CG A2 report

2019311779최재경

1. Creation of sphere

1.1. 2D circle.h to 3D sphere.h

- Change variable and inline member function names.
  - `circle_t` to `sphere_t`, create_circles to create_spheres)
- Change `center` type to `vec3` and in `create_spheres()`, create an unit sphere with center at 3D Cartesian coordinate origin O = (0,0,0) and radius with 1.
- Change matrix value.
  - To convert scale matrix of 2D circle to 3D sphere, change (3,3) value 1 to `radius`.
  - To convert translate matrix of 2D circle to 3D sphere, change (4,3) value 0 to `center.z`

  1.2. Define array of vertices at vertex buffer : `create_sphere_vertices()`
  Vertex is consist of position, normal vector and texture coordinate. Duplicated for loop statement generate calculated values by following formulas and `push_back` to vertex array buffer `v`. It loops just one more time so overlaps starting vertex and ending vertex because this is vertex creation.
- Cartesian position coordinate

$$P(x,y,z) = O + (rsin\theta cos\varphi, rsin\theta sin\varphi, rcos\theta)$$

  where $\varphi \in [0, 2\pi]$, $\theta \in [0, \pi]\, and\, r = 1$
- Normal vector

$$N(x,y,z) = (sin\theta cos\varphi, sin\theta sin\varphi, cos\theta)$$

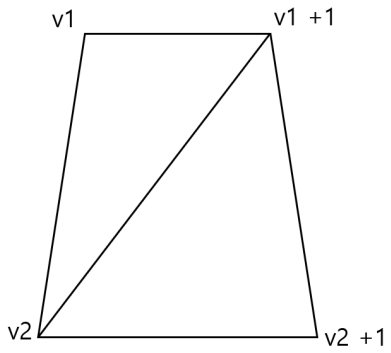
  where $\varphi \in [0, 2\pi]\, and\, \theta \in [0, \pi]$
- Texture coordinates

$$T(x,y) = (\varphi/2\pi, 1 - \theta/\pi)$$

  where$T_x \in [0,1]\, and\, T_y \in [0,1]$

  1.3 Connect to store in index buffer with counter-clockwise order
  : `update_vertex_buffer()`
  When vertices are generated as above, it looks like
  globe with vertex
  위와 같이 vertex들을 생성하면, Like a globe, the number of vertexes located at the same latitude is 36 + 1, and the number of vertices located at the same longitude is 72 + 1, and it can be seen that it is composed of the same plane as quad, except for the

first and last lines, which are both poles. Quads belonging to this line are divided into two triangles by dividing them diagonally, and index buffer is formed by storing indications counterclockwise. (e.g. (v1, v2, v1 + 1) and (v1+1, v2, v2+1) respectivly.)



1.4. World positions to canonical view volume
Jobs done as assignment pdf introduces.
Replace `aspect_matrix` to `view_projection_matrix` at `update()` function and vertex shader `sphere.vert`.

1.5. user_init()
Function name modified.

1.6 render()
With `c.update(t)` update position of vertices and Draw triangles with `glDrawElements()`. `num_indices` is total number of indices.
(e.g., 2 *num_indices = (LONG_NUM + 1)* (LATI_NUM – 1) *3 + (LONG_NUM + 1) 2 3 = 2* (LONG_NUM + 1) *LATI_NUM* 3)

2. Sphere texture toggle implementation

- Declare and initiate at `main.cpp` an global variable `soild_color_state= 0`
- At `keyboard` function, implement logic of 'D' key `GLFW_PRESS` keyboard event with by increment `+=1`.
- Implement `glGetUniformLocation` at `update()` to send updated `soild_color_state` to fragment shader as uniform variable.
- At main function of fragment shader `sphere.frag`, set `fragColor = vec4(tc.xy,0,1);`, other two `fragColor` values and uniform variable so that it can be toggled texture through if statement by update of uniform variable `soild_color_state`
- This way, sphere is texture in terms of texture coordinates and can be toggled.

3. Rotation of sphere

- At `keyboard` function, implement logic of start and stop rotate by 'R' key `GLFW_PRESS` keyboard event with toggling `b_rotate` boolean value.
- When controlling frame time, use same logic as used in assignment 1. At `update()`, get current frame time by `glfwGetTime()` and calculate `elapsed_time` by subtracting with

global variable `prev_time` (holds previous updated frame time). Update global variable `t` and prev_time by each `update()`.

- Then `t` is send as parameter at `c.update(t)`, which updates model matrix of instance. In `sphere_t::update()`, `t` is used as `theta` to rotate the sphere.
- Since our program's *eye*(carmera) and *at* are (1,0,0) and (0,0,0) respectively and after testing x y and z rotation direction compared to sample program, the sphere is rotating about z axis. So, rotation matrix is modified as below.

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Mandatory Requirements

- Vertex buffering is deleted.
- Backface culling, W key toggle to wireframe, and window resizing is already implemented due to reuse of g1-02-circle sample
- window size modified.
  - `window_size = ivec(1280, 720);`
- Initialize the vertex buffer(vertexarray object) only once
  - At create_sphere_vertices(), with unit radius.