Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Lab04 Report: 3D Object Transformation – Texture Mapping

Program Description

The key idea is to practice constructing 3D models based on primitive geometry (triangles and quads) with OpenGL draw modes such as GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN and GL_QUAD_STRIP; practice texture's UV mapping on each vertex; and practice simple animation such as translation and rotation around object's local axis.

Each shape's render() function takes parameters including that shape's dimensions (***height, big radius, small radius***), position (mouse-click x, y, ***depth***), ***texture ID*** and ***rotation per time unit***. Where only mouse-click x, y are user's input, the rest are randomized every time new shape is created.

User can toggle between wireframe mode and fill mode, paused rotation, reset rotation and resume rotation.

Usage instruction:

- Implemented shapes: **Circular Plane, Cube, Pyramid, Sphere, Cylinder, Cone, Torus, Paraboloid** and **Hyperboloid**.
- Program draws Cube by default.
- To draw other shapes:
  - Right-click menu, choose an item.
  - Left-click anywhere on the windows to draw shape at such x, y position (windows' x, y are remapped to GL coordinates); or
  - Press key *n* or *N* to render such new shape at (0, 0, ***z***).
- To change between WIREFRAME mode and FILL mode:
  - Right-click menu to choose; or
  - Press key *w* or *W* to toggle.
- To reset objects to its initial rotation
  - Press key r or R (stands for *reset rotation*)
- To pause objects' rotation
  - Press key space bar to toggle
- To add more textures
  - Put .jpg files in Textures/ folder with file name convention: tex<next-id>.jpg, e.g: tex3.jpg, tex22.jpg
  - Update file *total-number-of-textures.txt* with new number of textures

Key implementation steps of each object:

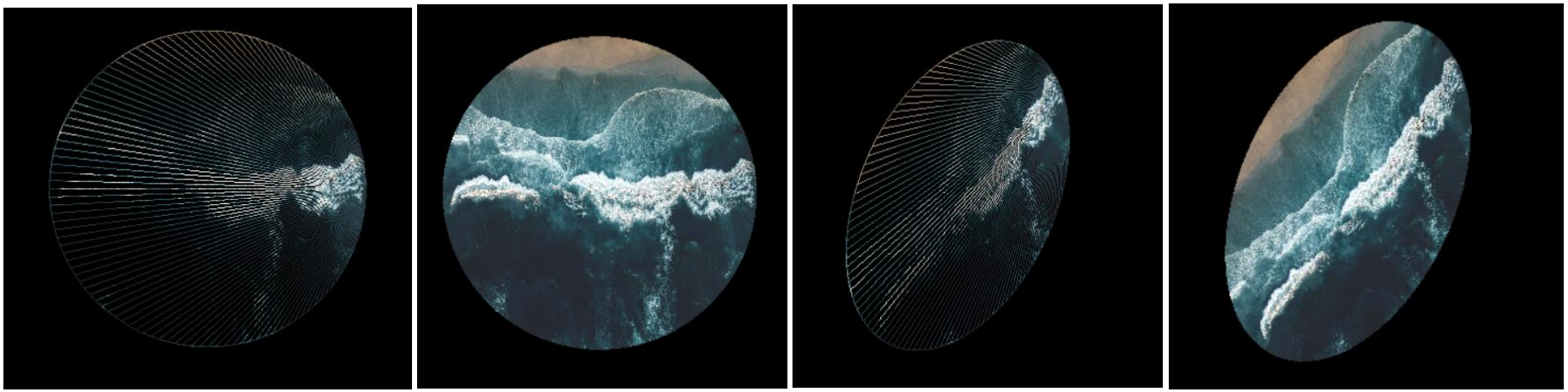
1. *Constructor ()*
   - Get random value of dimension, position, rotation and ID number of the texture (texture ID)
   - Check if images have been loaded into texture array, if not, load the array. Then retrieve the texture with such texture ID
   - *Calculate vertices (x,y,z,u,v) and store into std::vector (I do this in one shape in the hope that it should render faster).*
   - Output shape's information to console.
2. Render ()
   - Reset model-view matrix.
   - Translate object to random **depth**.
   - Rotate object with current value of rotation angle. If reset-rotation is on, then don't do rotation.
   - Calculate and draw vertices position (x, y, z) and texture mapping (u, v); or
   - *Draw vertices position (x, y, z) and texture mapping (u, v) based on the data already calculated in constructor*.
   - Update rotation angle for the next frame. If pause-rotation is on, then increase rotation angle by zero.

Load images into texture array:

- Read from a text file one integer which is the total number of textures in the *Textures/* folder.
- First iteration: load images into texture array using SOIL library with flag `SOIL_FLAG_POWER_OF_TWO | SOIL_FLAG_INVERT_Y` to preserve texture colors and invert V mapping.
- Second iteration: check if any texture in the array is null. If there is one, replace that slot with a texture that works (fallback to a texture).
- Texture generation using data from the bitmap (`glTexParameteri()`)

Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Circular Plane



Initialized (wireframe) | Initialized (fill) | Animated (wireframe) | Animated (fill)

Code:

```
GLfloat X, Y, Z, midTex = 0.5f;
glBegin(GL_TRIANGLE_FAN);
for (GLfloat i = 0; i <= 2 * PI; i += 2 * PI / total) {
    X = r * cos(i);
    Y = r * sin(i);
    Z = -0.05f;
    glTexCoord2f(midTex + midTex * cos(i), midTex + midTex * sin(i));
    glVertex3f(X, Y, Z);
}
glEnd();
```

Explain:

- A circular segment of an image origins at the image's center will be used as texture (fig 3)
- Draw triangle fan mode (fig 2)
- Increase the angle by an amount of 2*PI/total_segments
- X is the Ox projection of R in Cartesian coordinates (fig 1)
- Y is the Oy projection of R in Cartesian coordinates (fig 1)
- Z is hardcoded by small amount (less than 0, or else it is out-of-laptop-screen)
- Apply same order of vertices to texture UV, radius of texture is 0.5 (half width, height of texture)
- Draw such vertex

Graphical interpretation:

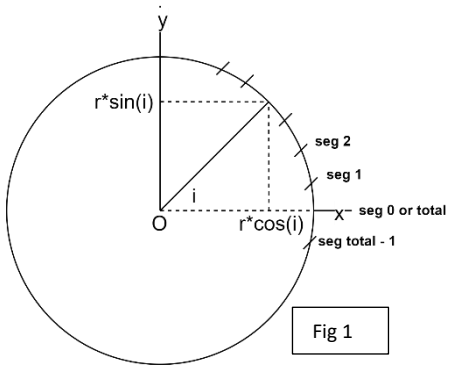


Fig 1

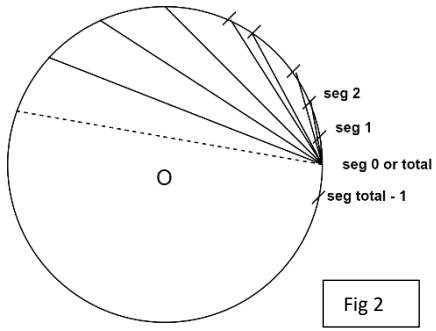


Fig 2

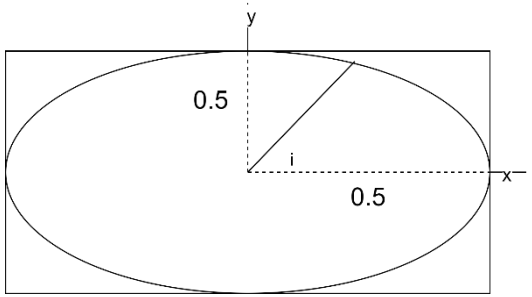
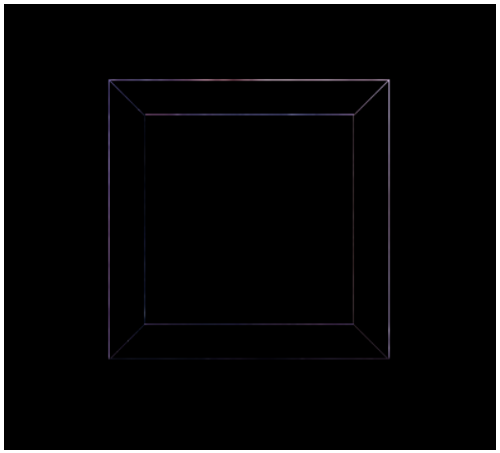


Fig 3

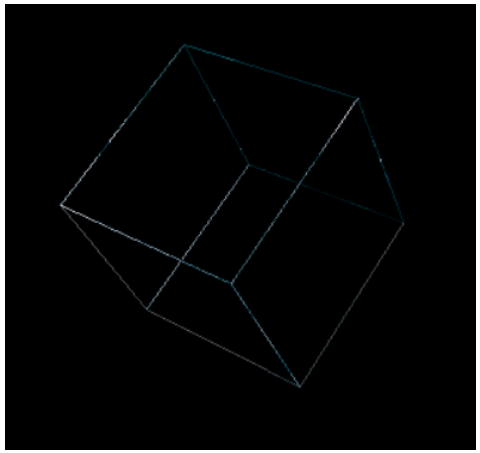
Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Cube

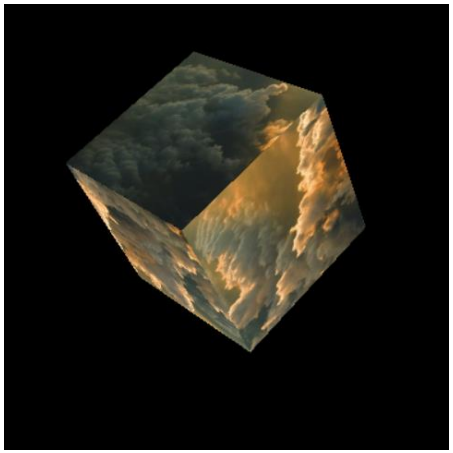


Initialized (wireframe)

Initialized (fill)



Animated (wireframe)



Animated (fill)

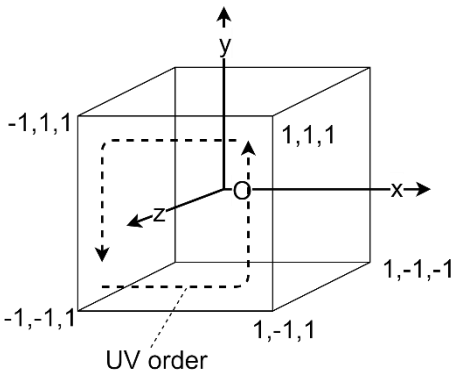
Code:

```
glBegin(GL_QUADS);
// Front Face
glTexCoord2f(0.0f, 0.6f); glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(0.0f, 0.3f); glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(0.25f, 0.3f); glVertex3f(1.0f, 1.0f, 1.0f);
glTexCoord2f(0.25f, 0.6f); glVertex3f(-1.0f, 1.0f, 1.0f);
// Top Face
glTexCoord2f(0.6f, 0.25f); glVertex3f(-1.0f, 1.0f, 1.0f);
glTexCoord2f(0.3f, 0.25f); glVertex3f(1.0f, 1.0f, 1.0f);
glTexCoord2f(0.3f, 0.5f); glVertex3f(1.0f, 1.0f, -1.0f);
glTexCoord2f(0.6f, 0.5f); glVertex3f(-1.0f, 1.0f, -1.0f);
// Back Face
glTexCoord2f(0.5f, 0.6f); glVertex3f(-1.0f, 1.0f, -1.0f);
glTexCoord2f(0.5f, 0.3f); glVertex3f(1.0f, 1.0f, -1.0f);
glTexCoord2f(0.75f, 0.3f); glVertex3f(1.0f, -1.0f, -1.0f);
glTexCoord2f(0.75f, 0.6f); glVertex3f(-1.0f, -1.0f, -1.0f);
// Bottom Face
glTexCoord2f(0.75f, 0.6f); glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(0.75f, 0.3f); glVertex3f(1.0f, -1.0f, -1.0f);
glTexCoord2f(1.0f, 0.3f); glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 0.6f); glVertex3f(-1.0f, -1.0f, 1.0f);
// Right face
glTexCoord2f(0.0f, 0.3f); glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
glTexCoord2f(0.25f, 0.0f); glVertex3f(1.0f, 1.0f, -1.0f);
glTexCoord2f(0.25f, 0.3f); glVertex3f(1.0f, 1.0f, 1.0f);
// Left Face
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(0.0f, 0.6f); glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(0.25f, 0.6f); glVertex3f(-1.0f, 1.0f, 1.0f);
glTexCoord2f(0.25f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
glEnd();
```

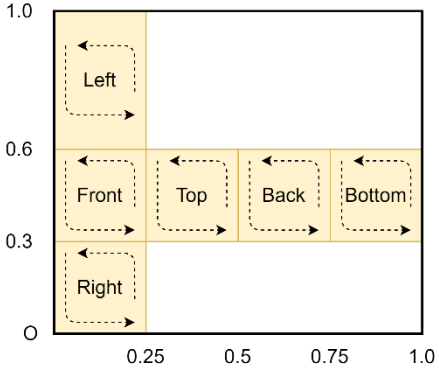
Graphical interpretation:

Note that occluded edges of the cube should be mathematically correctly drawn in dashed line.

Putting vertices and uv order

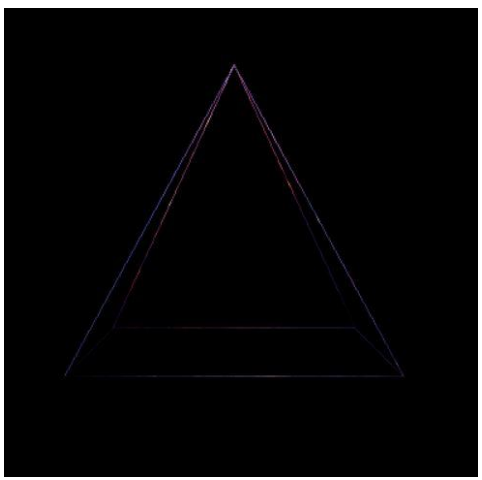


UV ordering

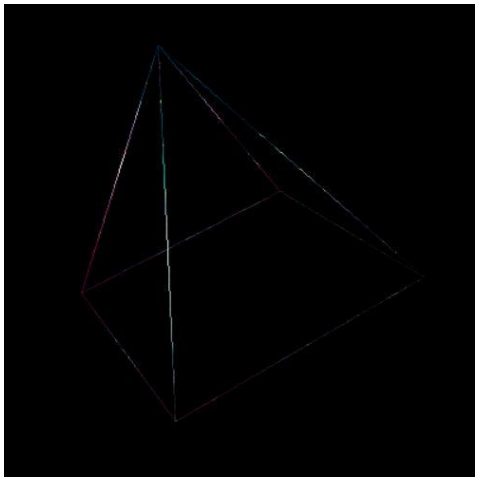


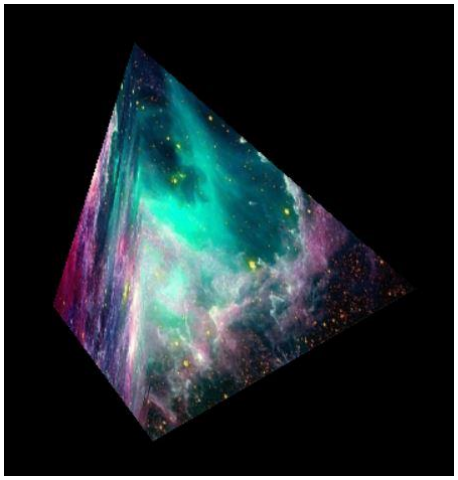
Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Pyramid


Initialized (wireframe)


Initialized (fill)


Animated (wireframe)


Animated (fill)

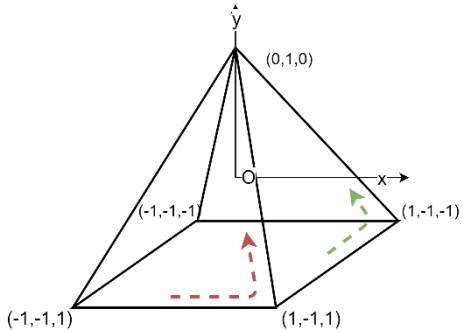
Code:

```
glBegin(GL_TRIANGLE_FAN);
glTexCoord2f(middleTex, ); glVertex3f(0, 1.0f, 0.0f);
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, -1.0f, -1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
glEnd();

glBegin(GL_QUADS);
// Bottom Face
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
glEnd();
```

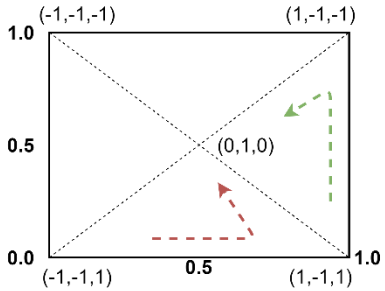
Graphical interpretation:

I let O be at the center of the shape (instead of on the base) so that the whole shape may rotate around its center axis. Note that occluded edges should be drawn in dashed line.

Putting vertices and uv order

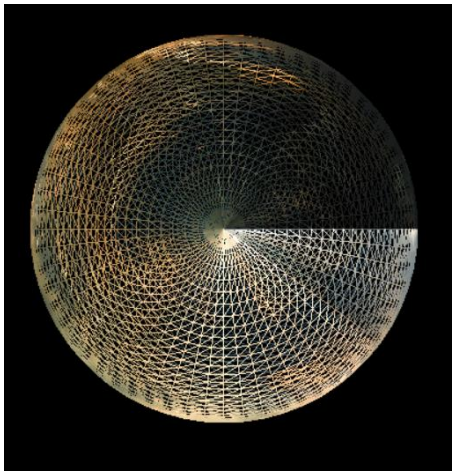


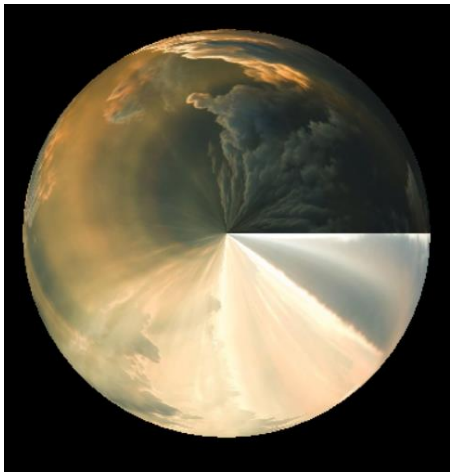
UV ordering

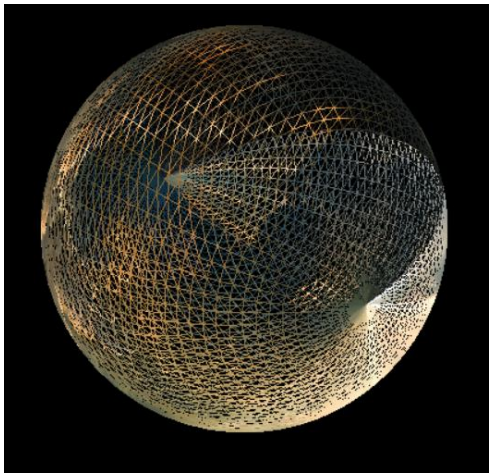


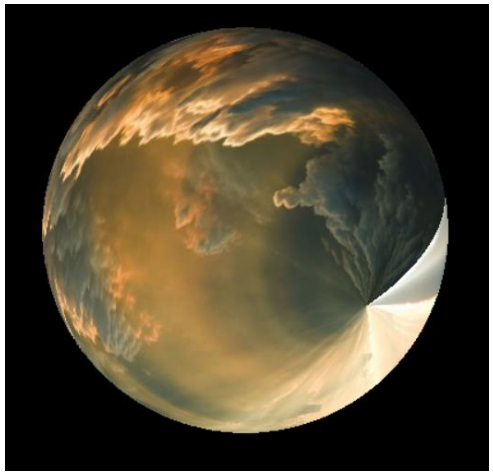
Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Sphere


Initialized (wireframe)


Initialized (fill)


Animated (wireframe)

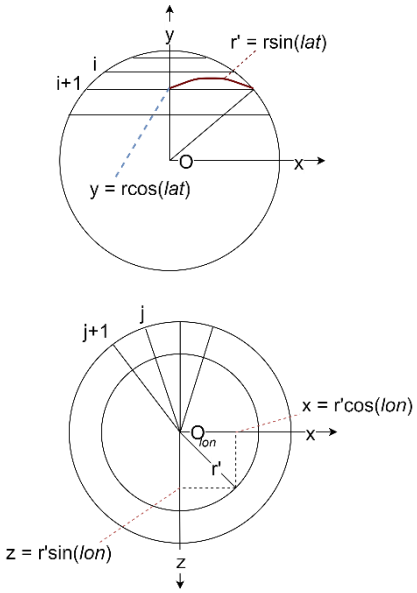

Animated (fill)

For this shape I try to optimize the render speed by pre-calculating the vertices and uv coordinates in its constructor. And only read such data in render function

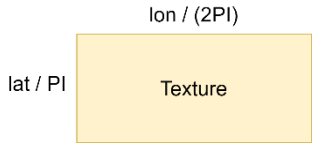
Code:

```
// constructor
for (int i = 0; i < total + 1; ++i) {
    float lon = i * 2 * PI / total;
    for (int j = 0; j < total + 1; ++j) {
        float lat = j * PI / total;
        _x = r * cos(lon) * sin(lat);
        _y = r * sin(lon) * sin(lat);
        _z = r * cos(lat);
        _u = lat / PI;
        _v = lon / PI / 2;
        std::vector<float> point{ _x, _y, _z, _u, _v};
        sphere[i][j] = point;
    }
}

//render()
for (int i = 0; i < total; ++i) {
    glBegin(GL_TRIANGLE_STRIP);
    for (int j = 0; j < total + 1; ++j) {
        glTexCoord2f(sphere[i][j][3], sphere[i][j][4]);
        glVertex3f(sphere[i][j][0], sphere[i][j][1], sphere[i][j][2]);
        glTexCoord2f(sphere[i + 1][j][3], sphere[i + 1][j][4] + 1 / total);
        glVertex3f(sphere[i + 1][j][0], sphere[i + 1][j][1], sphere[i + 1][j][2]);
    }
    glEnd();
}
```

Graphical interpretation:

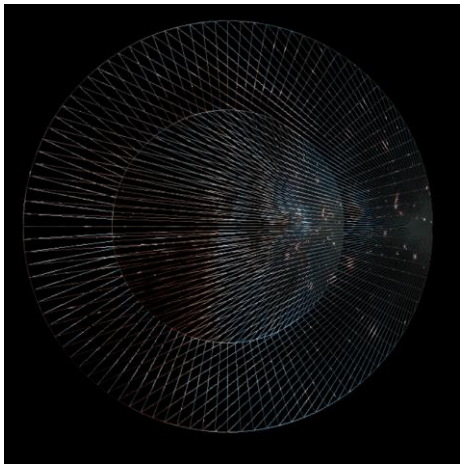


Longitude goes full circle (from 0 to 2PI), latitude only goes half a circle (PI).
For render(), in the same j^{th} loop, two points above **i** and below **i+1** is drawn. Because `GL_TRIANGLE_STRIP` is chosen, in the next $j+1^{th}$, the first above point **i** helps enclose the triangle and the surface made by those three points are drawn.
I think because this is a relatively simple shape, the render speed is not significantly different between pre-calculating and calculating-in-every-frame.

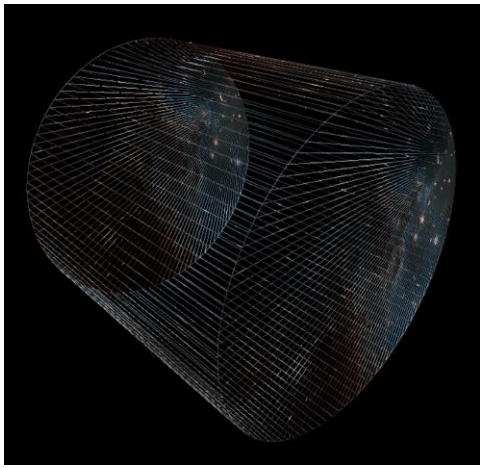


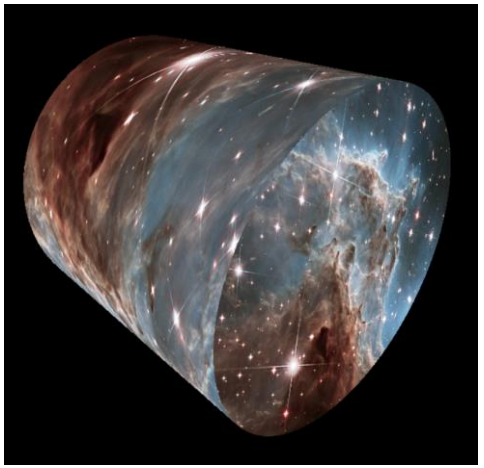
Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Cylinder


Initialized (wireframe)


Initialized (fill)


Animated (wireframe)


Animated (fill)

The two circular caps are drawn as Circular Plane explained above.

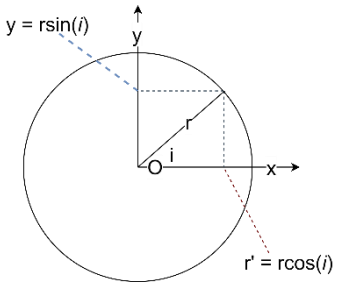
The cylinder is as followed:

Code:

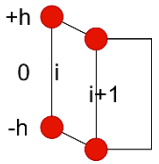
```
glBegin(GL_QUAD_STRIP);
for (float i = 0; i <= 2 * PI + PI / total; i += PI/total) {
    float textCoord = i / (2 * PI);
    glTexCoord2f(textCoord, 1.0);
    glVertex3f(cos(i) * r, sin(i) * r, +h);
    glTexCoord2f(textCoord, 0.0);
    glVertex3f(cos(i) * r, sin(i) * r, -h);
}
glEnd();
```

Graphical interpretation:

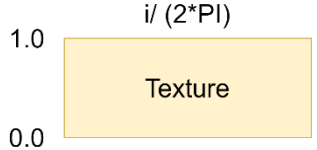
1. The cylinder base x, y is calculated as followed:



2. For render(), in the i^{th} loop, two points above **(x,y,+h)** and below **(x,y,-h)** is drawn. Because `GL_QUAD_STRIP` is chosen, in the next $i+1^{th}$, the two more points **(x',y',+h)** and **(x',y',-h)** helps enclose the quad and the surface made by those four points are drawn.

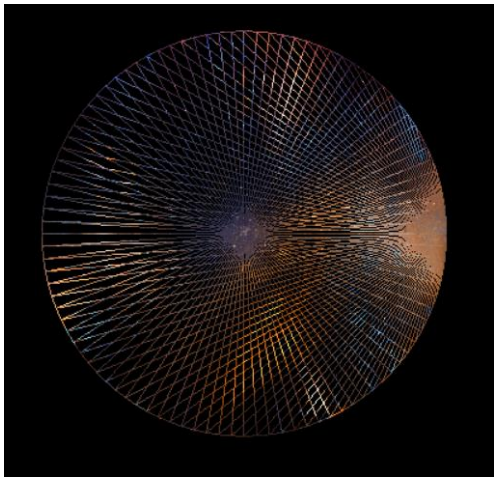


3. Texture mapping, for each step in *u* axis is i/(2*PI)



Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Cone




Initialized (wireframe) | Initialized (fill) | Animated (wireframe) | Animated (fill)

The circular cap is drawn as Circular Plane explained above.

The surround surface is calculated as followed:

Code:

```
GLfloat middleTex = 0.5f;
glBegin(GL_TRIANGLE_FAN);
glTexCoord2f(middleTex, middleTex);
glVertex3f(0, 0, h);
for (float i = -PI / total; i <= 2 * PI; i += PI / total) {
    glTexCoord2f(middleTex * cos(i) + middleTex, middleTex * sin(i)
                + middleTex);
    glVertex3f(r * cos(i), r * sin(i), -h);
}
glEnd();
```

Graphical interpretation:

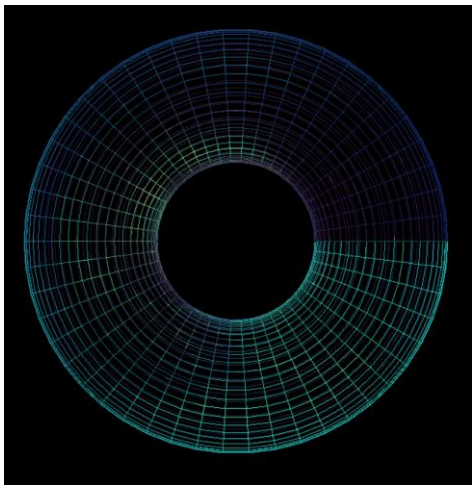
See figure *Animated (wireframe)* above.

Since `GL_TRIANGLE_FAN` mode is used, the origin point (the peak) must be drawn first.

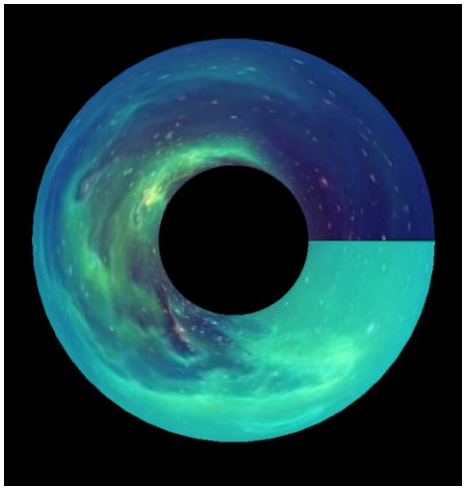
- The first vertex is at (0,0,h) and mapped with the center point of the texture.
- Then points on a circular path are specify within the `GL_TRIANGLE_FAN` mode so that they connect to the origin point to make triangles. This is quite similar to draw a circular plane.
- Texture is mapped the same way as circular plane except now that one more point which is the texture center (0.5, 0.5) is mapped with the peak. I do this way so that the surrounding face have continuous pattern.

Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

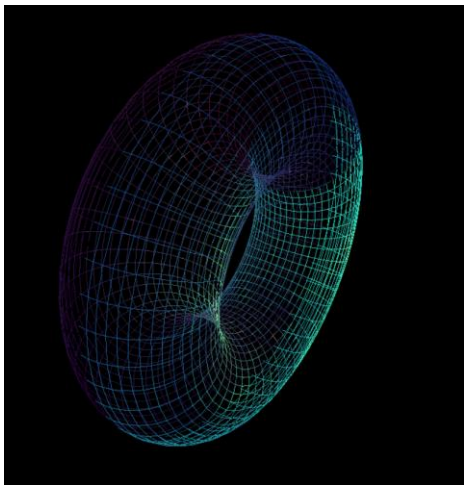
Torus (Ring)



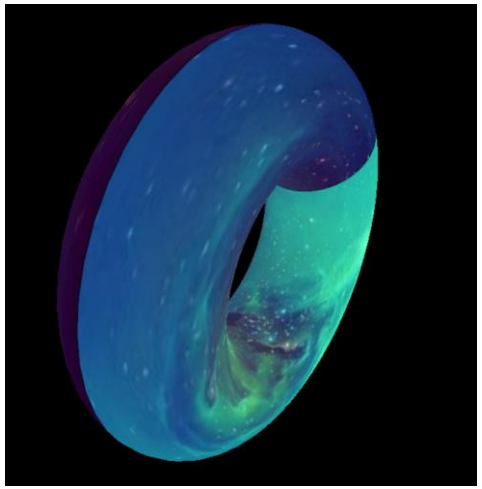
Initialized (wireframe)



Initialized (fill)



Animated (wireframe)



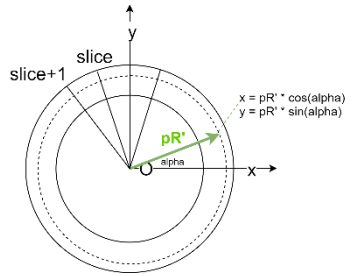
Animated (fill)

Code:

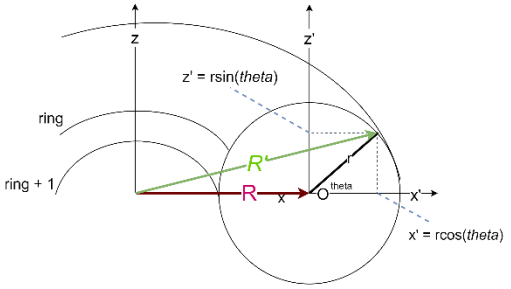
```
for (float slice = 0; slice < total; slice+=1) {
  glBegin(GL_QUAD_STRIP);
  for (float ring = 0; ring <= total; ring+=1) {
    X = (R + r * cos(slice * 2 * PI / total)) * cos(ring * 2 * PI / total);
    Y = (R + r * cos(slice * 2 * PI / total)) * sin(ring * 2 * PI / total);
    Z = r * sin(slice * 2 * PI / total);
    u = slice / total;
    v = ring / total;
    glTexCoord2f(u, v);
    glVertex3f(X, Y, Z);

    X = (R + r * cos((slice + 1) * 2 * PI / total)) * cos(ring * 2 * PI / total);
    Y = (R + r * cos((slice + 1) * 2 * PI / total)) * sin(ring * 2 * PI / total);
    Z = r * sin((slice + 1) * 2 * PI / total);
    u = (slice + 1) / total;
    v = ring / total;
    glTexCoord2f(u, v);
    glVertex3f(X, Y, Z);
  }
  glEnd();
}
```

Graphical interpretation:

1. The initial position shows that the ring facing us is lying on xOy plane like this:



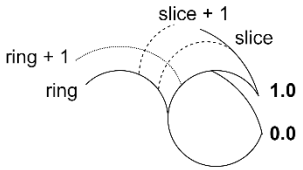
2. The vector ***R'*** actually depends on xOz as followed:



3. The Ox projection of vector ***R'*** is **pR'** calculated as: pR' = R + rcos(theta)

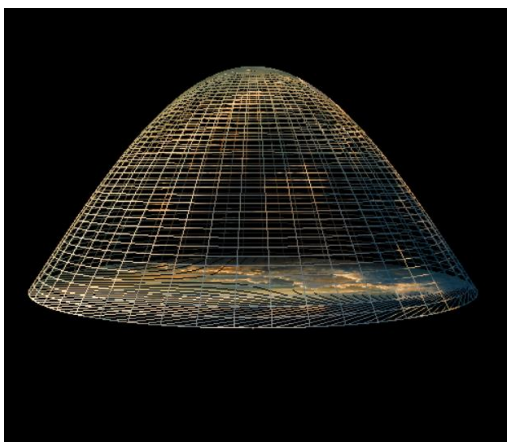
4. Now, alpha is angle between each slice. theta is angle of rotating r around Oy', which exactly the iteration over rings

5. UV mapping is as following order:

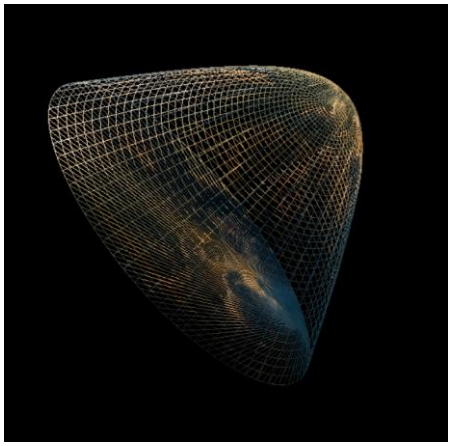


Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Paraboloid


Initialized (wireframe)


Initialized (fill)


Animated (wireframe)


Animated (fill)

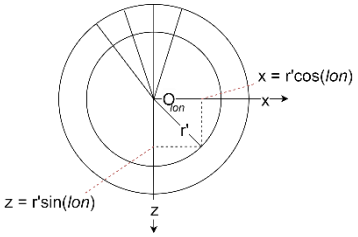
The circular cap is drawn as Circular Plane explained above.
The surround surface is calculated as followed:

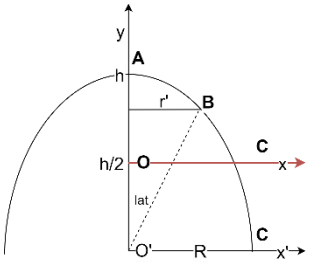
Code:

```
for (GLfloat lat = 0; lat <= PI / 2; lat += PI / 2 / total) {
glBegin(GL_QUAD_STRIP);
  for (GLfloat lon = 0; lon <= 2 * PI + 2 * PI / total;
                                    lon += 2 * PI / total) {
    r = lat / (PI / 2) * R;
    Z = r * sin(lon);
    X = r * cos(lon);
    Y = h * cos(lat) - h / 2;
    u = midTex + X / R / 2;
    v = midTex + Z / R / 2;
    glTexCoord2f(u, v);
    glVertex3f(X, Y, Z);

    r = (lat + PI / 2 / total) / (PI / 2) * R;
    Z = r * sin(lon);
    X = r * cos(lon);
    Y = h * cos(lat + PI / 2 / total) - h / 2;
    u = midTex + X / R / 2;
    v = midTex + Z / R / 2;
    glTexCoord2f(u, v);
    glVertex3f(X, Y, Z);
  }
  glEnd();
}
```

Graphical interpretation:

1. The initial position shows that the cone peak is facing upwards, the shape stands on xOz plane. The longitude goes full circle (0 to 2PI). Note that r' is r in the code.



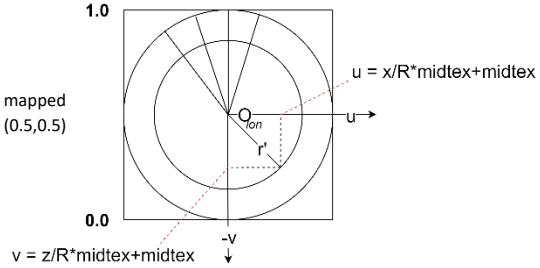


2. Latitude goes a quarter of a circle (PI / 2)

3. r'/R ≈ arcAB / arcAC ≈ lat/(PI/2)

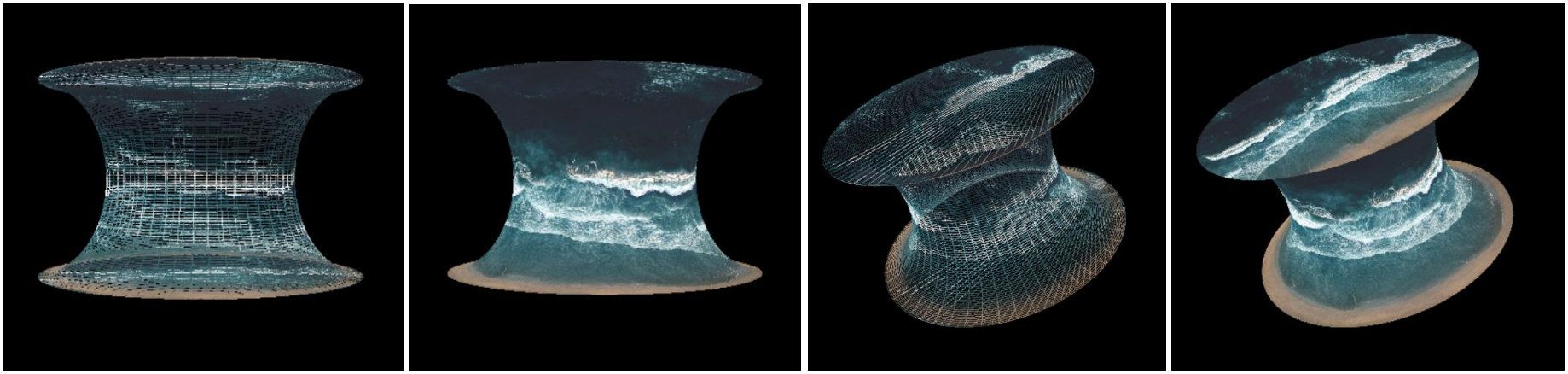
4. y' = O'B*cos(lat) = h*cos(lat) so y = y'-0.5h to translate shape's center to O(0,0).

5. The shape's peak is mapped with the center point (0.5,0.5) of the texture.



Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Hyperboloid



Initialized (wireframe) | Initialized (fill) | Animated (wireframe) | Animated (fill)

The two circular caps are drawn as Circular Plane explained above.
The surround surface is calculated as followed:

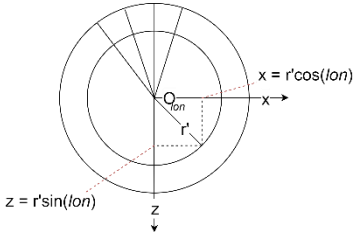
Code:

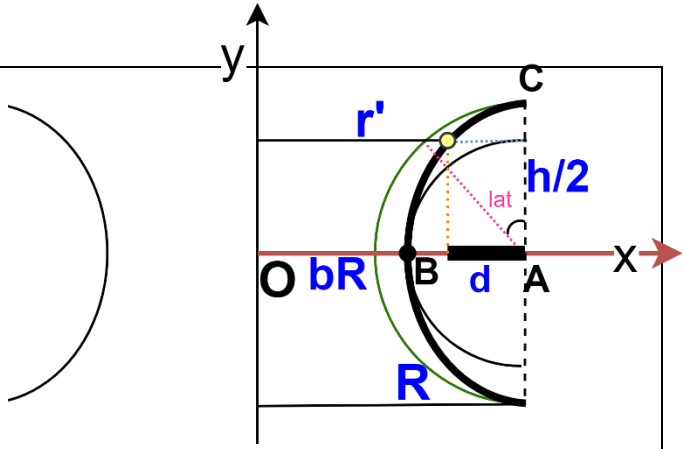
```
for (GLfloat lat = 0; lat <= PI; lat += PI / 2 / total) {
  glBegin(GL_QUAD_STRIP);
  for (GLfloat lon = 0; lon <= 2 * PI + 2 * PI / total;
                        lon += 2 * PI / total) {
    r = R - (R - bR) * sin(lat);
    Z = r * sin(lon);
    X = r * cos(lon);
    Y = h / 2 * cos(lat);
    u = lon / (2 * PI);
    v = lat / PI;
    glTexCoord2f(u, v);
    glVertex3f(X, Y, Z);

    r = R - (R - bR) * sin(lat + PI / 2 / total);
    Z = r * sin(lon);
    X = r * cos(lon);
    Y = h / 2 * cos(lat + PI / 2 / total);
    u = lon / (2 * PI);
    v = (lat + PI / 2 / total) / PI;
    glTexCoord2f(u, v);
    glVertex3f(X, Y, Z);
  }
  glEnd();
}
```

Graphical interpretation:

Considering the yellow point on the right-side figure

1. The initial position shows that the the shape's cap is facing upwards, the shape stands on xOz plane. The longitude goes full circle (0 to 2PI). Here, r' is r in the code.





2. Latitude goes a half a circle (PI),

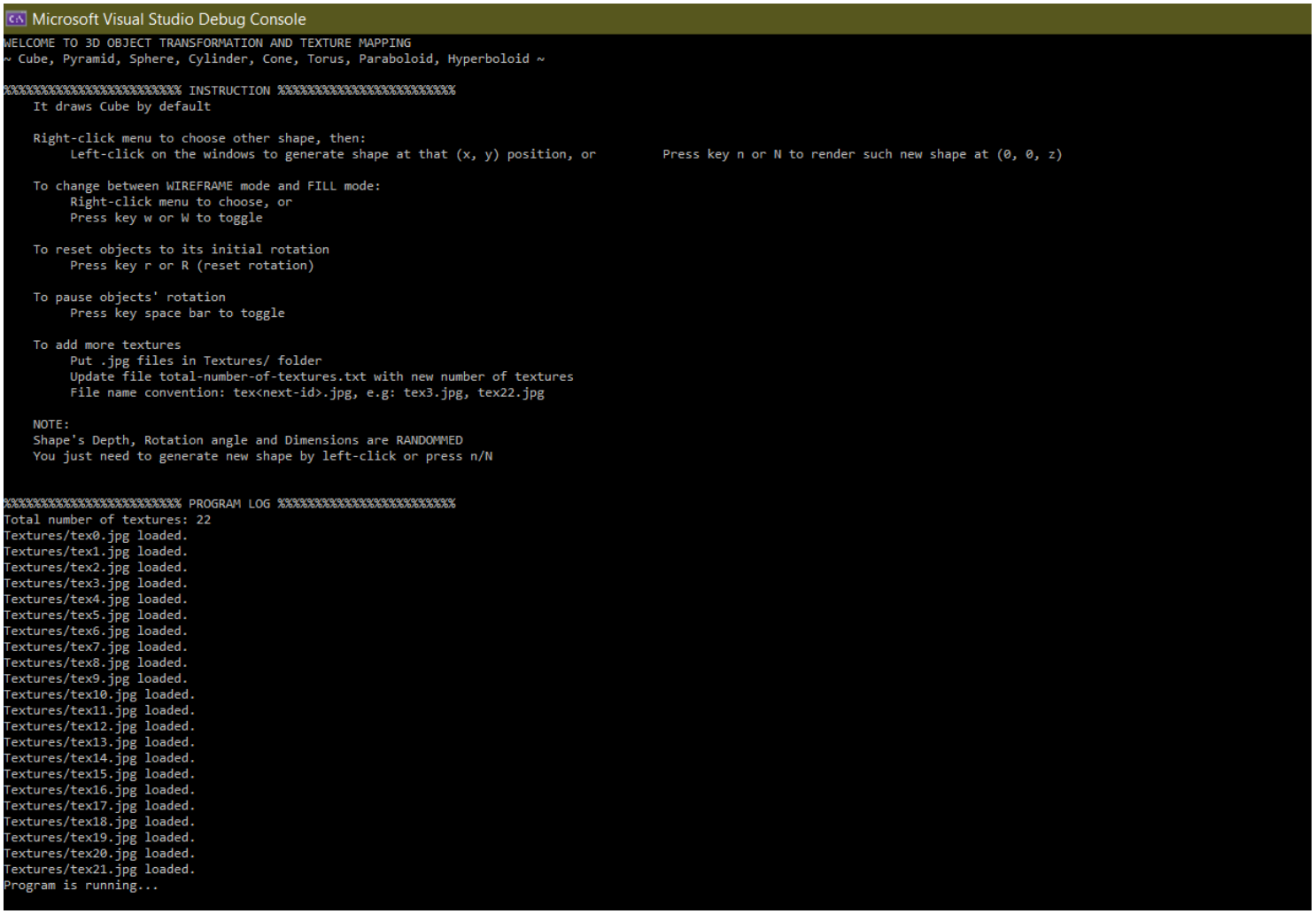
3. r' = R – d
d = **AB** * sin(lat) = (R - bR) * sin(lat)

4. **AC = h/2,** the green path denote the circle drawn by radius = h/2
so **y = AC * cos (lat) = h / 2 * cos (lat)**

5. Texture is treated the same manner as Torus's UV mapping

Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

Sample program console:



```
Cube:
     Rotation per time unit: 2.8rad
     Depth: -8
     Texture: tex12.jpg

Circular Plane:
     Rotation per time unit: 3.2rad
     Depth: -6
     Radius: 1.8
     Number of segments: 120
     Texture: tex7.jpg

Pyramid:
     Rotation per time unit: 2.8rad
     Depth: -9
     Texture: tex13.jpg

Sphere:
     Rotation per time unit: 1rad
     Depth: -6
     Radius: 0.6
     Number of segments: 60
     Texture: tex5.jpg

Cylinder:
     Rotation per time unit: 3.8rad
     Depth: -14
     Radius: 0.2
     Height: 1.6
     Number of segments: 50
     Texture: tex20.jpg

Cylinder:
     Rotation per time unit: 2.6rad
     Depth: -15
     Radius: 3.2
     Height: 0.2
     Number of segments: 50
     Texture: tex9.jpg

Cone:
     Rotation per time unit: 3.4rad
     Depth: -11
     Radius: 3.6
     Height: 1.4
     Number of segments: 50
     Texture: tex0.jpg

Torus:
     Rotation per time unit: 2rad
     Depth: -9
     Big Radius: 4
     Small Radius: 3
     Number of segments: 50
     Texture: tex6.jpg

Paraboloid:
     Rotation per time unit: 1.8rad
     Depth: -7
     Foot Radius: 3.4
     Height: 2.8
     Number of segments: 50
     Texture: tex16.jpg
```

Student ID: 1751036
Student Name: Tran Thi Anh Thu
CS411 – Lab04

```
    Hyperboloid:
         Rotation per time unit: 2.4rad
         Depth: -9
         Big Radius: 3.6
         Small Radius: 0.6
         Height: 1
         Number of segments: 50
         Texture: tex17.jpg

Program is terminated, release memory
Deleted shape
Deleted shape
Deleted shape
Deleted shape
Deleted shape
Deleted shape
Deleted shape
Deleted shape
Deleted shape
Deleted shape
Deleted shape

E:\school\1751036\subjects\CS411 - Computer Graphics\hw\1751036-lab04\Source\Debug\1751036-lab04.exe (process 23088) exited with code 0.
Press any key to close this window . . .
```

The end.