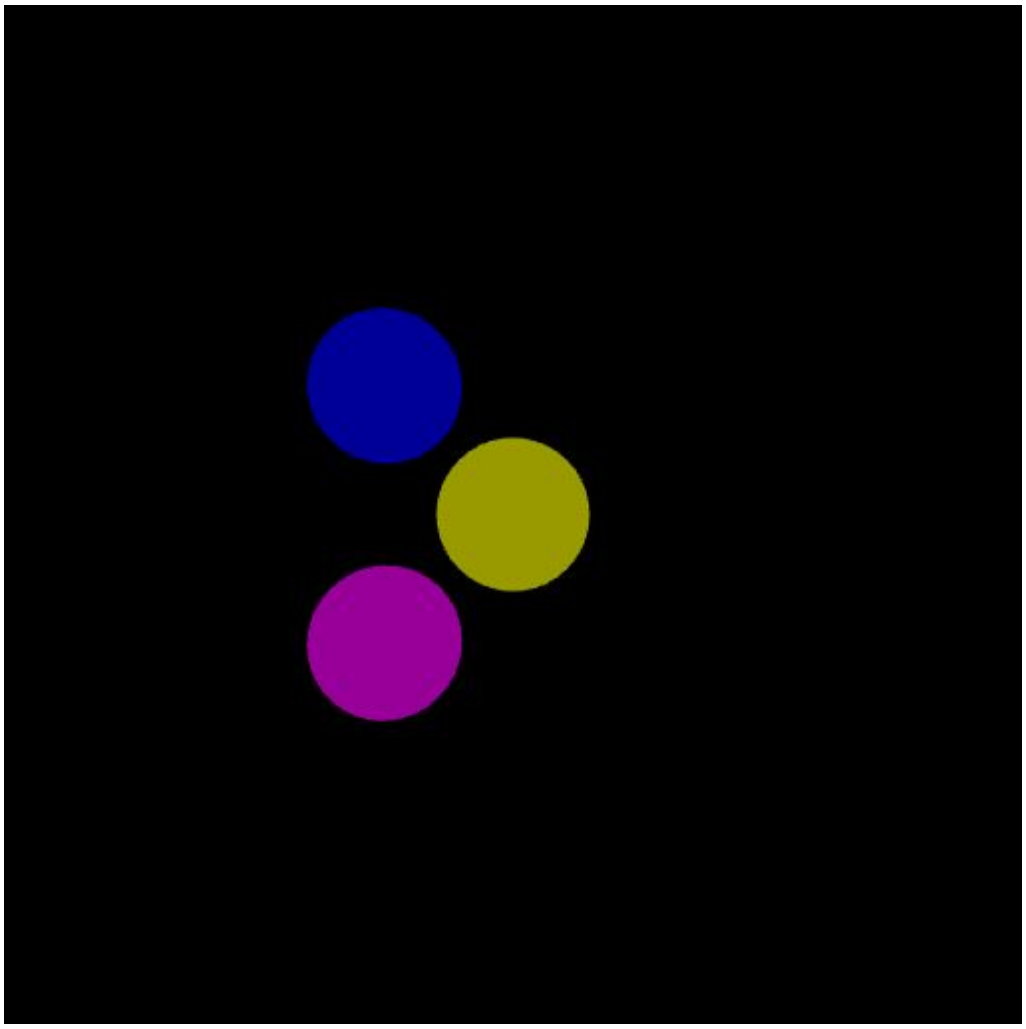
To render triangles without lighting, please uncheck the **“Use Light”** checkbox under the canvas. The color of every pixel of the triangles will be the unmodified diffuse color defined in the material, as shown below.



Part 2: Render the input ellipsoids, without lighting

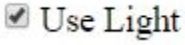


To render ellipsoids without lighting, please uncheck the **“Use Light”** checkbox under the canvas. The ellipsoids will rendered as the figure below.

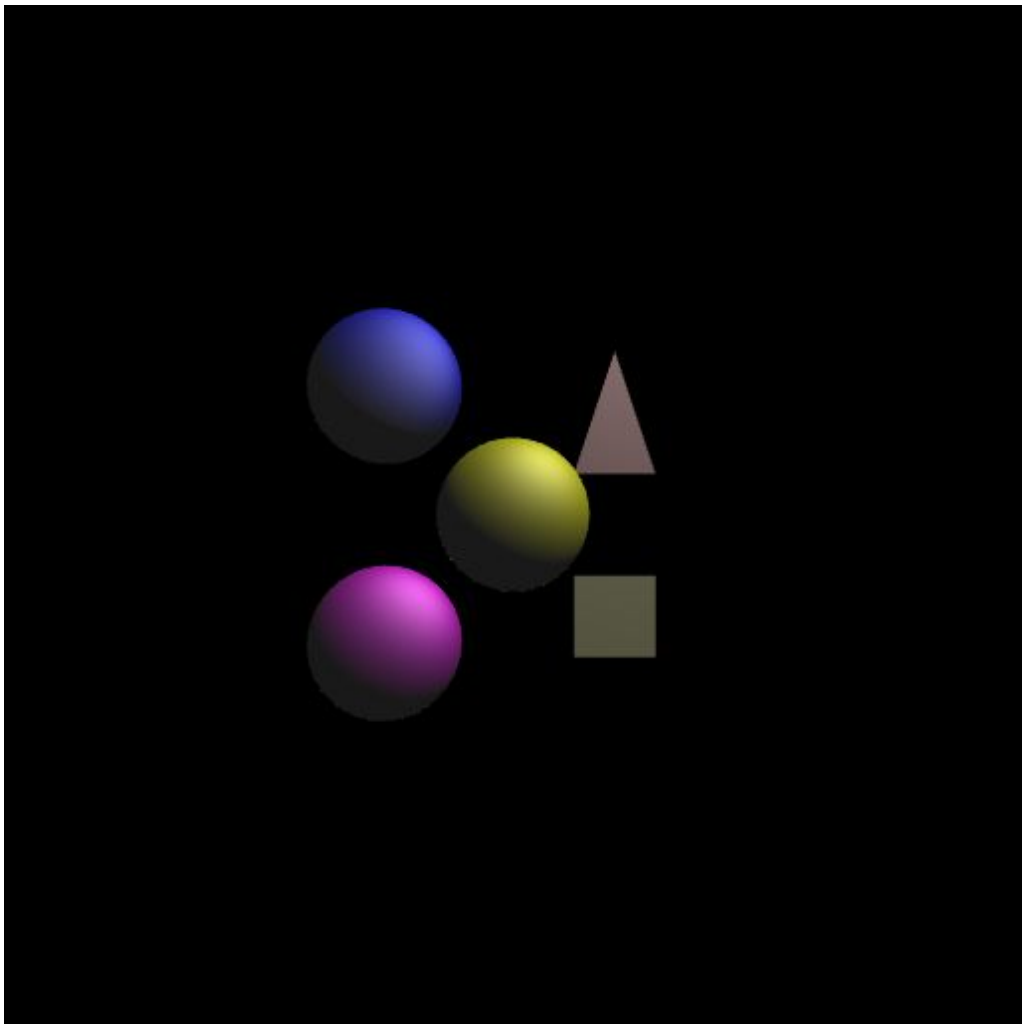


I used the latitude and longitude model to divide the ellipsoids into triangles. I divided the latitude into 20 pieces and longitude 40 pieces.

Part 3: Light the ellipsoids and triangles



To light the ellipsoids and triangles, please check the **“Use Light”** checkbox under the canvas. The ellipsoids and triangles will rendered as the figure below.



Part 4: interactively change view

Use the following key to action table to enable the user to change the view:

a and d — translate view left and right along view X

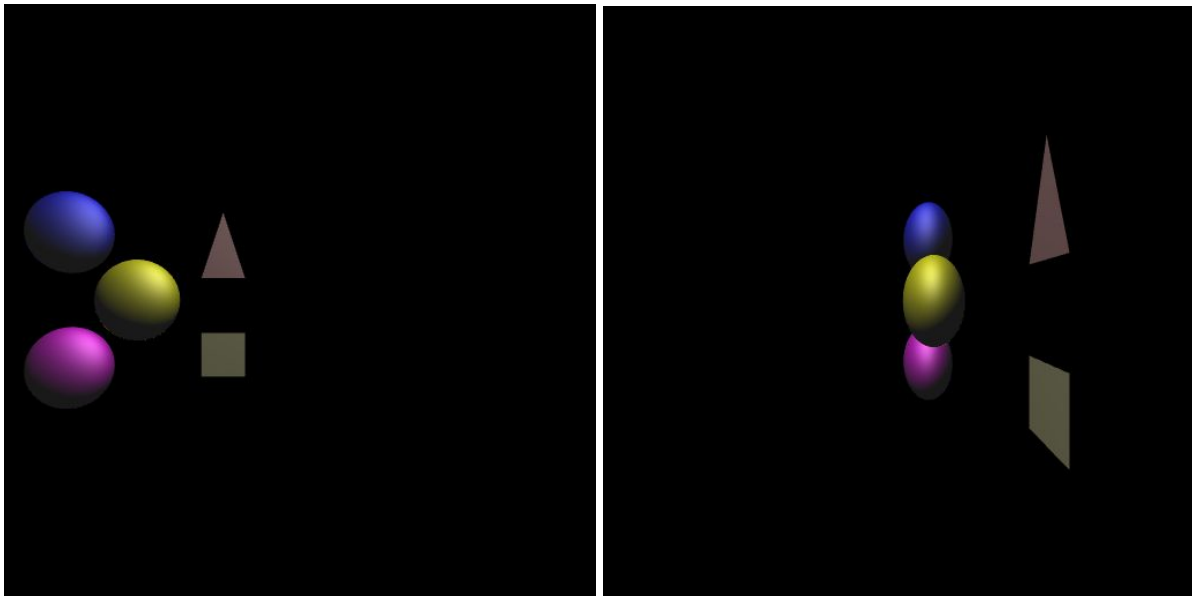
w and s — translate view forward and backward along view Z

q and e — translate view up and down along view Y

A and D — rotate view left and right around view Y (yaw)

W and S — rotate view forward and backward around view X (pitch)

For example, when you press d to translate view right, the image will move left.



Left figure: when translate view right;

Right figure: walk to the right side of the ellipsoids and triangle sets.

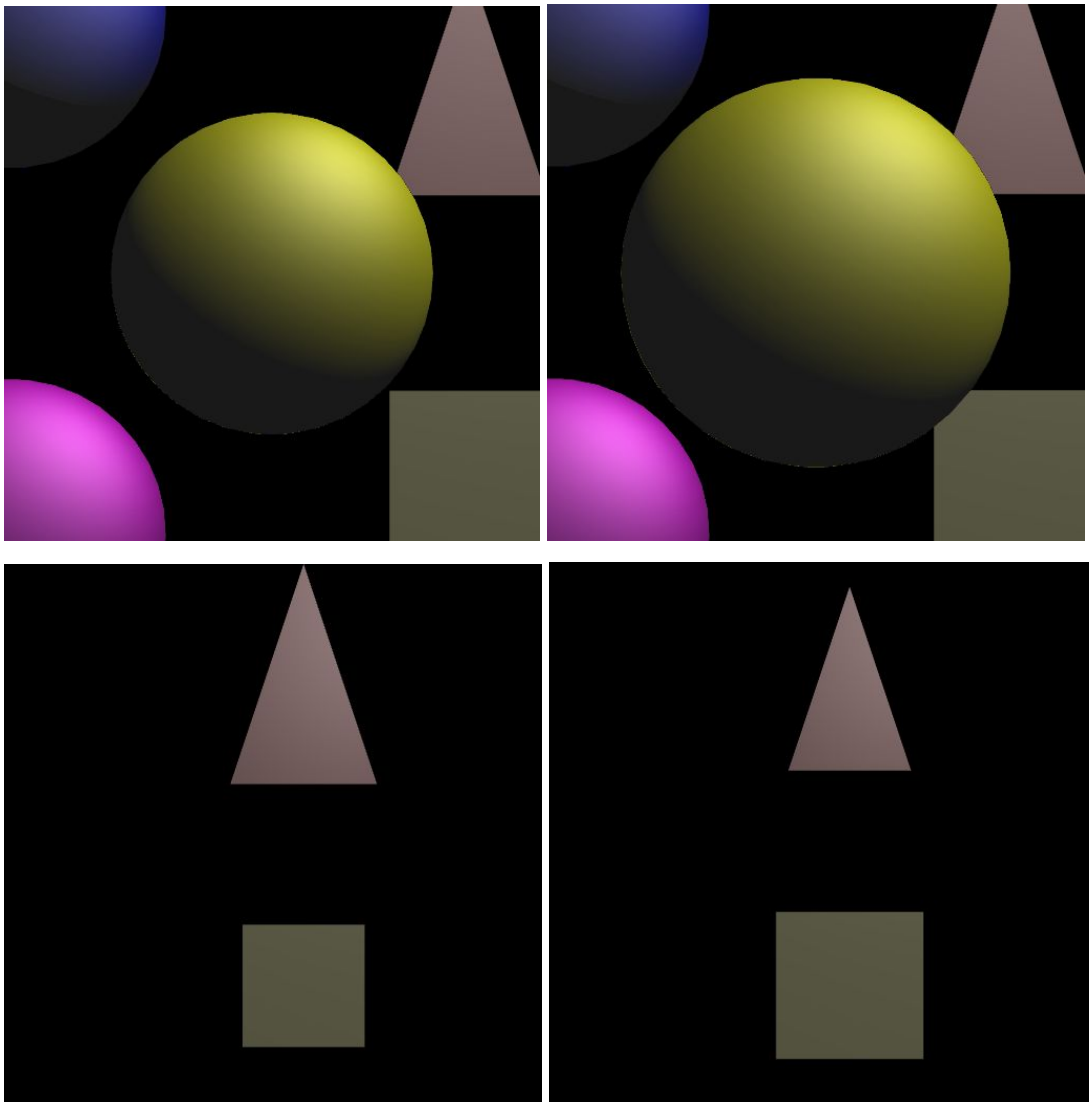
Part 5: Interactively select a model

Use the following key to action table to interactively select a certain model:

left and right — select and highlight the next/previous triangle set (previous off)

up and down — select and highlight the next/previous ellipsoid (previous off)

space — deselect and turn off highlight



Top left figure: figure: no selection;
Bottom left figure: select triangle;

Top right figure: select yellow ellipsoid;
Bottom right figure: select square.

Part 6: Interactively change lighting on a model

Use the following key to action table to interactively change lighting on the selected model:

b — toggle between Phong and Blinn-Phong lighting

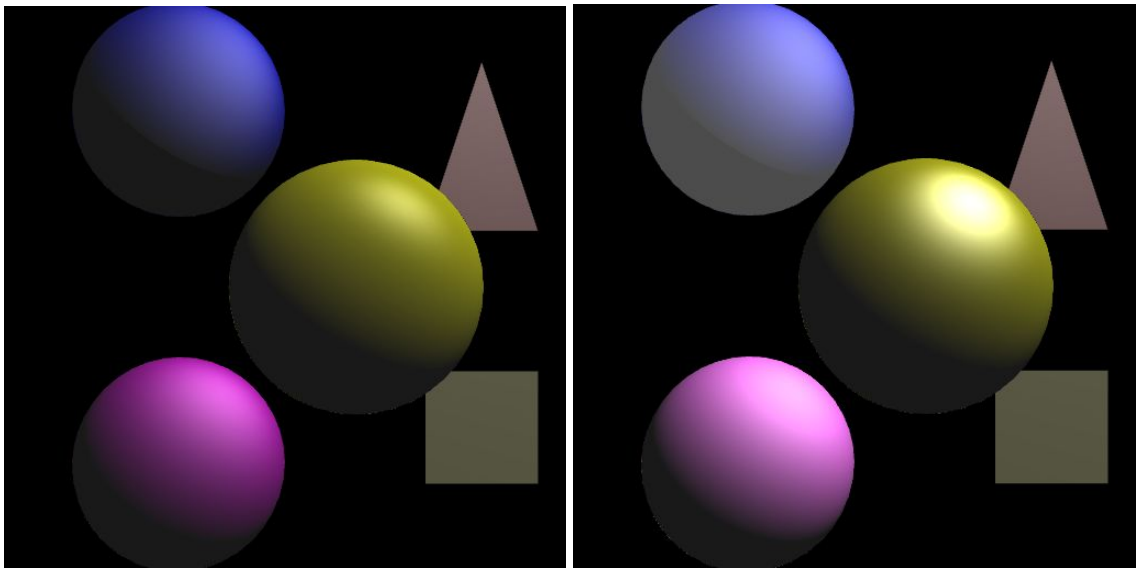
n — increment the specular integer exponent by 1 (wrap from 20 to 0)

1 — increase the ambient weight by 0.1 (wrap from 1 to 0)

2 — increase the diffuse weight by 0.1 (wrap from 1 to 0)

3 — increase the specular weight by 0.1 (wrap from 1 to 0

The default lighting of each model is Blinn-Phong lighting.



Left figure: toggle Phong lighting on yellow ellipsoid

Right figure: increased the specular integer and specular weight of yellow ellipsoid;
increased the ambient weight of blue ellipsoid;
increased the diffuse weight of the purple ellipsoid.

Part 7: Interactively transform models

Use the following key to action table to interactively transform the selected model:

k and ; — translate selection left and right along view X

o and l — translate selection forward and backward along view Z

i and p — translate selection up and down along view Y

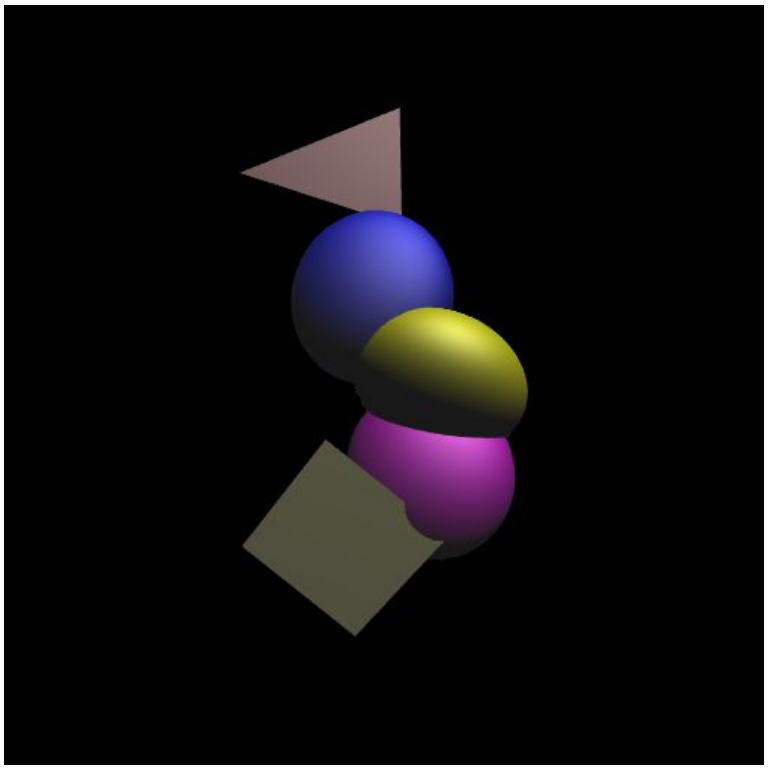
K and : — rotate selection left and right around view Y (yaw)

O and L — rotate selection forward and backward around view X (pitch)

I and P — rotate selection clockwise and counterclockwise around view Z (roll)

The actual transform also depends on the view direction. If user used the A D W S rotated the view, the transform of the selected model will translate along the new view axis. When no model is selected, this transform will not respond.

You can translate the objects to any position and degree, the figure below is an example.

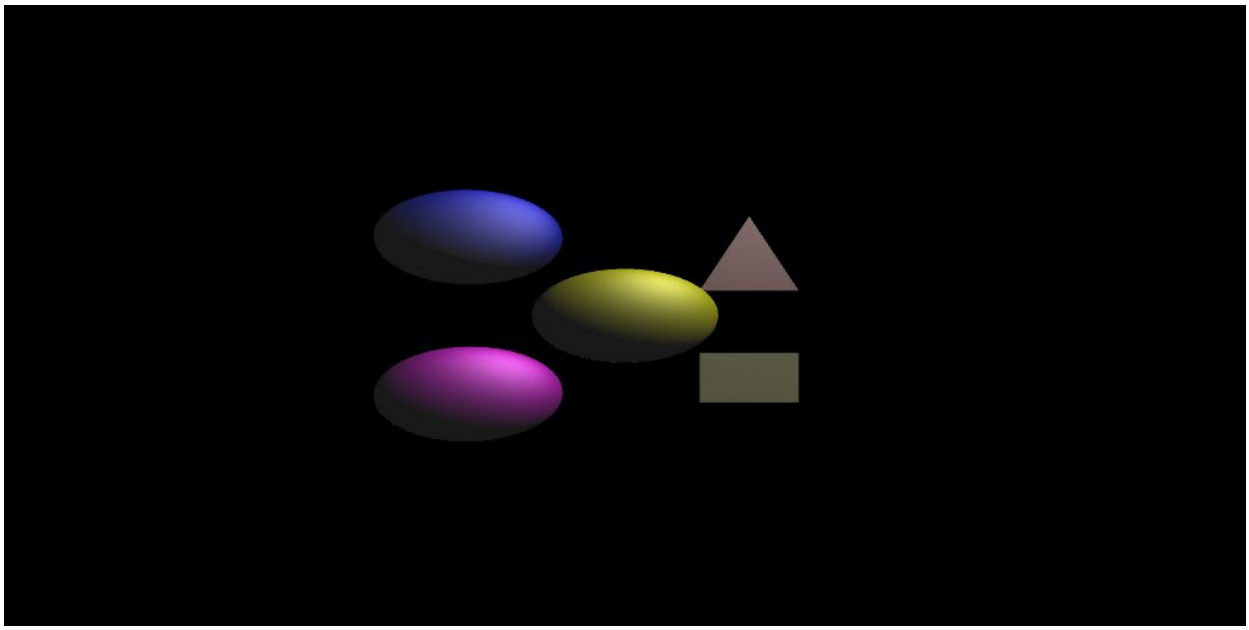


Extra credit: Arbitrarily sized viewports

The width and height of the viewport can be set on the web page below the image:

After setting the width and height, please click the **Set Parameters** button to apply.

When setting the width and height, if their ratio is inconsistent to the ration of window, the image would appear to be stretched. For example, if I set width to 1024 and height to 512, I would get an image as the figure below:



In the next chapter, I will talk about how to set the window width and height, which could make the image normal again.

Extra credit: Support off-axis and rectangular projections

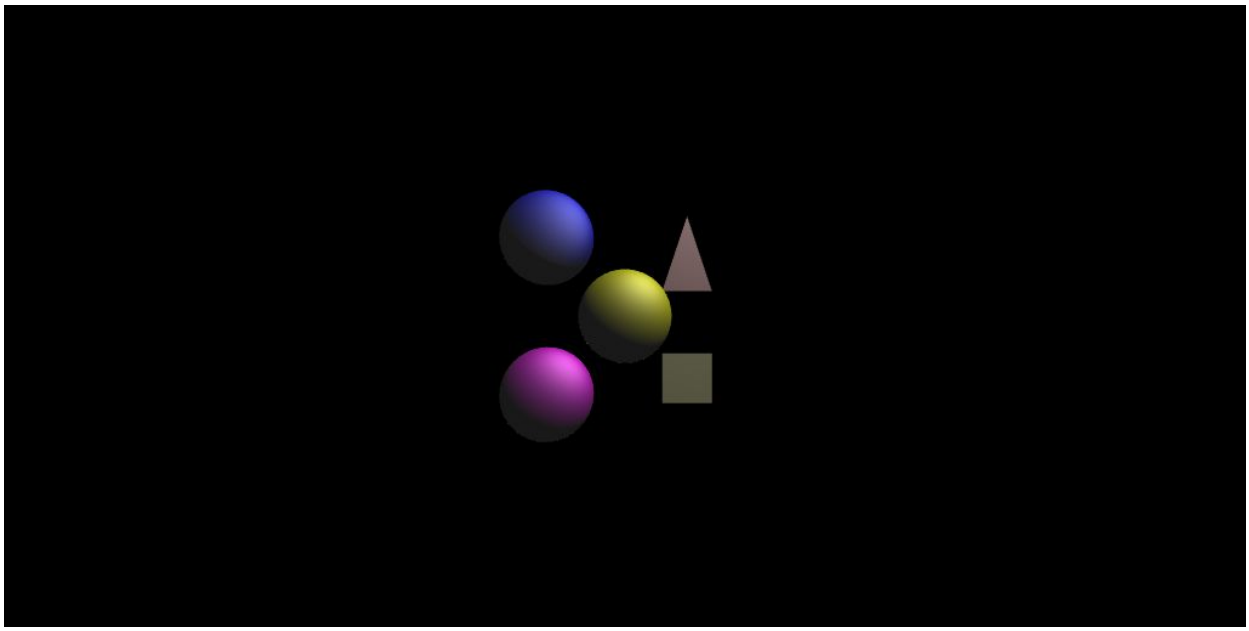
Besides left, right, top and bottom, I also added near and far for the window so as to define the whole box. Note that the near and far parameter are the distance from camera rather than the coordinate in viewing coordinates, thus they are positive. To define the default window used in this program, set the parameters on the page as follows:

Window

Left	-1	Right	1
Bottom	-0.5	Top	0.5
Near	0.5	Far	1.5

After setting the width and height, please click the **Set Parameters** button to apply.

As an example, assume the width and height of the viewport are set to 1024 and 512 respectively. I set the left and right edge of the window to -1 and 1, and then I can make the stretched image in the previous chapter normal again, since the ratio of the width and height become (1- (-1))/(0.5-(-0.5) = 2 which equals that of the viewport. The figure below shows the result.



Extra credit: Multiple and arbitrarily located lights

The program read the parameters from the JSON file to initialize the lights. To set the lights, input the JSON file URL as shown in the figure above. Then click the **Set Parameters** button to apply.

To test multiple lights, I defined a two lights JSON file, its URL is:

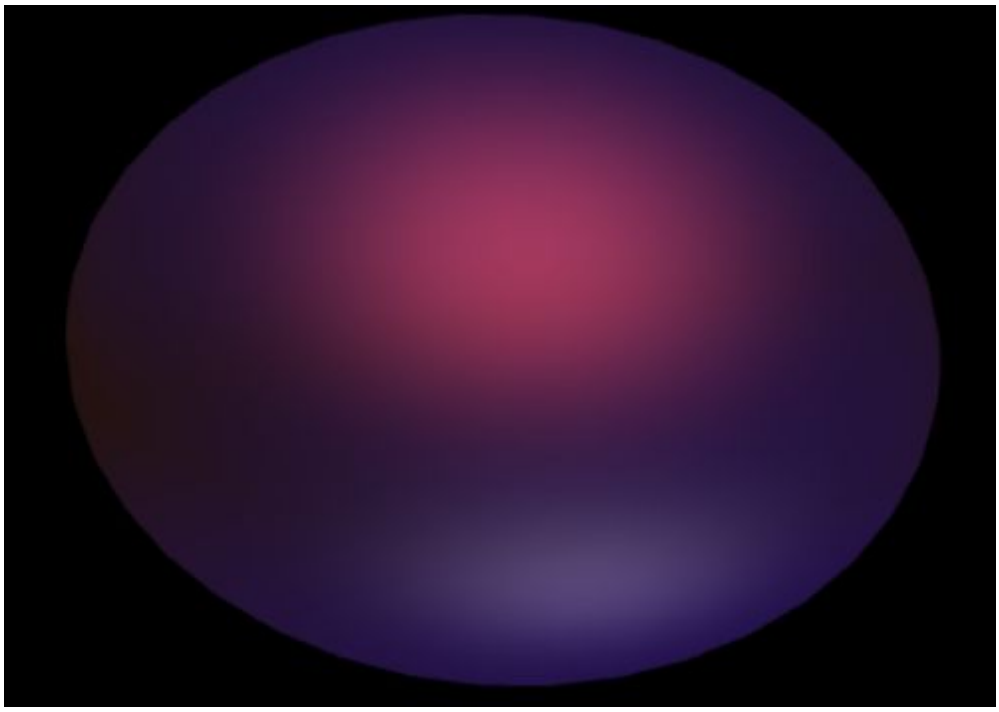
https://kunmiaoyang.github.io/prog2/dualLights.json

I also increased the specular RGB and integer to enhance the effect. Then the blue ellipsoid looks like this:

The first light is stronger and its color is close to red. It creates a purple specular on top of the blue ellipsoid. The second light is white, so its specular is also white on the bottom.

Extra credit: Smooth shading with vertex normals

As you might notice that there are no faceted silhouettes and surfaces on my ellipsoids, as shown below. That's because I smoothed the shading with vertex normals.



When implementing per-fragment shading, I used varying vec3 vTransformedNormal to interpolate the vertex normal. And in the fragment shader, I used these interpolated normal to calculate the color.

When the model is rotated, the nomal vertex is rotated by multiplying an inverse transpose matrix of the modeling transform matrix. I calculated it by using mat3.normalFromMat4 function.

When modeling an ellipsoid, the calculation of normal vector is a little different from modeling a ball. On a ball surface whose center is origin, the normal vector equals the coordinates of the point. However, on an ellipsoid surface, this is not true. I calculated the normal of the surface by the parametric coordinates below:

$$(\cos\phi * \sin\theta / a, \quad \cos\theta / b, \quad \sin\phi * \sin\theta / c)$$