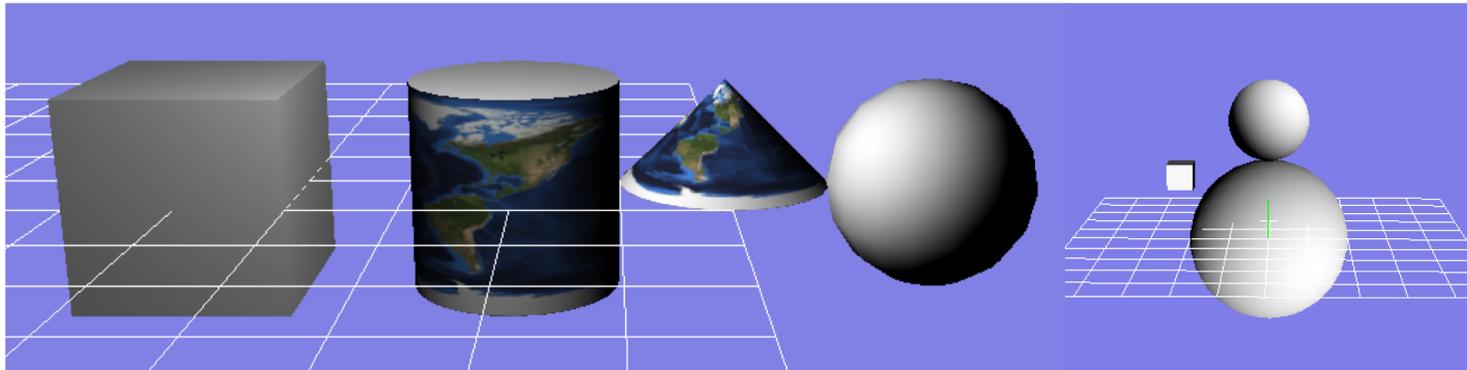

TD1 - formes de base et objet complexe

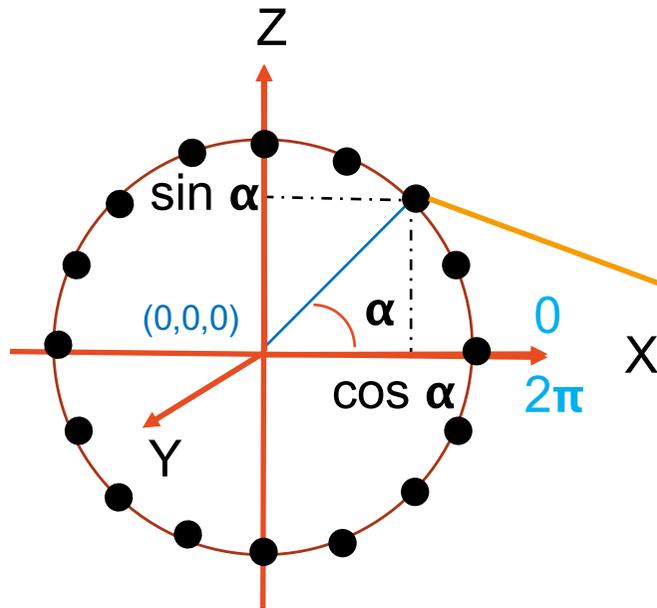


Eric Galin (semestre printemps)
Florence Zara (semestre automne)
LIRIS-ORIGAMI, Université Lyon 1

Modélisation d'un disque en 3D

Disque centré en
(0,0,0) et de rayon 1

Disque dans le plan $y=0$



Définition des sommets

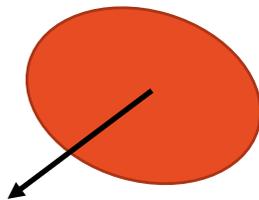
Sommets du cercle de coordonnées

$$x = \cos \alpha$$

$$y = 0$$

$$z = \sin \alpha$$

Variation de l'angle de 0 à 2π pour
définir les points du cercle



Normale : vecteur $(0, 1, 0)$

Définition de la normale à la surface

Modélisation d'un disque en 3D

Disque centré en
(0,0,0) et de rayon 1

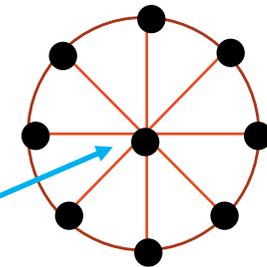
Disque dans le plan $y=0$

```
void ViewerEtudiant::init_disque()
{
    // Variation de l'angle de 0 à 2π
    const int div = 25;
    float alpha;
    float step = 2.0 * M_PI / (div);

    // Choix primitive OpenGL
    m_disque = Mesh( GL_TRIANGLE_FAN );

    m_disque.normal( Vector(0,1,0) ); // Normale à la surface
    m_disque.vertex( Point(0,0,0) ); // Point du centre du disque

    // Variation de l'angle de 0 à 2π
    for (int i=0; i<=div; ++i)
    {
        alpha = i * step;
        m_disque.normal( Vector(0,1,0) );
        m_disque.vertex( Point(cos(alpha), 0, sin(alpha)) );
    }
}
```



Modélisation d'un cube en 3D

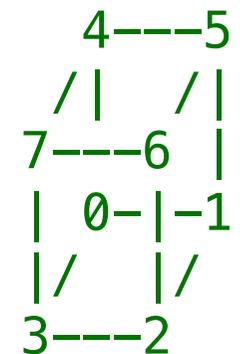
Cube centré en
(0,0,0) et de côté 2

```
void Viewer::init_cube()
{
    // 8 sommets
    static float pt[8][3] = { {-1,-1,-1}, {1,-1,-1}, {1,-1,1}, {-1,-1,1}, {-1,1,-1}, {1,1,-1}, {1,1,1}, {-1,1,1} };

    // 6 faces
    static int f[6][4] = { {0,1,2,3}, {5,4,7,6}, {2,1,5,6}, {0,3,7,4}, {3,2,6,7}, {1,0,4,5} };

    // les 6 normales pour chacune des faces
    static float n[6][3] = { {0,-1,0}, {0,1,0}, {1,0,0}, {-1,0,0}, {0,0,1}, {0,0,-1} };

    // Choix des primitives OpenGL employées
    m_cube = Mesh(GL_TRIANGLE_STRIP);
```



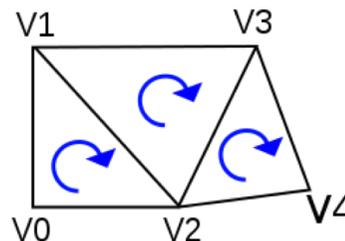
GL_TRIANGLE_STRIP

Note that only one additional vertex is needed to draw the second triangle. In OpenGL, the order in which the vertices are specified is important so that surface normals are consistent.

Quoting directly from the [OpenGL Programming Guide](#):

GL_TRIANGLE_STRIP

Draws a series of triangles (three-sided polygons) using vertices v_0, v_1, v_2 , then v_2, v_1, v_3 (note the order), then v_2, v_3, v_4 , and so on. The ordering is to ensure that the triangles are all drawn with the same orientation so that the strip can correctly form part of a surface.



From Wikipedia, the free encyclopedia

Modélisation d'un cube en 3D

1^{er} triangle : v0, v1, v2
2^e triangle : v2, v1, v3

```
// Parcours des 6 faces
for (int i=0; i<6; i++) // i = numéro de la face
{
    // La normale à la face
    m_cube.normal(n[i][0], n[i][1], n[i][2]);

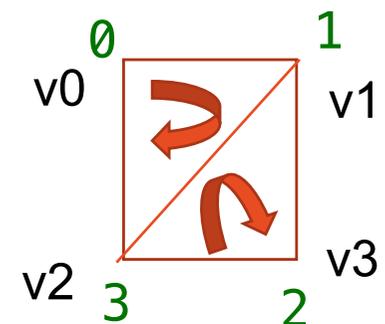
    // Les 4 sommets de la face
    m_cube.vertex( pt[ f[i][0] ][0], pt[ f[i][0] ][1], pt[ f[i][0] ][2] );
    m_cube.vertex( pt[ f[i][1] ][0], pt[ f[i][1] ][1], pt[ f[i][1] ][2] );
    m_cube.vertex( pt[ f[i][3] ][0], pt[ f[i][3] ][1], pt[ f[i][3] ][2] );
    m_cube.vertex( pt[ f[i][2] ][0], pt[ f[i][2] ][1], pt[ f[i][2] ][2] );

    m_cube.restart_strip(); // Demande un nouveau strip
}
} //void
```

Coordonnées en x, y et z de la normale à la face i

indice du sommet 0 de la face i

Coordonnées en x, y et z du sommet 0 de la face i

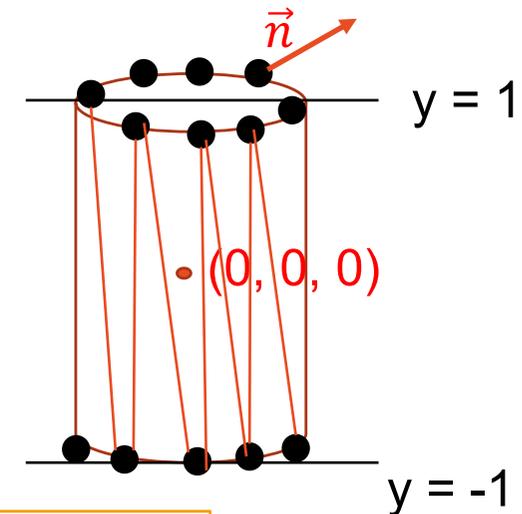
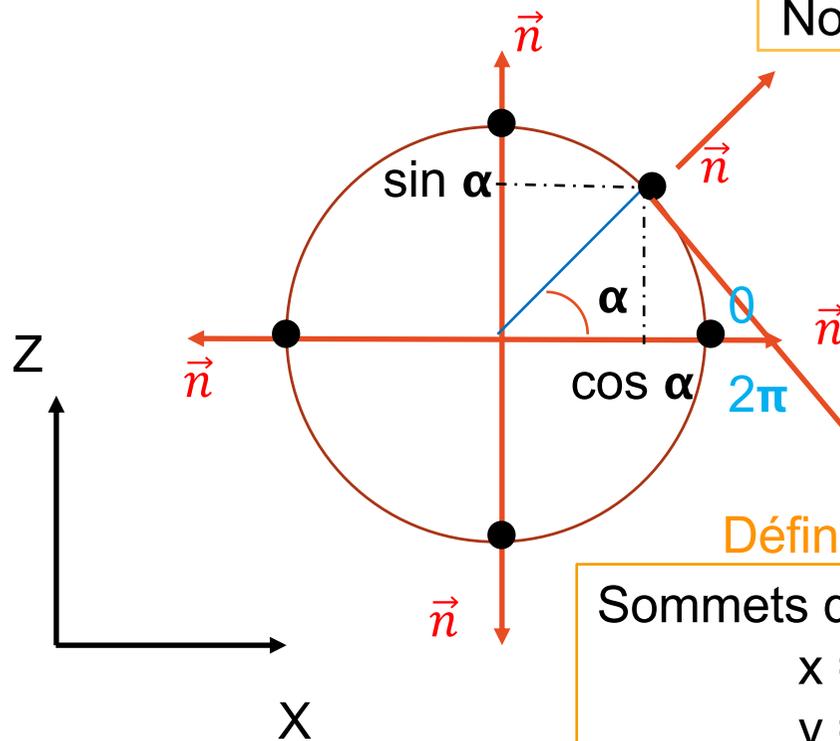


Modélisation d'un cylindre en 3D

Centre : (0, 0, 0)
Base de rayon 1
Hauteur : 2

Définition des normales

Normale = vecteur $(\cos \alpha, 0, \sin \alpha)$



Définition des sommets

Sommets du cercle de coordonnées

$$x = \cos \alpha,$$

$$y = 1 \text{ (haut) ou } y = -1 \text{ (bas)}$$

$$z = \sin \alpha$$

Variation de l'angle de 0 à 2π pour définir les points des cercles

Modélisation d'un cylindre en 3D

Centre : (0, 0, 0)
Base de rayon 1
Hauteur : 2

```
void ViewerEtudiant::init_cylinder()
{
    // Variation de l'angle de 0 à 2π
    int i;
    const int div = 25;
    float alpha;
    float step = 2.0 * M_PI / (div);

    // Choix primitive OpenGL
    m_cylindre = Mesh(GL_TRIANGLE_STRIP);
}
```

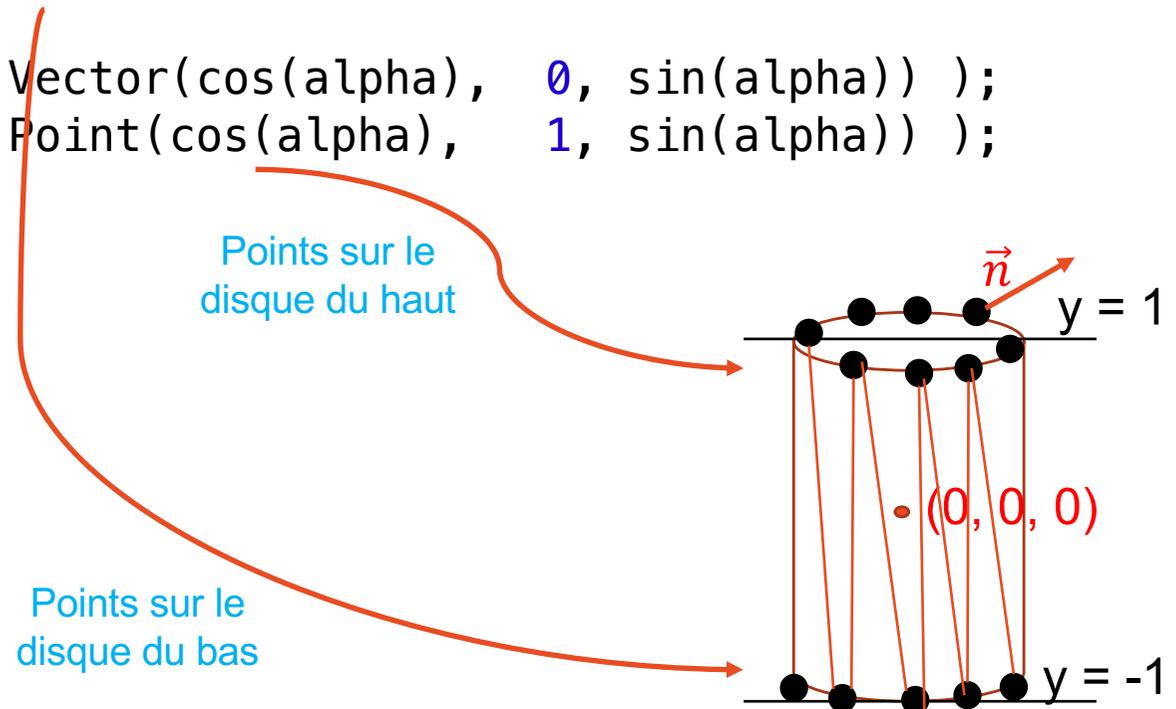
Modélisation d'un cylindre en 3D

Centre : (0, 0, 0)
Base de rayon 1
Hauteur : 2

```
for (int i=0; i<=div; ++i)
{
    // Variation de l'angle de 0 à 2π
    alpha = i * step;

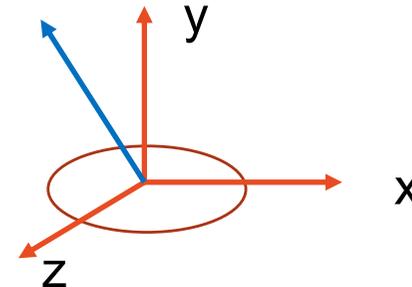
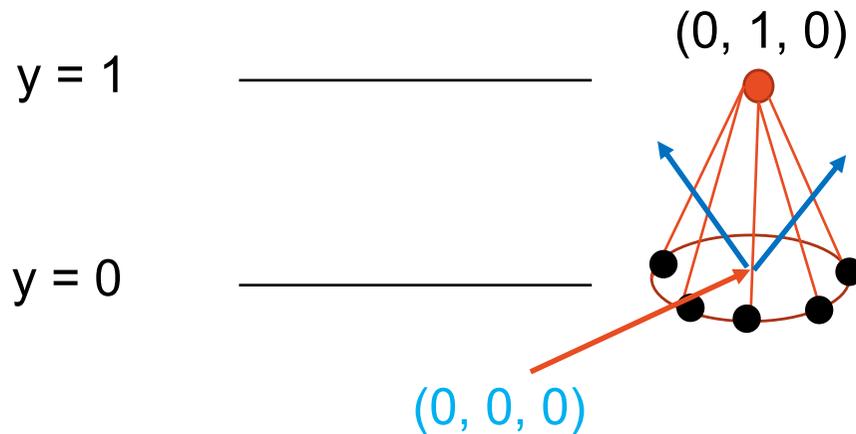
    m_cylindre.normal( Vector(cos(alpha), 0, sin(alpha)) );
    m_cylindre.vertex( Point(cos(alpha), -1, sin(alpha)) );

    m_cylindre.normal( Vector(cos(alpha), 0, sin(alpha)) );
    m_cylindre.vertex( Point(cos(alpha), 1, sin(alpha)) );
}
} //void
```



Modélisation d'un cône en 3D

Base centrée en $(0, 0, 0)$
Base de rayon 1
Hauteur : 1



Définition des sommets

Sommets du cercle de coordonnées

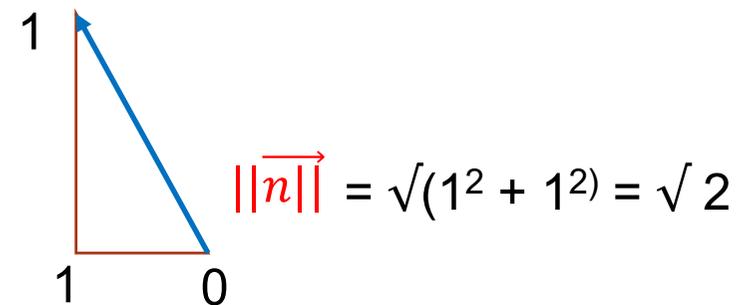
$$x = \cos \alpha,$$

$$y = 0$$

$$z = \sin \alpha$$

Sommet du haut

$$x = 0, y = 1, z = 0$$



Définition des normales

Normale = vecteur $(\cos \alpha / \sqrt{2}, 0, \sin \alpha / \sqrt{2})$

Modélisation d'un cône en 3D

```
void ViewerEtudiant::init_cone()
{
    // Variation de l'angle de 0 à 2π
    const int div = 25;
    float alpha;
    float step = 2.0 * M_PI / (div);

    // Choix de la primitive OpenGL
    m_cone = Mesh(GL_TRIANGLE_STRIP);
}
```

Modélisation d'un cône en 3D

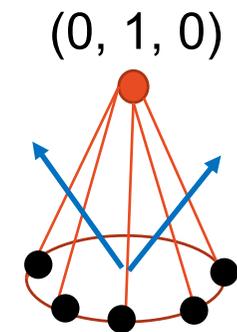
```
for (int i=0;i<=div;++i)
{
    alpha = i * step; // Angle varie de 0 à 2π

    m_cone.normal(
        Vector( cos(alpha)/sqrtf(2.f),
                1.f/sqrtf(2.f),
                sin(alpha)/sqrtf(2.f) ));

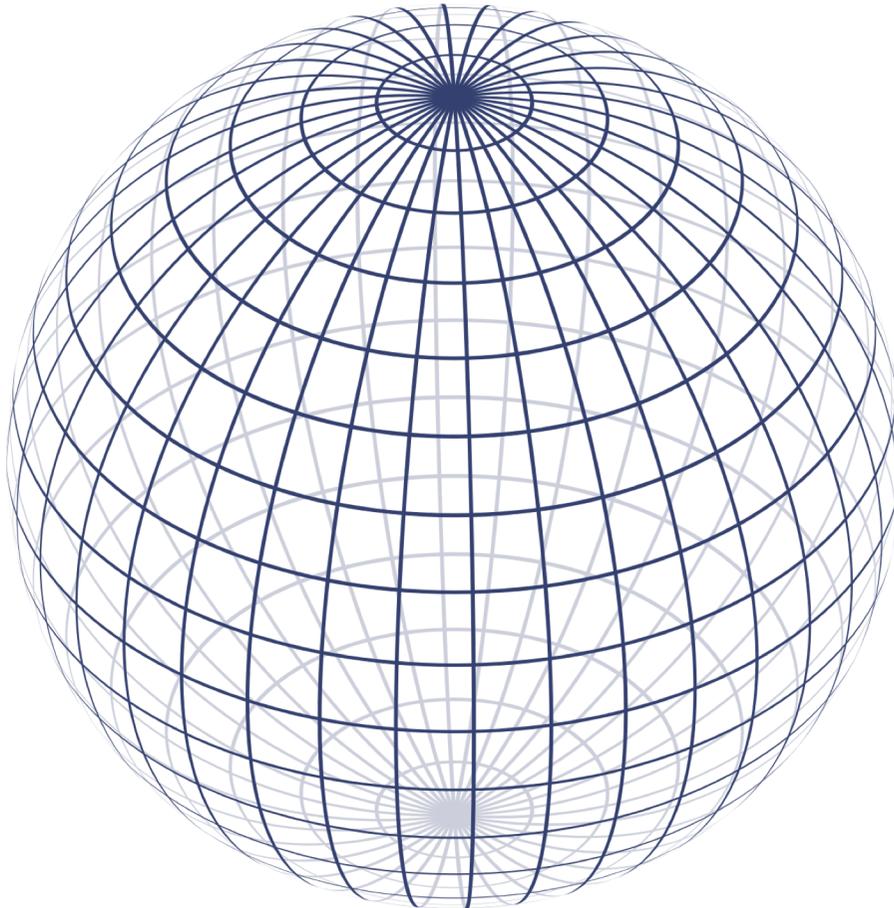
    m_cone.vertex( Point( cos(alpha), 0, sin(alpha) ));

    m_cone.normal(
        Vector( cos(alpha)/sqrtf(2.f),
                1.f/sqrtf(2.f),
                sin(alpha)/sqrtf(2.f) ));

    m_cone.vertex( Point(0, 1, 0) );
}
```



Modélisation d'une sphère en 3D

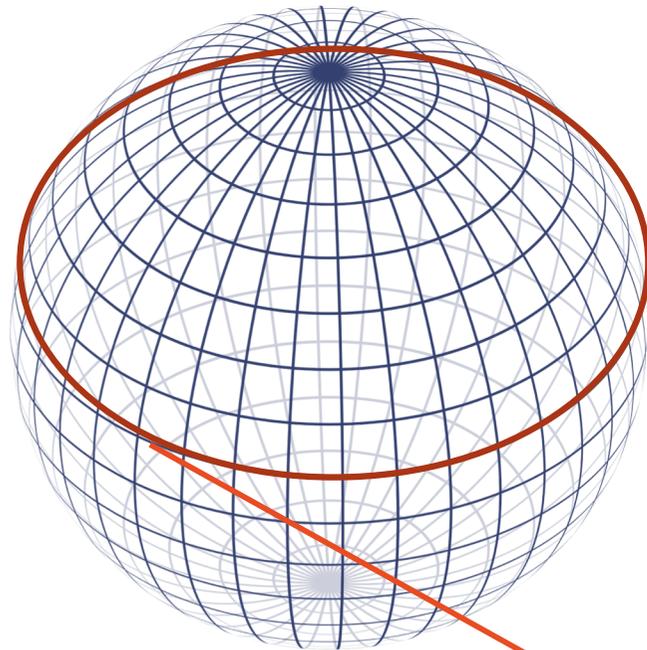


Centrée en $(0, 0, 0)$
De rayon 1

Image issue de Wikipédia

Modélisation d'une sphère en 3D

Centrée en (0, 0, 0)
De rayon 1



Définition des sommets

Sommets du cercle (de rayon r) de coordonnées :

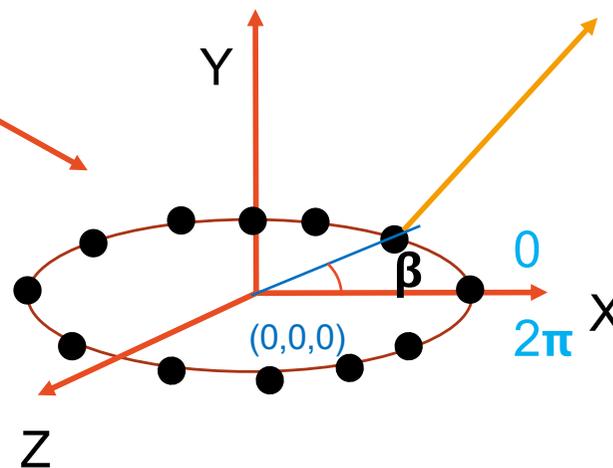
$$x = r * \cos \beta$$

$$y = ?$$

$$z = r * \sin \beta$$

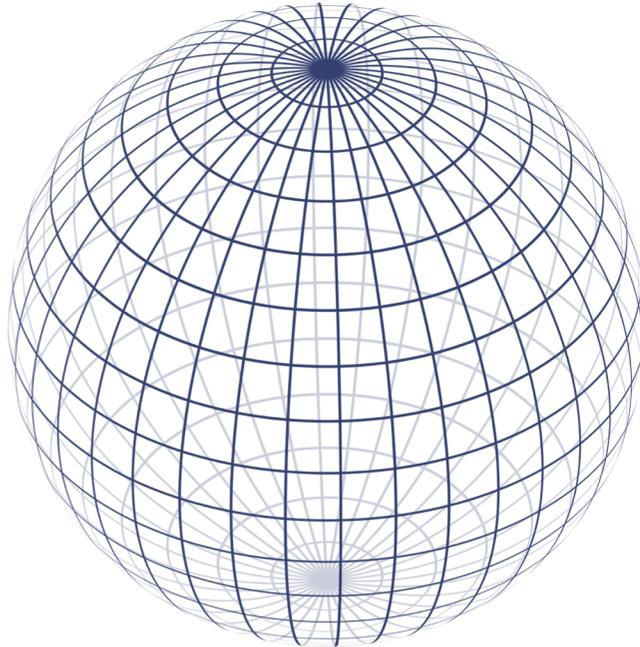
Variation de l'angle β de 0 à 2π

Sphère =
superposition de
cercles



Modélisation d'une sphère en 3D

Centrée en $(0, 0, 0)$
De rayon 1



Sphère =
superposition de
cercles

Questions :

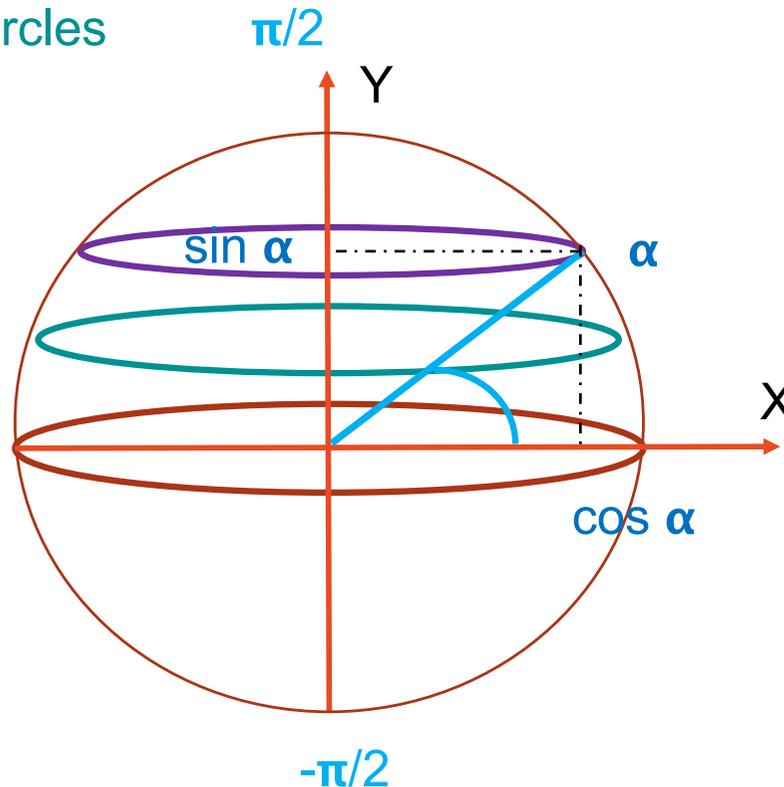
- Rayon r des cercles superposés ?
- Position en y de ces cercles ?

Modélisation d'une sphère en 3D

Centrée en $(0, 0, 0)$
De rayon 1

Sphère = superposition de cercles

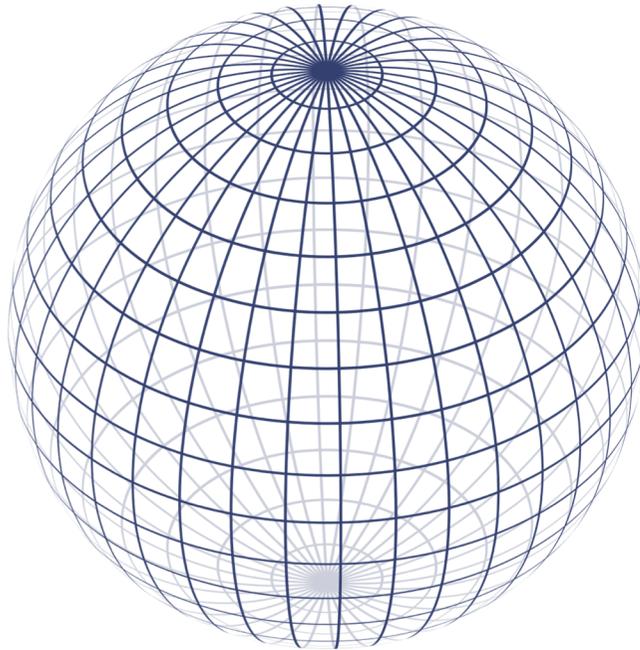
Variation de l'angle α
de $-\pi/2$ à $\pi/2$



- Rayon r des cercles superposés = $\cos \alpha$
- Position en y de ces cercles = $\sin \alpha$

Modélisation d'une sphère en 3D

Centrée en (0, 0, 0)
De rayon 1



Définition des sommets

Sommets de la sphère de coordonnées :

$$x = \cos \alpha * \cos \beta$$

$$y = \sin \alpha$$

$$z = \cos \alpha * \sin \beta$$

Variation de l'angle β de 0 à 2π

Variation de l'angle α de $-\pi/2$ à $\pi/2$

Modélisation d'une sphère en 3D

Centrée en (0, 0, 0)
De rayon 1

```
void ViewerEtudiant::init_sphere()
{
    // Variation des angles alpha et beta
    const int divBeta = 16;
    const int divAlpha = divBeta/2;
    int i,j;

    float beta, alpha, alpha2;

    // Choix de la primitive OpenGL
    m_sphere = Mesh(GL_TRIANGLE_STRIP);
}
```

Modélisation d'une sphère en 3D

Centrée en (0, 0, 0)
De rayon 1

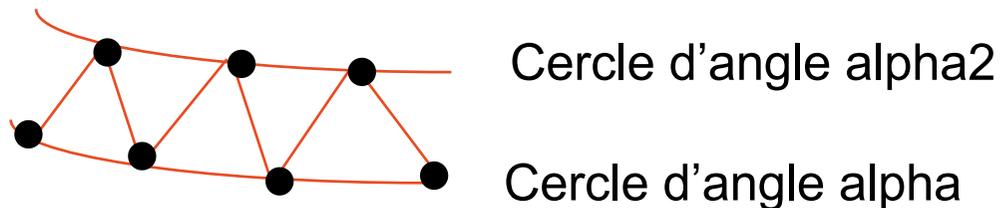
```
// Variation des angles alpha et beta
for(int i=0; i<divAlpha; ++i)
{
    alpha = -0.5f * M_PI + float(i) * M_PI / divAlpha;
    alpha2 = -0.5f * M_PI + float(i+1) * M_PI / divAlpha;

    for(int j=0; j<=divBeta; ++j)
    {
        beta = float(j) * 2.f * M_PI / (divBeta);
```

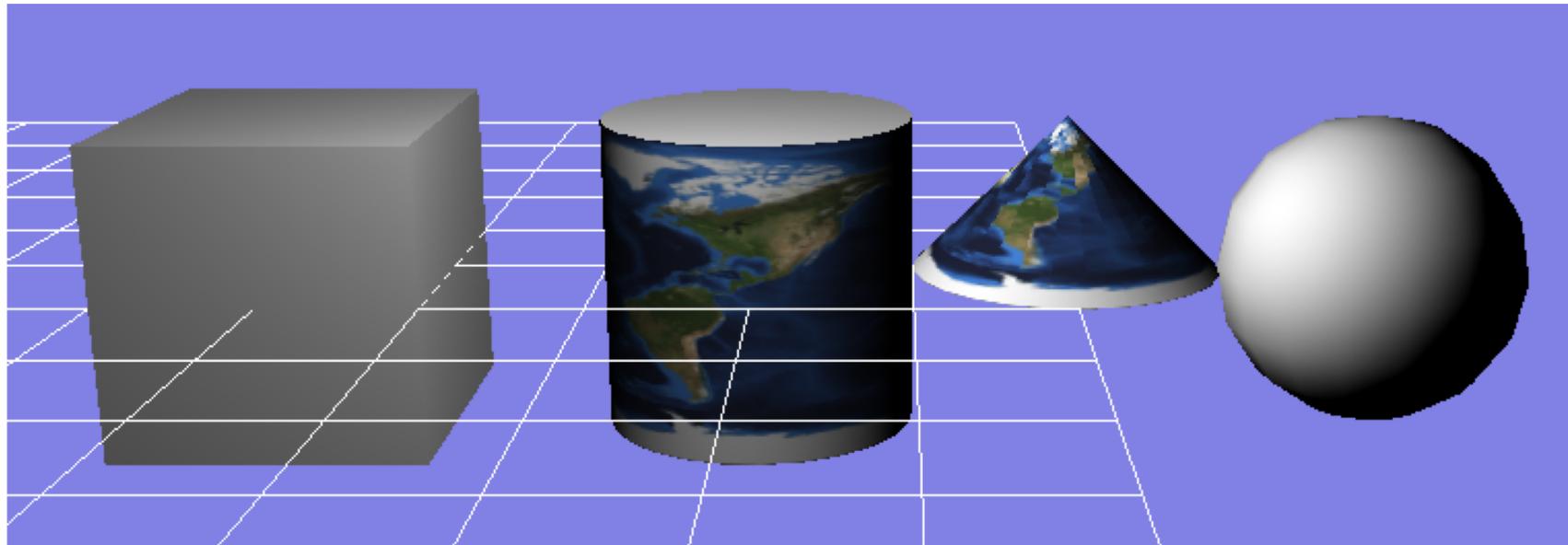
Modélisation d'une sphère en 3D

Centrée en (0, 0, 0)
De rayon 1

```
m_sphere.normal( Vector(cos(alpha)*cos(beta), sin(alpha), cos(alpha)*sin(beta)) );  
m_sphere.vertex( Point(cos(alpha)*cos(beta), sin(alpha), cos(alpha)*sin(beta)) );  
  
m_sphere.normal( Vector(cos(alpha2)*cos(beta), sin(alpha2), cos(alpha2)*sin(beta)) );  
m_sphere.vertex( Point(cos(alpha2)*cos(beta), sin(alpha2), cos(alpha2)*sin(beta)) );  
  
} // boucle sur les j, angle beta, dessin des sommets d'un cercle  
  
m_sphere.restart_strip(); // Demande un nouveau strip  
  
} // boucle sur les i, angle alpha, sphère = superposition de cercles  
  
} // void
```



Affichage des formes de base créées



Affichage des formes de base créées

- 1) Appel des fonctions créées dans `ViewerEtudiant::init()`
- 2) **Affichage** des **Mesh** ainsi créés dans `ViewerEtudiant::render()`

```
int ViewerEtudiant::render()
{
    // Transformation géométrique appliquée au Mesh

    // Cas où pas de transformation géométrique
    Transform T_init = Identity();

    gl.model( T_init );

    gl.draw( m_sphere );
}

// Matrice identité
// Applique une transformation géométrique
// Affichage du Mesh
// Définition de la transformation géométrique de type Transform
```

Affichage d'un objet composé de plusieurs formes

- Affichage d'un cylindre « fermé »
 - Corps du cylindre
 - Utilisation du `Mesh m_cylindre`
 - Disque pour remplir le haut du cylindre
 - Utilisation du `Mesh m_disque`
 - Disque pour remplir le bas du cercle
 - Utilisation du `Mesh m_disque`

Définition de la fonction `draw_cylindre(const Transform & T)`

T = transformation géométrique pour positionner l'objet dans la scène

Affichage d'objets composés de plusieurs formes

```
int ViewerEtudiant::render()
{

    T = Translation( 2, 0, 0 );

    // Appel de la fonction qui va faire l'affichage
    // en tenant compte d'une transformation géométrique

    draw_cylindre(T);

}
```

On souhaite que les 3 éléments constituant le cylindre subissent la transformation géométrique T

Affichage d'objets composés de plusieurs formes

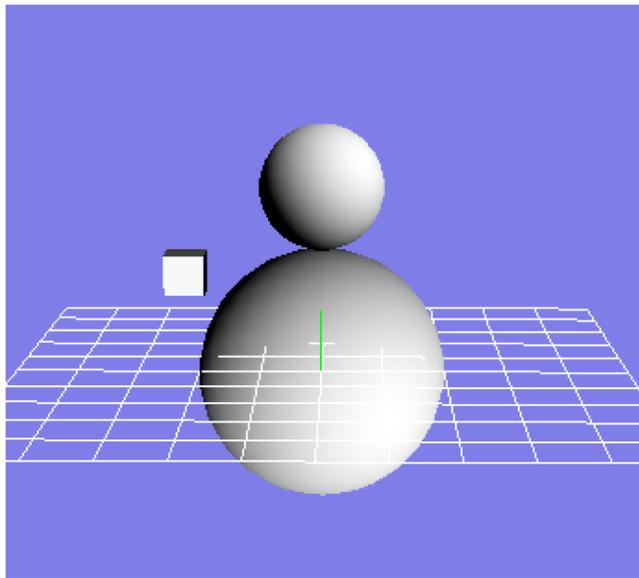
```
void ViewerEtudiant::draw_cylinder(const Transform& T)
{
    // Corps du cylindre
    gl.model( T );
    gl.draw( m_cylindre );

    // Disque du haut
    Transform Tch = T * Translation( 0, 1, 0 );
    gl.model( Tch );
    gl.draw( m_disque );

    // Disque du bas
    Transform Tcb = T * Translation( 0, -1, 0 )
                    * Rotation( Vector(1,0,0), 180 );
    gl.model( Tcb );
    gl.draw( m_disque );
}
```

Transformation
géométrique
pour tous les
éléments de
l'objet

Affichage d'un objet complexe



- Un corps blanc
 - de diamètre 4
 - centre en $(0,0,0)$;
- Une tête blanche
 - de diamètre 2 ;

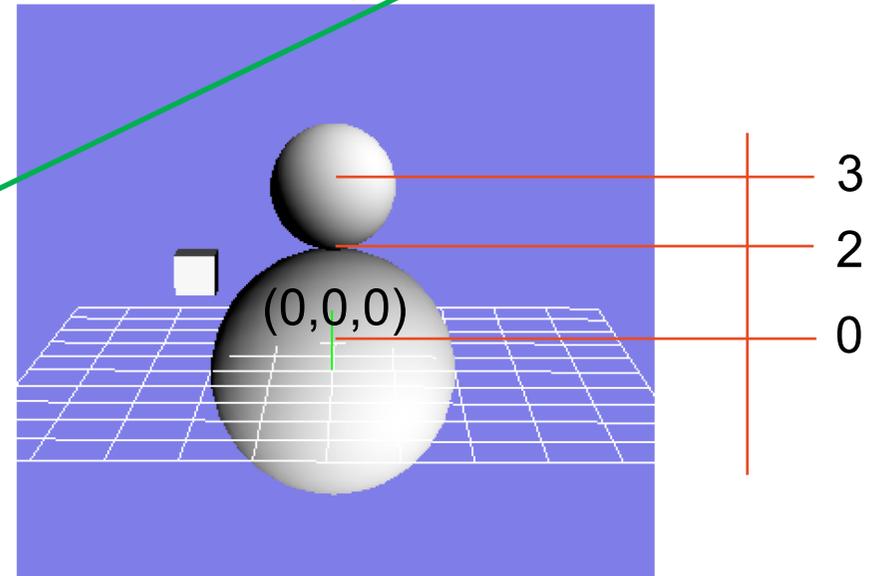
Affichage en utilisant le `Mesh m_sphere` créé par la fonction `init_sphere()`

Affichage d'un objet complexe

Transformation géométrique pour tous les éléments de l'objet

```
void ViewerEtudiant::draw_snow(const Transform & T)
{
    // Corps
    gl.model( T * Scale(2, 2, 2) );
    gl.draw(m_sphere);

    // Tête
    gl.model( T * Translation(0, 3, 0) );
    gl.draw(m_sphere);
}
```



- Un corps blanc
 - de diamètre 4 = rayon 2
 - centre en (0,0,0) ;
- Une tête blanche
 - de diamètre 2 = rayon 1

Puis appel de draw_snow(T) dans render()

Affichage d'un objet complexe

- **En TP, sur le sujet il est noté de réaliser un avion**
 - On vous demandera de varier les objets complexes
 - Des idées : avion, fusée complexe, voiture, vélo avec porte bagage, bâtiment, personnage, animal, etc.