Université Gustave Eiffel

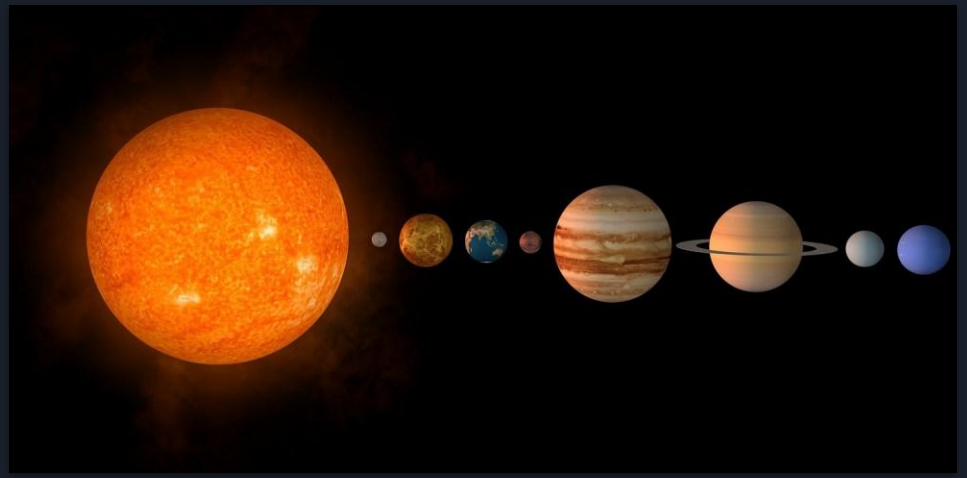
INSTITUT D'ÉLECTRONIQUE ET D'INFORMATIQUE GASPARD-MONGE

Système solaire OpenGL/C3GA

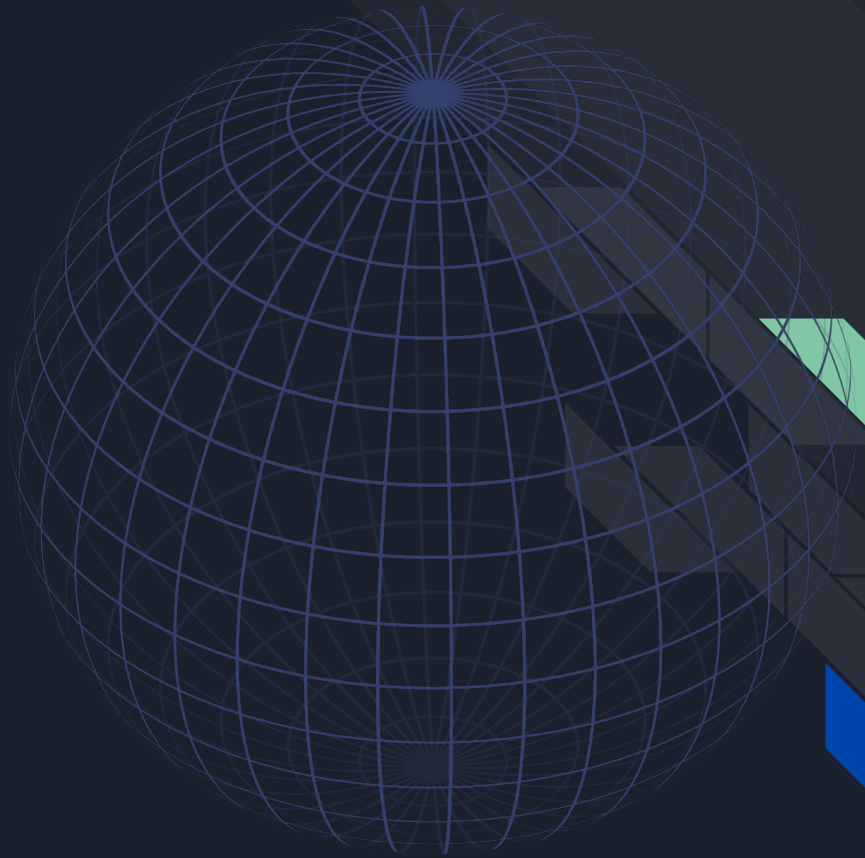
BEN HAMOUDA Amel
HERBRETEAU Amélie

Sommaire

- La sphère
- Les transformations
  - Translation
  - Scale
  - Rotation
- Les astres
- Démo

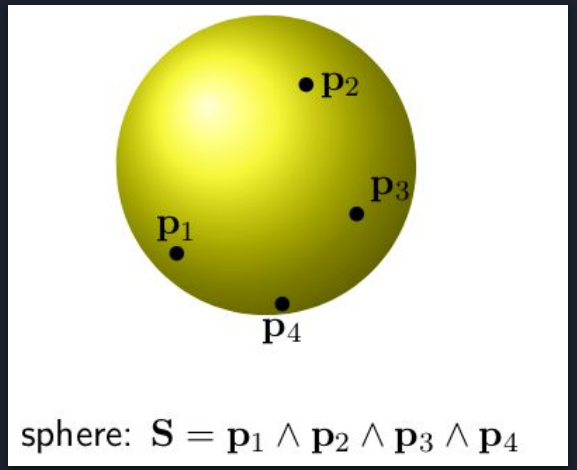


La sphère



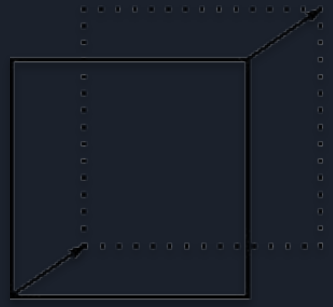
Formule C3GA

Application

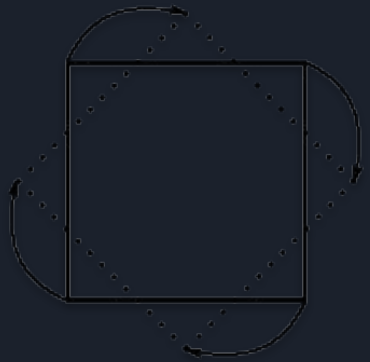



```
c3ga::Mvec<double> Sphere::sphere(float Rsphere) {
    c3ga::Mvec<double> x1, x2, x3, x4, p1, p2, p3, p4;
    c3ga::Mvec<double> dualSphere;
    x1 = (0.0 * c3ga::e1<double>()) + (0.0 * c3ga::e2<double>()) + (Rsphere * c3ga::e3<double>());
    p1 = c3ga::e0<double>() + x1 + (0.5 * x1.quadraticNorm()*c3ga::ei<double>());

    x2 = (0.0 * c3ga::e1<double>()) + (0.0 * c3ga::e2<double>()) + (-Rsphere * c3ga::e3<double>());
    p2 = c3ga::e0<double>() + x2 + (0.5 * x2.quadraticNorm()*c3ga::ei<double>());

    x3 = (Rsphere * c3ga::e1<double>()) + (0.0 * c3ga::e2<double>()) + (0.0 * c3ga::e3<double>());
    p3 = c3ga::e0<double>() + x3 + (0.5*x3.quadraticNorm()*c3ga::ei<double>());

    x4 = (0.0 * c3ga::e1<double>()) + (Rsphere * c3ga::e2<double>()) + (0.0 * c3ga::e3<double>());
    p4 = c3ga::e0<double>() + x4 + (0.5 * x4.quadraticNorm()*c3ga::ei<double>());

    s = (p1 ^ p2 ^ p3 ^ p4);
    return s;
}
```

Les transformations



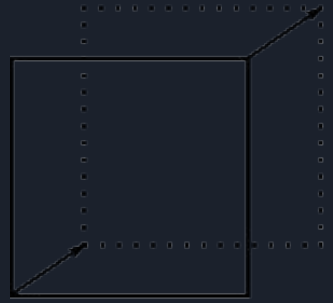
Translation



Rotation

Scaling

Translation



Formule C3GA

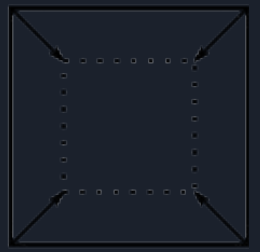
$T \times O \times T^{-1}$

Avec $T = 1 - \frac{1}{2}\mathbf{t}e_{\infty}$, où **t** est le vecteur de translation et **O** l'objet à translater.

Application

```
c3ga::Mvec<double> Transformation::translate(c3ga::Mvec<double> vect, double facteur, c3ga::Mvec<double> translation) {
    c3ga::Mvec<double> translator = 1 - 0.5 * translation * c3ga::ei<double>() * facteur;
    vect = translator * vect * translator.inv();
    return vect;
}

glm::vec3 Transformation::applyTranslationX(Sphere sphere) {
    c3ga::Mvec<double> s = sphere.getSphere();
    return glm::vec3(abs(s[c3ga::E0123]) + abs(s[c3ga::E123i]) + abs(s[c3ga::E0123i]), 0, 0);
}
```

Scale



Formule C3GA

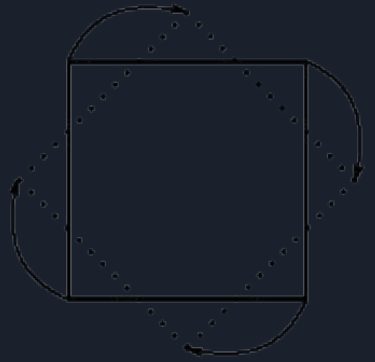
$D \times O \times D^{-1}$

Avec $D = 1 - \frac{1-s}{1+s}\mathbf{e}_{o\infty}$, où **s** est le facteur de scale et **O** l'objet à scaler.

Application

```
c3ga::Mvec<double> Transformation::scale(c3ga::Mvec<double> vect, double scale) {
    c3ga::Mvec<double> dilator = 1. - ((1. - scale) / (1. + scale)) * c3ga::e0i<double>();
    vect = dilator * vect * dilator.inv();
    vect.roundZero(1.0e-10);
    return vect;
}

glm::vec3 Transformation::applyScale(Sphere sphere) {
    c3ga::Mvec<double> s = sphere.getSphere();
    return glm::vec3(s[c3ga::E0123], s[c3ga::E0123], s[c3ga::E0123]);
}
```

Rotation



Formule C3GA

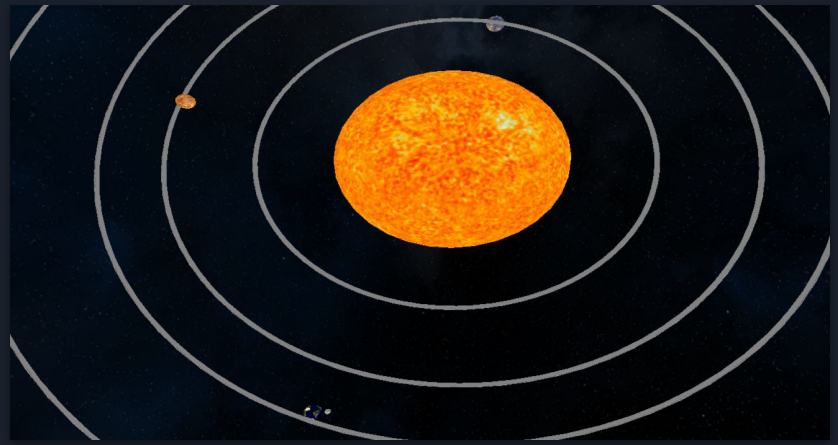
$R \times O \times R^{-1}$

Avec $R = \cos(\frac{1}{2}\alpha) - B\sin(\frac{1}{2}\alpha)$, où $\boldsymbol{\alpha}$ est l'angle de rotation, **B** le bi-vecteur associé à l'axe de rotation et **O** l'objet à rotater.

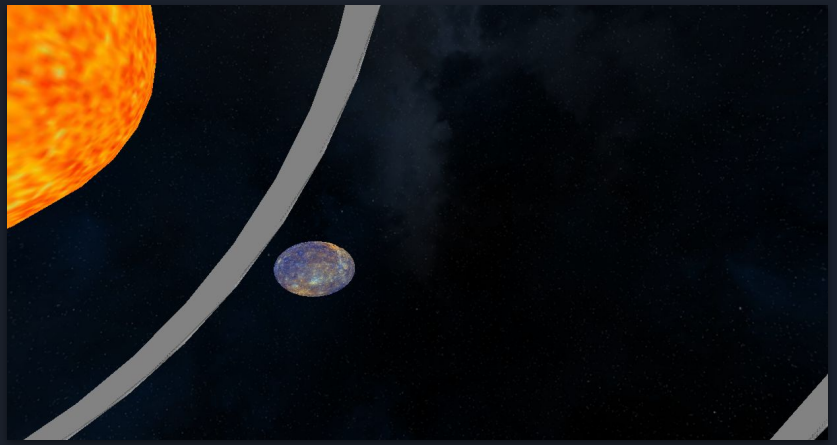
Application

```
c3ga::Mvec<double> Transformation::rotate(c3ga::Mvec<double> vect, double angle, c3ga::Mvec<double> biVect) {
    c3ga::Mvec<double> rotor = cos(0.5 * angle * M_PI / 180) - biVect * sin(0.5 * angle * M_PI / 180);
    vect = rotor * vect * rotor.inv();
    vect.roundZero(1.0e-10);
    return vect;
}

glm::vec3 Transformation::applyRotation(Sphere sphere) {
    c3ga::Mvec<double> s = sphere.getSphere();
    return glm::vec3(s[c3ga::E0123], s[c3ga::E0123], s[c3ga::E0123]);
}
```

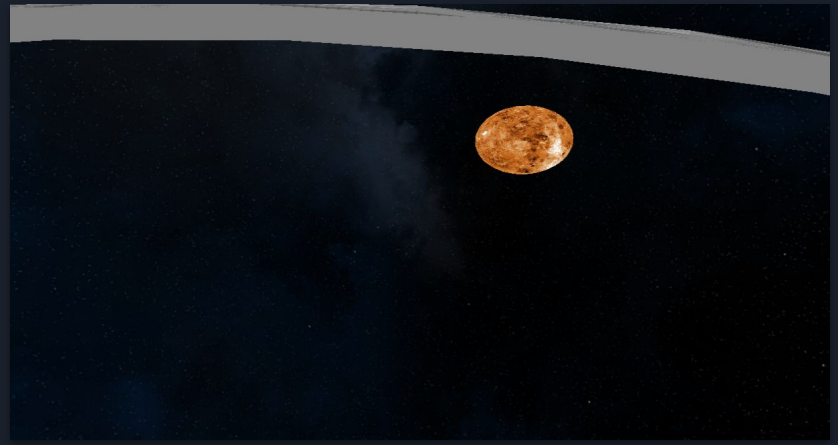
Les astres



Soleil

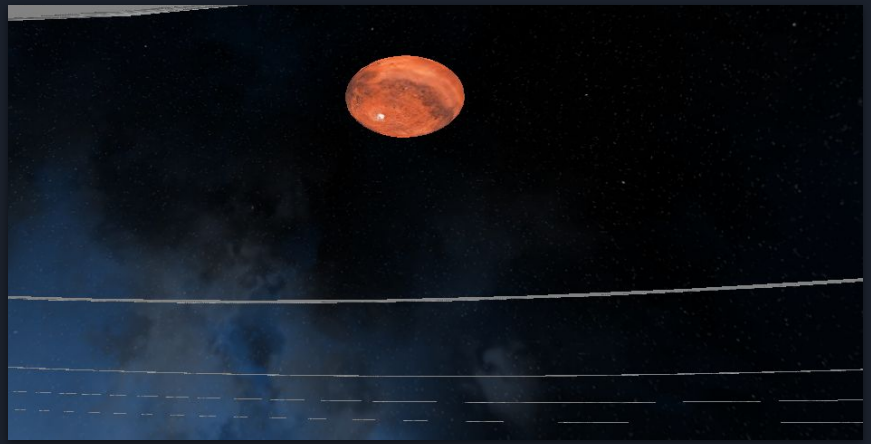


Mercure

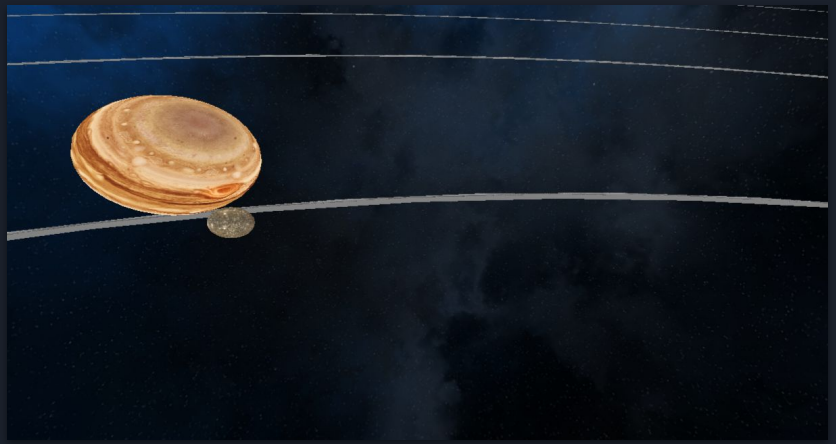


Vénus

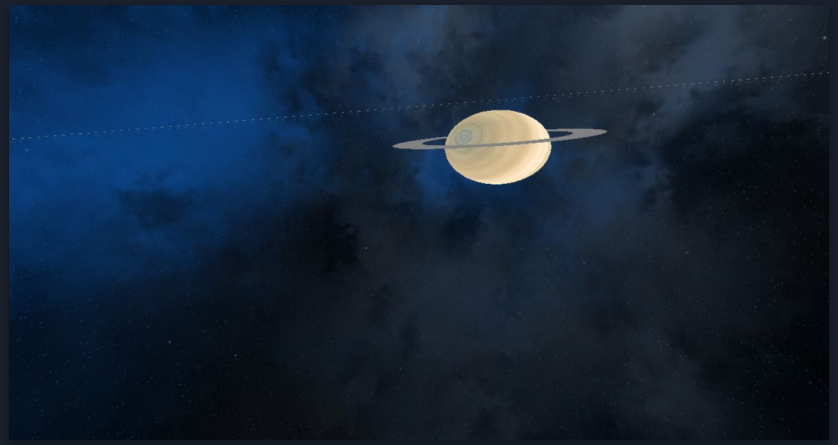
Terre et Lune



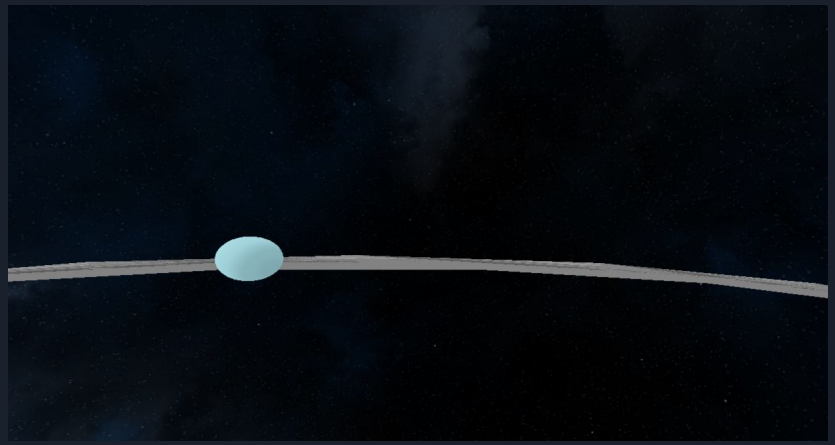
Mars



Jupiter et Callisto

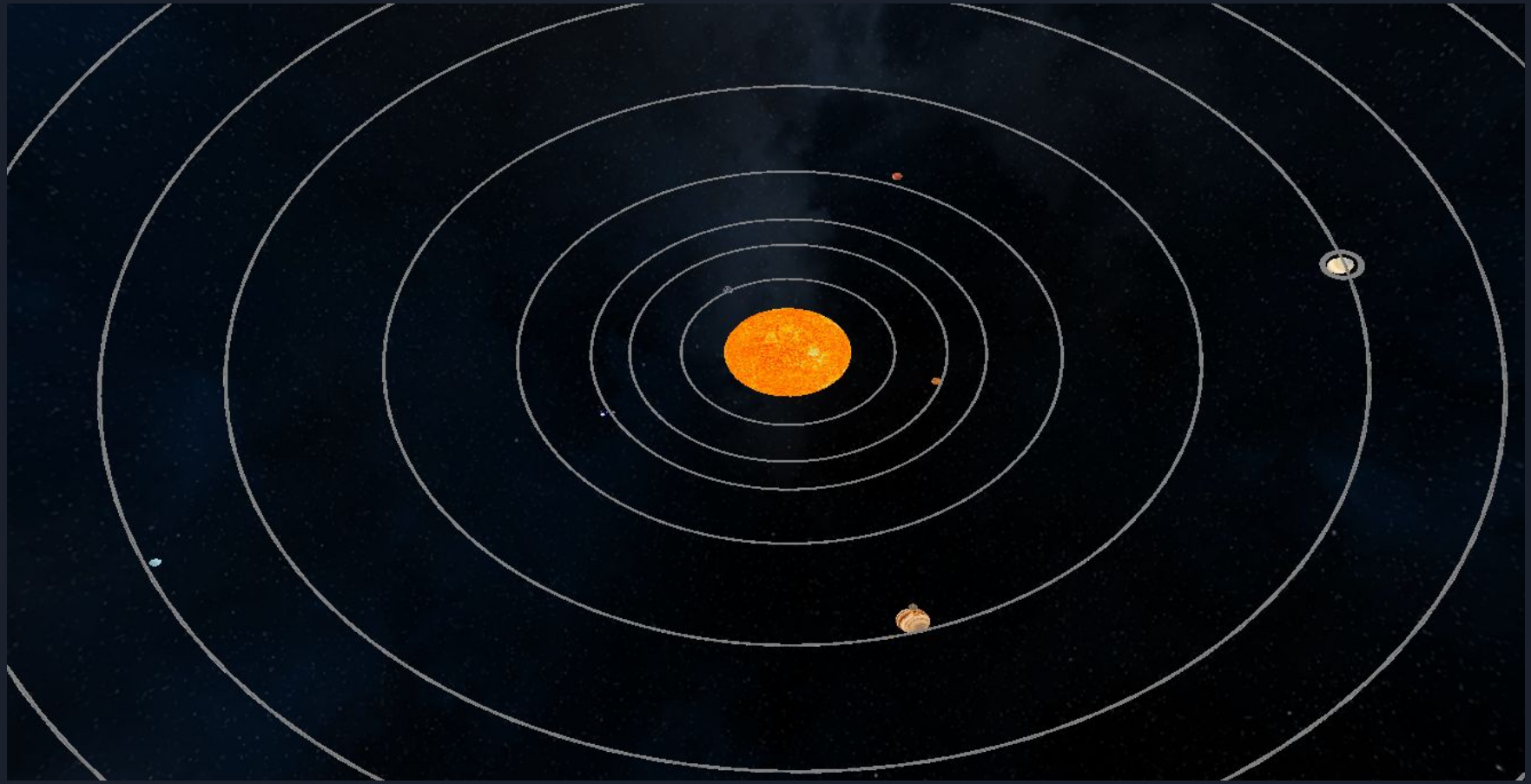


Saturne



Uranus

Neptune



Système solaire

Démo du projet