Computer Graphics
- Primitives

Sung Soo Hwang

(Polygon) Mesh

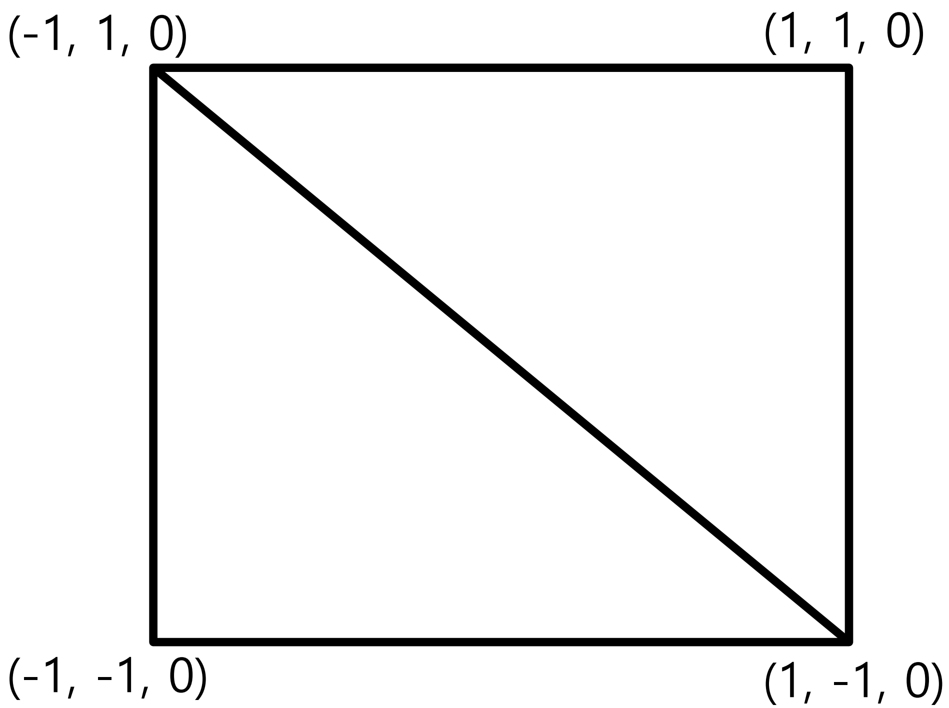
- A mesh is a set of topologically related planar polygons (often triangles)
- represented as follows:

  vertices: a list of 3-tuples

  faces: a set of index lists to the vertices

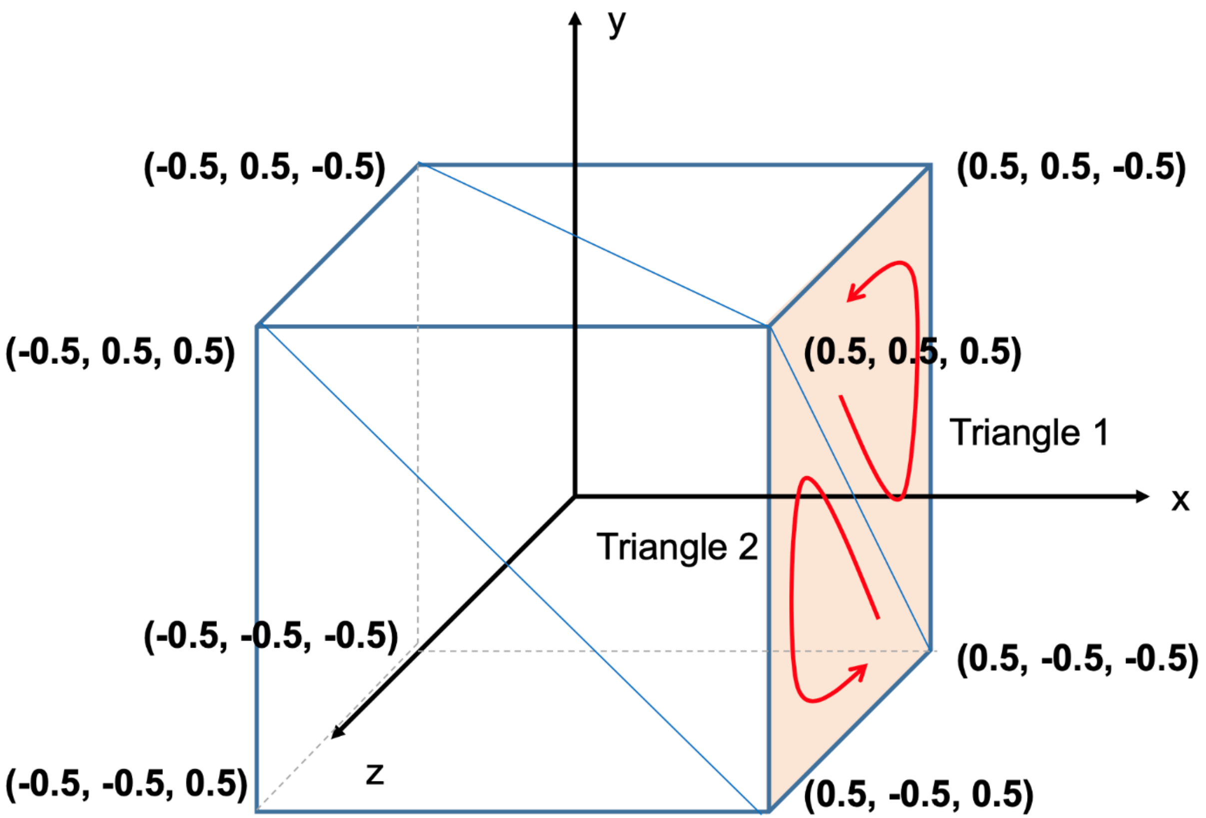
Counter-clockwise
Edge implied

$\boldsymbol{v_1}$	-1, -1, 0
$\boldsymbol{v_2}$	1, -1, 0
$\boldsymbol{v_3}$	1, 1, 0
$\boldsymbol{v_4}$	-1, 1, 0
$\boldsymbol{f_1}$	1, 2, 4
$\boldsymbol{f_2}$	-1, 1, 0



Cube Primitive

- Box Geometry Construction
  - A unit cube



One side at x = 0.5:

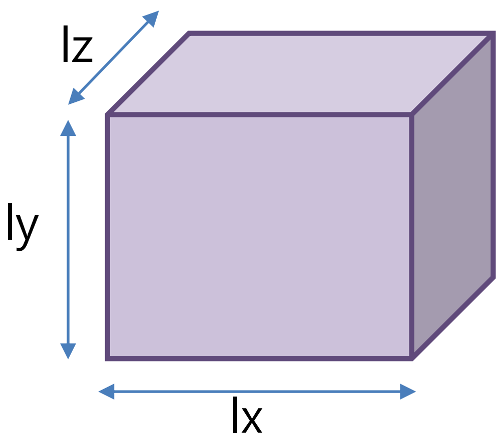
Triangle 1:
(0.5, 0.5, 0.5),
(0.5,-0.5,-0.5),
(0.5, 0.5,-0.5)

Triangle 2:
(0.5,-0.5,-0.5),
(0.5, 0.5, 0.5),
(0.5,-0.5, 0.5)

Cube Primitive

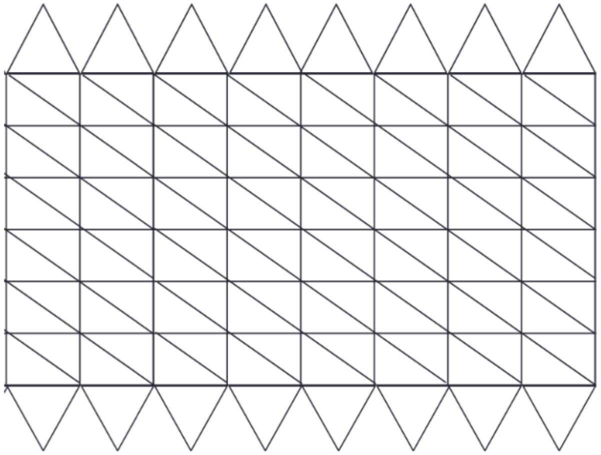
- Computing the vertices of a box

```
void get_box_3d(std::vector<GLfloat>& p, GLfloat lx, GLfloat ly, GLfloat lz)
{
    static const GLfloat box_vertices[] = {
         0.5f, 0.5f,-0.5f,  -0.5f,-0.5f,-0.5f,  -0.5f, 0.5f,-0.5f,  // side at z = -0.5
         0.5f, 0.5f,-0.5f,   0.5f,-0.5f,-0.5f,  -0.5f,-0.5f,-0.5f,
        -0.5f,-0.5f,-0.5f,  -0.5f,-0.5f, 0.5f,  -0.5f, 0.5f, 0.5f,  // side at x = -0.5
        -0.5f,-0.5f,-0.5f,  -0.5f, 0.5f, 0.5f,  -0.5f, 0.5f,-0.5f,
         0.5f,-0.5f, 0.5f,  -0.5f,-0.5f,-0.5f,   0.5f,-0.5f,-0.5f,  // side at y = -0.5
         0.5f,-0.5f, 0.5f,  -0.5f,-0.5f, 0.5f,  -0.5f,-0.5f,-0.5f,
        -0.5f, 0.5f, 0.5f,  -0.5f,-0.5f, 0.5f,   0.5f,-0.5f, 0.5f,  // side at z = 0.5
         0.5f, 0.5f, 0.5f,  -0.5f, 0.5f, 0.5f,   0.5f,-0.5f, 0.5f,
         0.5f, 0.5f, 0.5f,   0.5f,-0.5f,-0.5f,   0.5f, 0.5f,-0.5f,  // side at x = 0.5
         0.5f,-0.5f,-0.5f,   0.5f, 0.5f, 0.5f,   0.5f,-0.5f, 0.5f,
         0.5f, 0.5f, 0.5f,   0.5f, 0.5f,-0.5f,  -0.5f, 0.5f,-0.5f,  // side at y = 0.5
         0.5f, 0.5f, 0.5f,  -0.5f, 0.5f,-0.5f,  -0.5f, 0.5f, 0.5f
    };

    p.resize(sizeof(box_vertices) / sizeof(GLfloat));
    memcpy(p.data(), box_vertices, sizeof(box_vertices));
    size_t n = p.size()/3;
    for (int i = 0; i < n; ++i) {
        p[3 * i + 0] *= lx;
        p[3 * i + 1] *= ly;
        p[3 * i + 2] *= lz;
    }
}
```

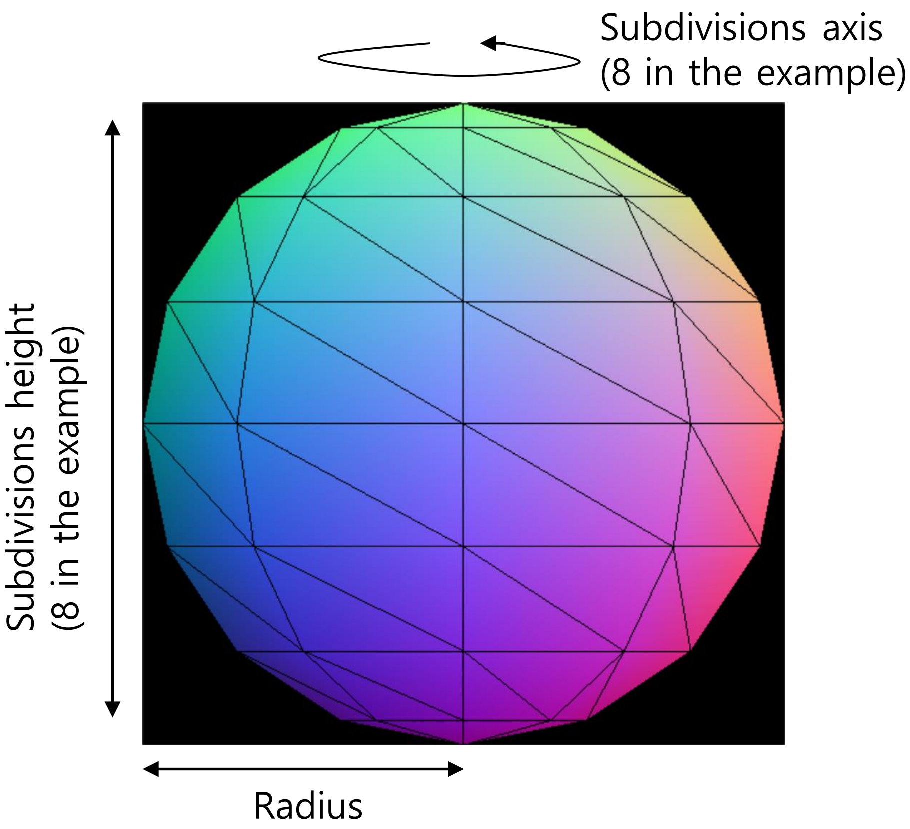


Sphere Primitive

- Required Parameters
  - Radius (r): the radius of a sphere
  - Subdivisions height (subh): the number of subdivisions along the direction of height
  - Subdivisions axis (suba): the number of subdivisions around the axis of rotation

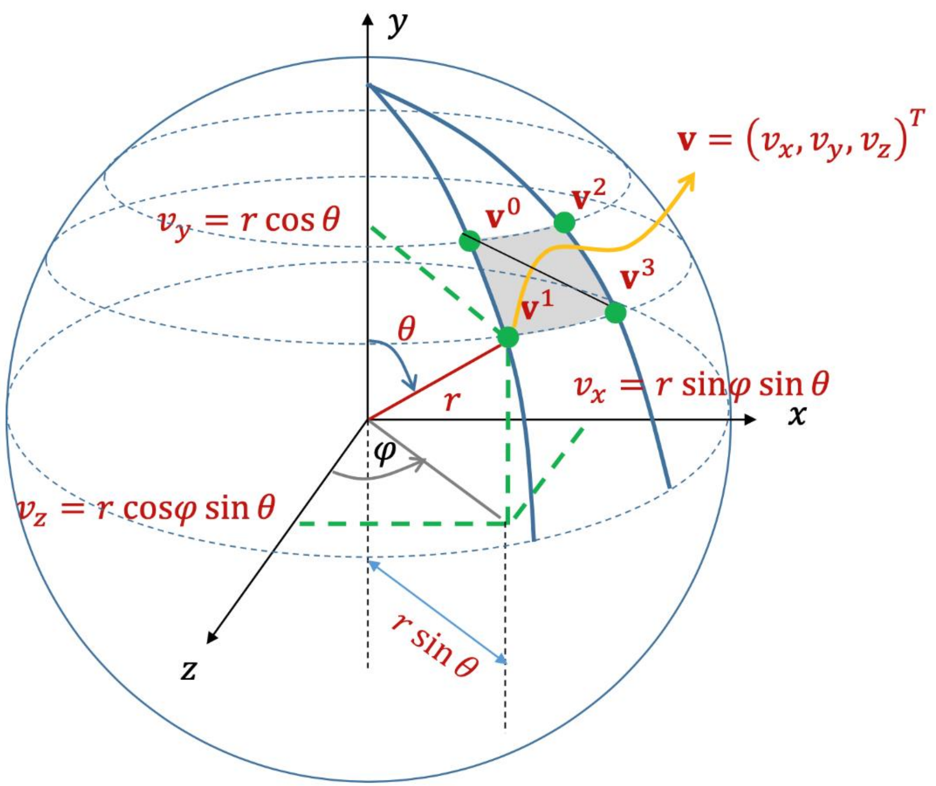


Topology of sphere

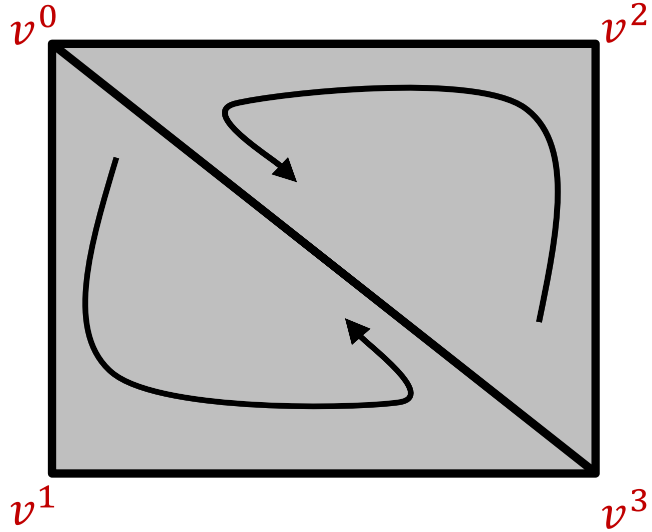


Sphere Primitive

- Sampling vertices of a sphere



$$v_x = r\ sin\varphi\ sin\theta$$
$$v_y = r\ cos\theta$$
$$v_z = r\ cos\varphi\ sin\theta$$



First triangle: $v^0 - v^1 - v^3$
Second triangle: $v^3 - v^2 - v^0$

Sphere Primitive

- Function header and macros to generate a sphere

```
#define FPUSH_VTX3(p,vx,vy,vz)
do {
        p.push_back(vx);
        p.push_back(vy);
        p.push_back(vz);
} while(0)

#define FSET_VTX3(vx,vy,vz, valx,valy,valz)
do {
        vx=(float)(valx);
        vy=(float)(valy);
        vz=(float)(valz);
} while(0)

void get_sphere_3d(
        GLvec& p,
        float r,
        int subh,
        int suba);
```

- `GLvec& p` → Vector to store 3d vertex positions of a sphere
- `float r` → Radius of a sphere
- `int subh` → Divisions along the directions of height
- `int suba` → Divisions around the axis of rotation

Sphere Primitive

- Function header and macros to generate a sphere

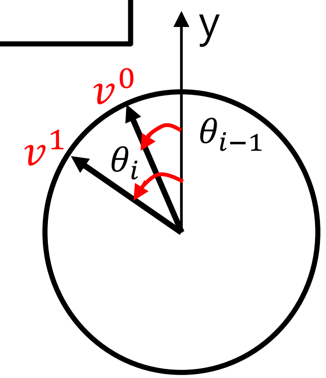
```
void get_sphere_3d(
        GLvec& p,
        float r,
        int subh,
        int suba);
{
        for (int i = 1; i <= subh; ++i) {
                double theta0 = M_PI * (i-1) / subh;
                double theta1 = M_PI * i / subh;

                double y0 = r * cos(theta0);
                double rst0 = r * sin(theta0);
                double y1 = r * cos(theta1);
                double rst1 = r * sin(theta1);

          for (int j = 1; j <= suba; ++j) {
                  double phi0 = 2 * M_PI * (j-1) / suba;
                  double phi1 = 2 * M_PI * j / suba;

                  double cp0 = cos(phi0);
                  double sp0 = sin(phi0);
                  double cp1 = cos(phi1);
                  double sp1 = sin(phi1);
```

(continued in the next slide)

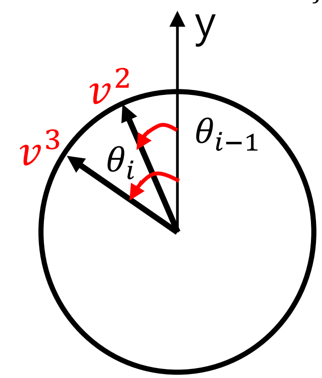
$v_x = sin\varphi \cdot rst$
$v_y = r\ cos\theta$
$v_z = cos\varphi \cdot rst$
Here, $rst = r\ sin\theta$

$v_x = sin\varphi \cdot rst$
$v_y = r\ cos\theta$
$v_z = cos\varphi \cdot rst$

$v_x = sin\varphi \cdot rst$
$v_y = r\ cos\theta$
$v_z = cos\varphi \cdot rst$



Circular Section for φ_{j-1}



Circular Section for φ_j

Sphere Primitive

- Function header and macros to generate a sphere

```
float vx0, vy0, vz0, vx1, vy1, vz1;
float vx2, vy2, vz2, vx3, vy3, vz3;

FSET_VTX3(vx0, vy0, vz0, sp0*rst0, y0, cp0*rst0);
FSET_VTX3(vx1, vy1, vz1, sp0*rst1, y1, cp0*rst1);
FSET_VTX3(vx2, vy2, vz2, sp1*rst0, y0, cp1*rst0);
FSET_VTX3(vx3, vy3, vz3, sp1*rst1, y1, cp1*rst1);

if (i < subh) {
            // first triangle (v0 – v1 – v3)
            FPUSH_VTX3(p, vx0, vy0, vz0);
            FPUSH_VTX3(p, vx1, vy1, vz1);
            FPUSH_VTX3(p, vx3, vy3, vz3);
}

if (1 < i) {
            // second triangle (v3 – v2 – v0)
            FPUSH_VTX3(p, vx3, vy3, vz3);
            FPUSH_VTX3(p, vx2, vy2, vz2);
            FPUSH_VTX3(p, vx0, vy0, vz0);
}
      } // for (int j = 1; j <= suba; ++j)
  } // for (int i = 1; i <= subh; ++i)
}
```

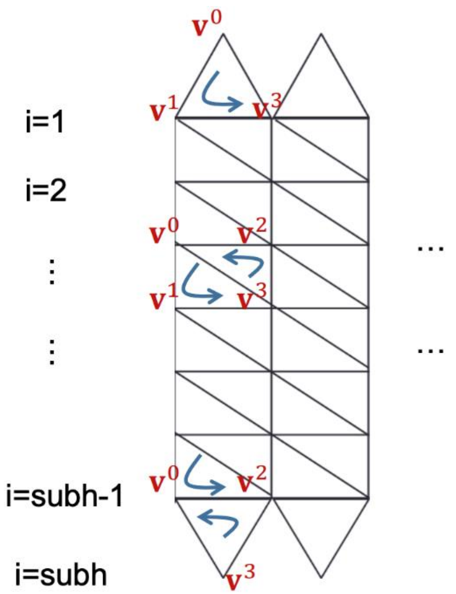
$v_x = sin\varphi \cdot rst$

$v_y = r\ cos\theta$

$v_z = cos\varphi \cdot rst$

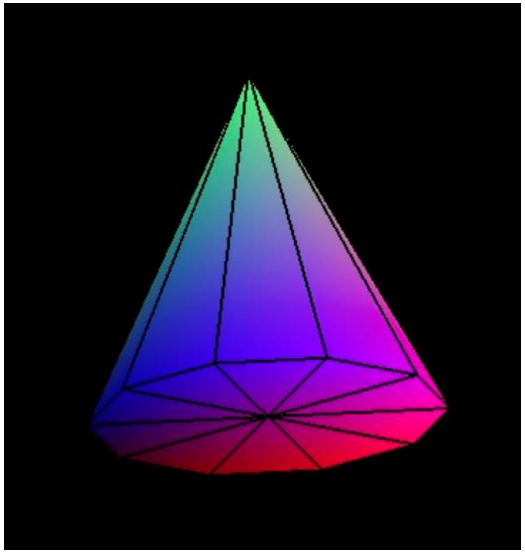
Here, $rst = r\ sin\theta$

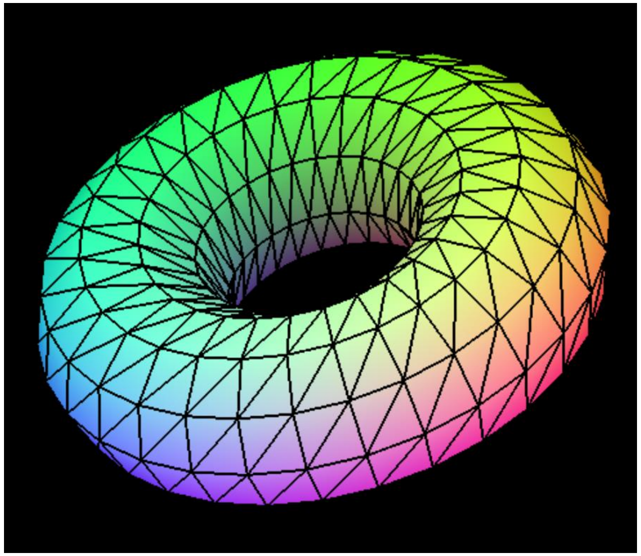
Compute the coordinates of v^0, v^1, v^2, v^3

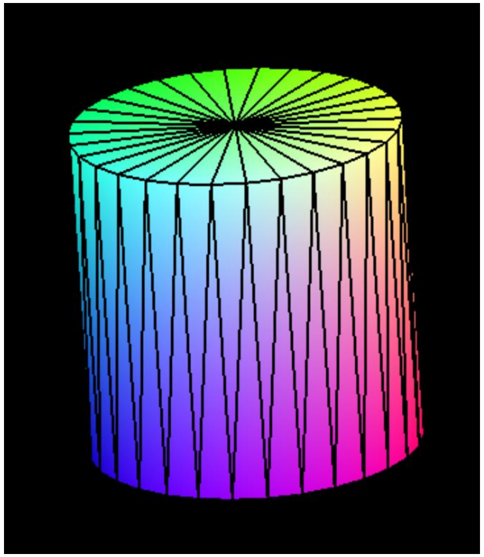


More Geometric Primitives

- Cone Primitives
- Cylinder Primitives
- Torus Primitives

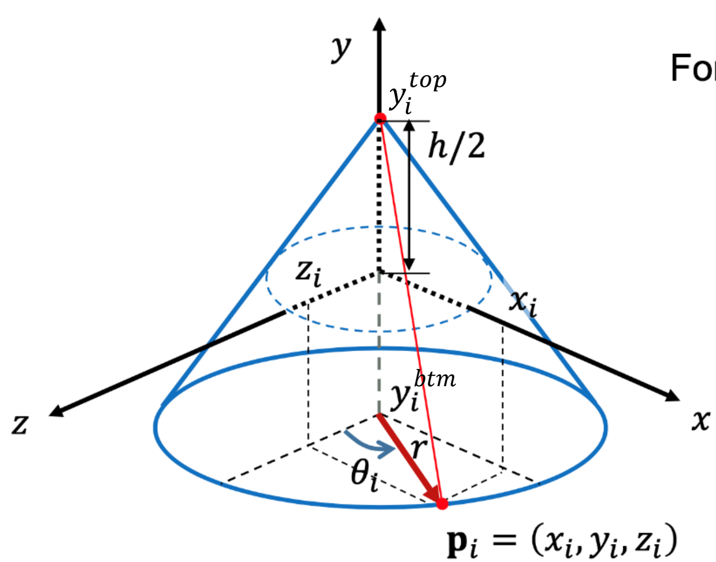






Cone Primitives

- Required Parameters
  - Height (h): the height of a cone
  - Radius (r): the radius of the circle on the bottom
  - Subdivisions (n): the number of sides



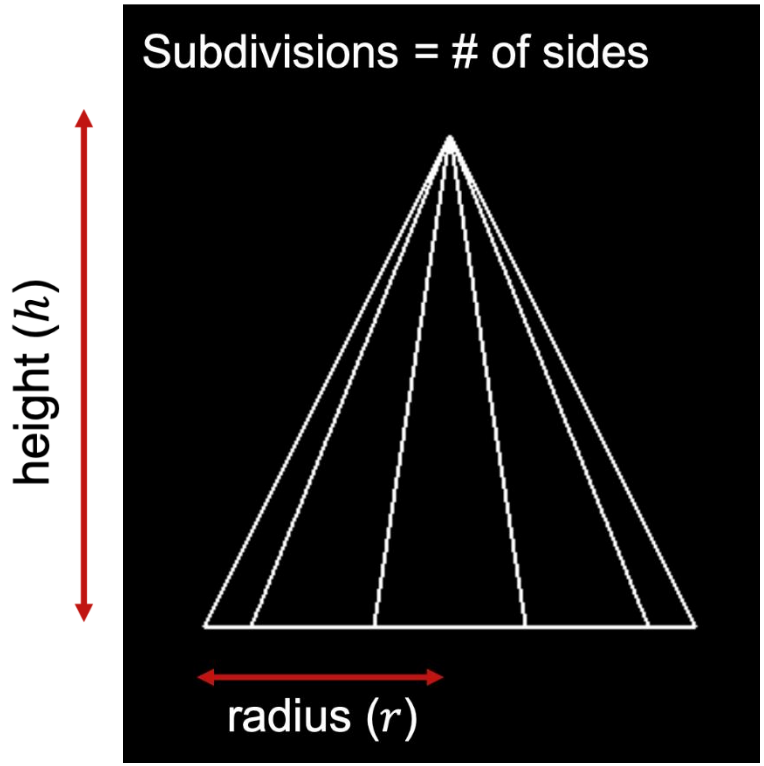
For $i = 0,1,\dots,n$

$$\theta_i = 2\pi \cdot i/n$$

$$x_i = r \sin\theta_i$$

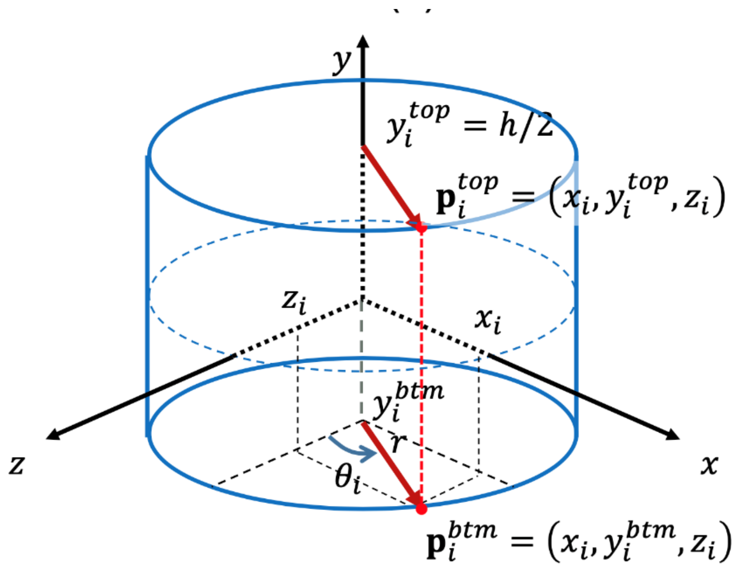
$$y_i^{btm} = -h/2$$

$$z_i = r \cos\theta_i$$



Cylinder Primitives

- Required Parameters
  - Height (h): the height of a cylinder
  - Radius (r): the radius of the circles on the top and bottom
  - Subdivisions (n): the number of sides



For $i = 0,1,\ldots,n$

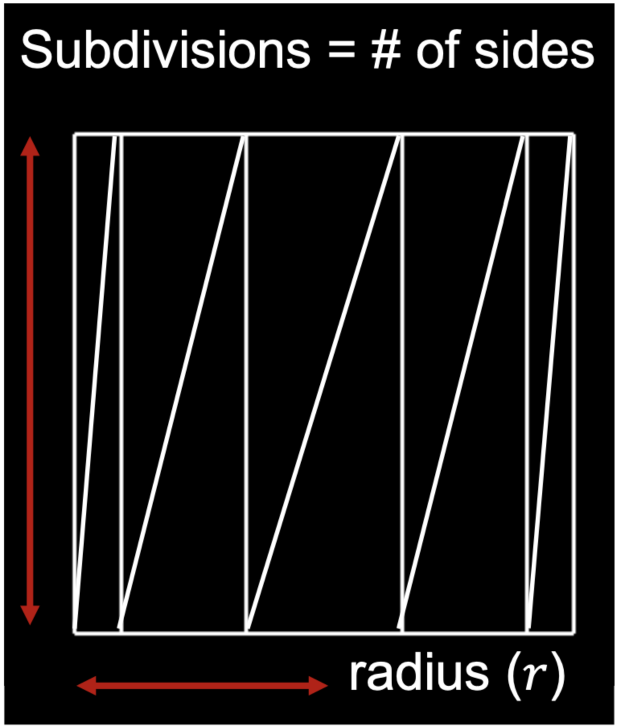
$$\theta_i = 2\pi \cdot i/n$$

$$x_i = r \sin\theta_i$$

$$y_i^{top} = h/2$$

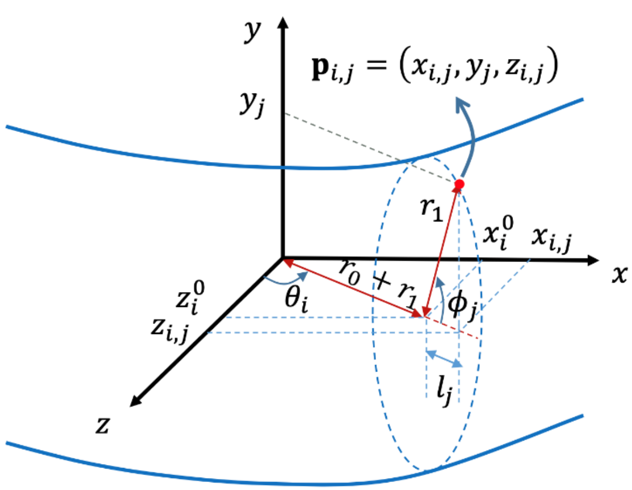
$$y_i^{btm} = -h/2$$

$$z_i = r\cos\theta_i$$



Torus Primitives

- Required Parameters
  - Torus radius (r_0): the inner radius of the torus
  - Selection radius (r_1): the radius of each radial section
  - Subdivision Axis (n_a): the number of divisions around the axis of rotation
  - Subdivision Height (n_h): the number of divisions of



$$x_{i,j} = x_i^0 + \Delta x_{i,j} \qquad \theta_i = 2\pi \cdot i/n_0$$
$$y_j = r_1 \sin\phi_j \qquad \phi_j = 2\pi \cdot j/n_1$$
$$z_{i,j} = z_i^0 + \Delta z_{i,j}$$

$$\left[\begin{array}{l} x_i^0 = (r_0 + r_1)\sin\theta_i \\ z_i^0 = (r_0 + r_1)\cos\theta_i \\ \\ \Delta x_{i,j} = l_j \sin\theta_i \\ \Delta z_{i,j} = l_j \cos\theta_i \\ \\ l_j = r_1 \cos\phi_j \end{array}\right.$$