

The background is a dark blue-grey color. It features several thin, gold-colored lines that form abstract, angular shapes. These lines radiate from the central text box, extending towards the corners and edges of the frame, creating a sense of dynamic movement and structure.

Projet Modélisation

Groupe G5

ALMEIDA Néo, LAGNEAU Simon, MACIEIRA Matteo, VANHEE Paul

Sommaire

Démo du projet

Différences Livrable
1/Livrable 2

Influences

Fonctionnalités

Livrable 1
Livrable 2

Éléments techniques

Designs Pattern
MVC
Tests Unitaires
Différentes méthodes

Organisation du projet

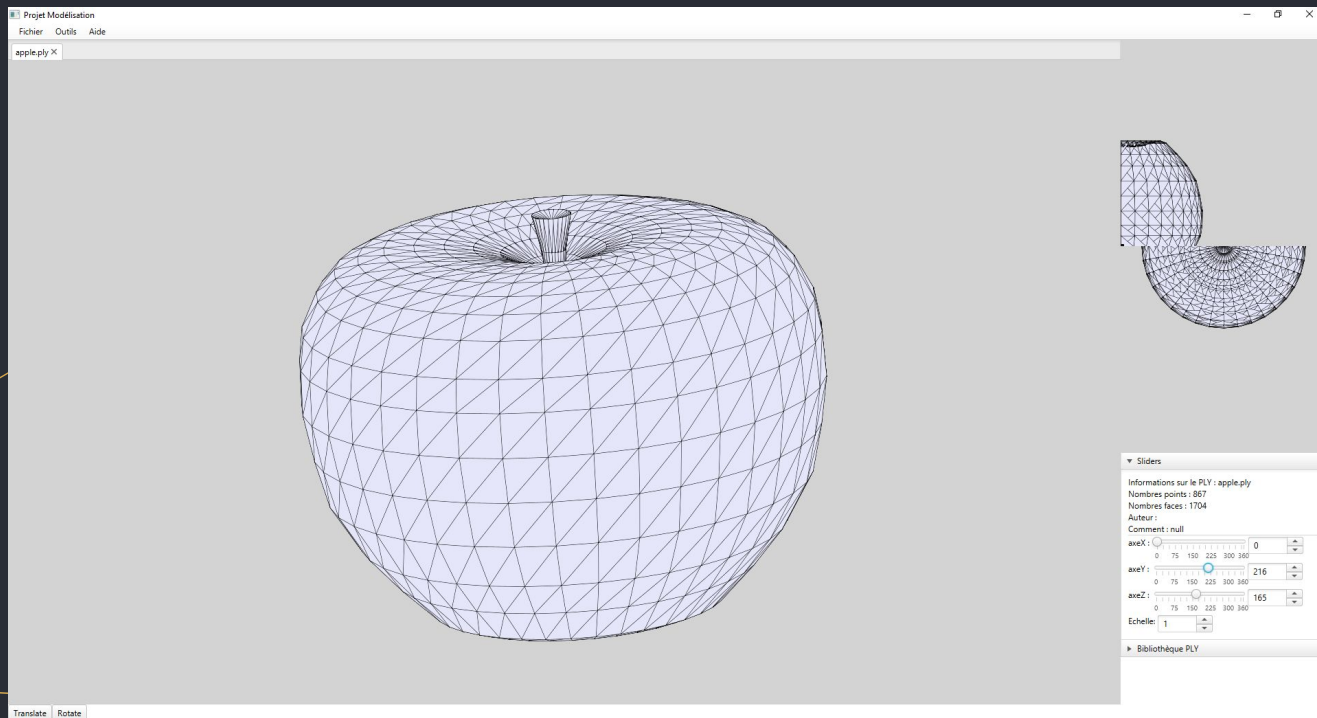
Rôle de chacun des
acteurs du projet

Démo Projet

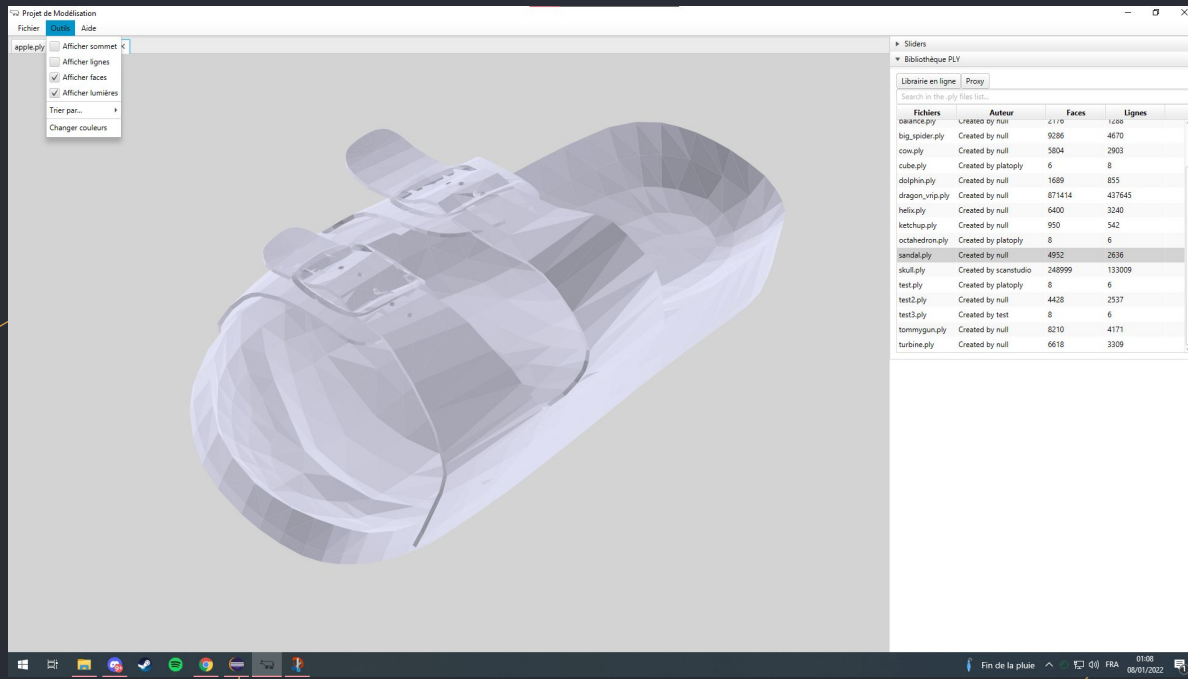


Différences Livrable 1/Livrable 2

Livrable 1







Livrable 2











Organisation du projet

Open 2 Closed 22 All 24

[Edit issues](#)[New issue](#)

Recent searches ▾ Search or filter results...

Created date ▾ ⌵

| | |
|---|--|
| Changer le graphic des ThemesButtons Rectangle -> Canvas #24 · opened 3 months ago by Néo Almeida | CLOSED  0 updated 2 months ago |
| Initier model3d (init)z #23 · opened 3 months ago by Néo Almeida | CLOSED  0 updated 2 months ago |
| Créer le model 3d (avant affichage) #22 · opened 3 months ago by Néo Almeida | CLOSED  0 updated 2 months ago |
| IJFX : Popup : Erreurs ?? #21 · opened 3 months ago by Néo Almeida | CLOSED  1 updated 1 minute ago |
| IJFX : Canvas : Interactif souris / flèches clavier #20 · opened 3 months ago by Néo Almeida Doing | CLOSED  1 updated 1 minute ago |
| IJFX : Canvas : En charger plusieurs ??? #19 · opened 3 months ago by Néo Almeida | CLOSED  0 updated 2 months ago |
| IJFX : Canvas : Recevoir modèle valide #18 · opened 3 months ago by Néo Almeida | CLOSED  1 updated 1 minute ago |
| Tests Unitaires Fichiers Erreurs Calculs Maths (Rotation et cie) #17 · opened 3 months ago by Néo Almeida |  0 updated 3 months ago |

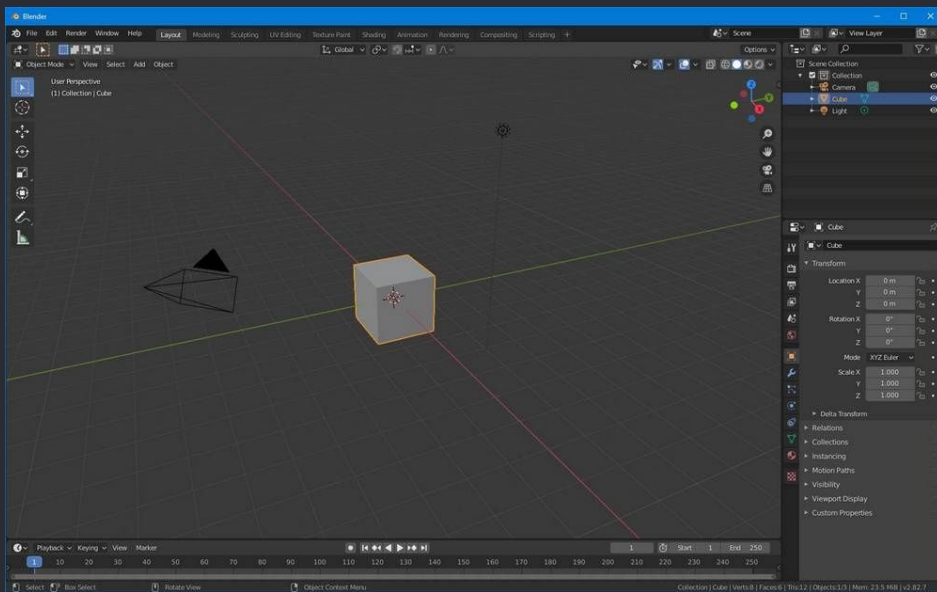


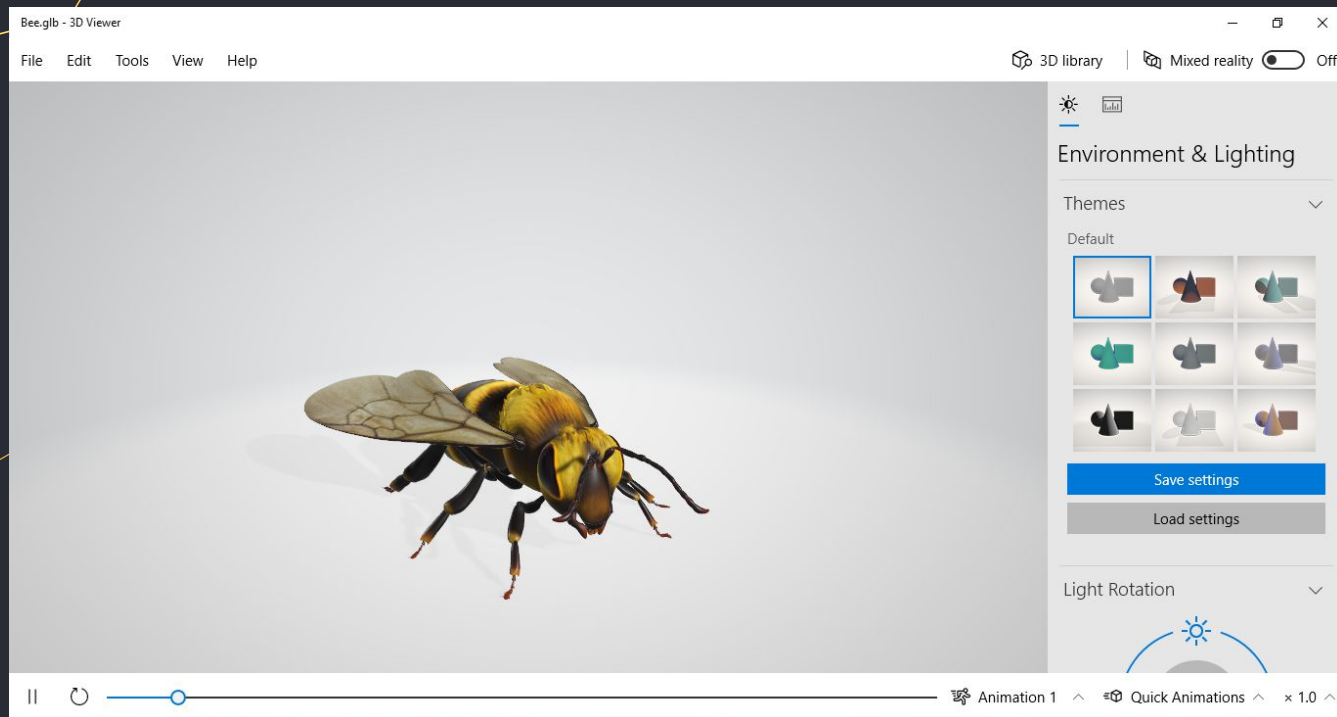
GitLab



CodeTogether

Influences interface





Fonctionnalités

Livrable 1 | Livrable 2

Fonctionnalités : Livrable 1

| Demandées | Supplémentaires |
|---|---|
| <ul style="list-style-type: none">- Librairie pour choisir les PLY- 3 Vues : Vue principale, vue de droite, vue de haut- Transformer le modèle :<ul style="list-style-type: none">- Faire tourner- Changer d'échelle- Translation $\leftarrow \uparrow \downarrow \rightarrow$- Les vues réagissent en conséquences- Le contrôle s'effectue grâce à des boutons dans l'interface graphique | <ul style="list-style-type: none">+ Ouvrir plusieurs modèles+ Bouger le modèle grâce à la souris (en fonction des boutons cochés)+ Changer les thèmes+ Exporter en .png+ Choisir la manière d'afficher (faces/arêtes)+ Raccourcis clavier (voir fenêtre de contrôles)+ Ouvrir récents |

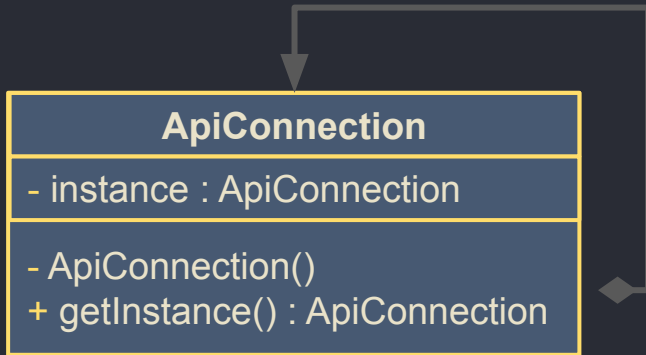
Fonctionnalités : Livrable 2

| Demandées | Supplémentaires |
|---|---|
| <ul style="list-style-type: none">- MVC- Affichage des faces et/ou segments- Affichage centré- Amélioration de la librairie PLY- Éclairage- Contrôleur horloge | <ul style="list-style-type: none">+ Ombres+ Bibliothèque en ligne<ul style="list-style-type: none">> Proxy pour l'université+ Stratégie de rendu+ Changement de couleur (non permanent) des sommets, arrêtes, faces, arrière-plan |

Éléments techniques

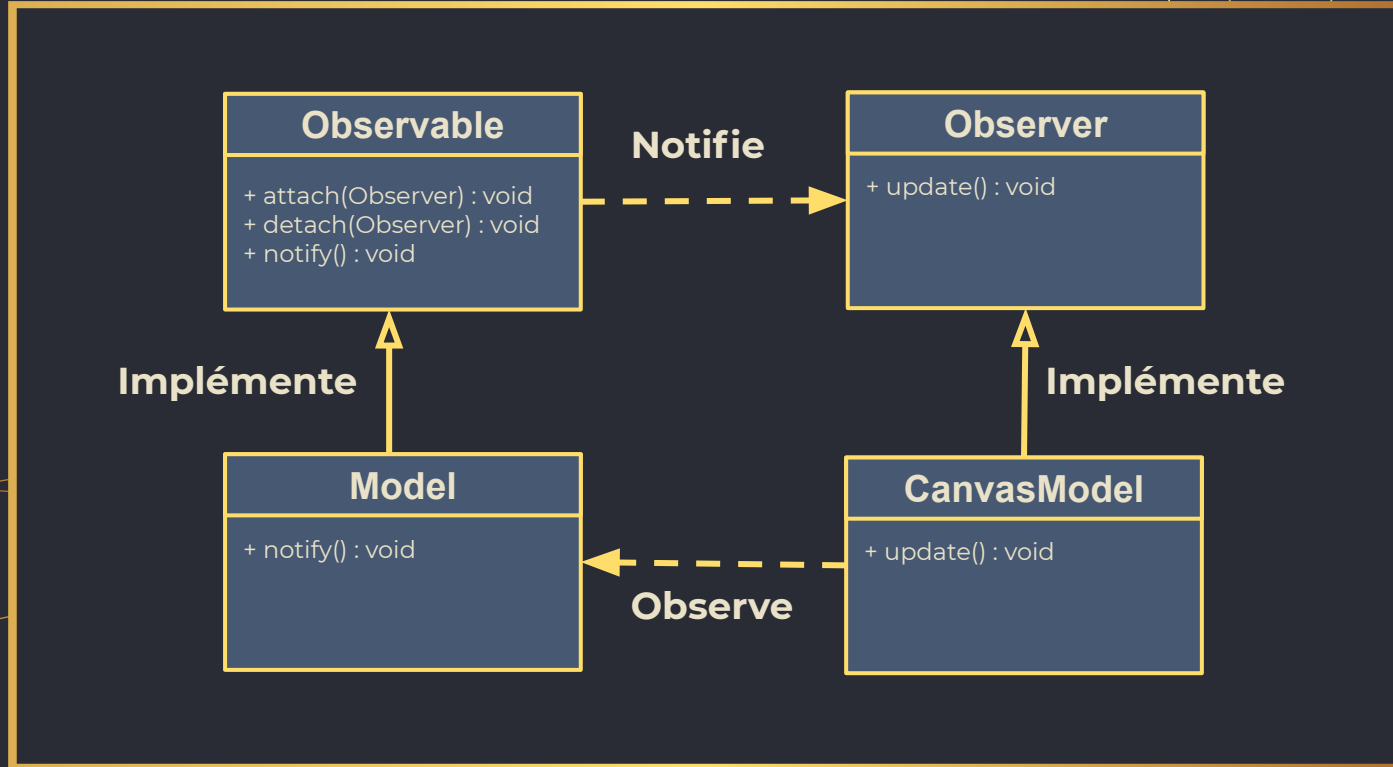
Designs Patterns | MVC | Tests Unitaires | Méthodes

Design Pattern : Singleton









- Une seule connexion !
- La première connexion est plus lente, mais les suivantes sont très rapides !

Design Pattern : Observers Observés




Design Pattern : MVC


Modèle

| | |
|--|--|
| ▼  model | Relatif ... |
| >  io | Aux fichiers |
| >  maths | Aux mathématiques et aux calculs |
| >  models | Au modèle que nous pouvons lire dans I/O |
| >  observers | Au design pattern Observer |
|  InternetUtil | À notre API |


Vue/Contrôleur

▼  view


Relatif ...

▼  components

Aux composants de notre application ...

>  items


“basique”

>  render


spécifique au rendu

>  control

Aux zones spécifique de notre BorderLayout

>  stages

Aux différentes fenêtres ouvrables

>  utils

Aux classes utilitaires

Tests Unitaires

Test calculs matriciels

| | |
|---------------------------------|-------|
| ✓ Test Results | 51 ms |
| ✓ MatrixTest | 51 ms |
| ✓ test_can_multiply_matrix() | 31 ms |
| ✓ test_translation_matrix() | 7 ms |
| ✓ test_multiply_matrix_matrix() | 2 ms |
| ✓ test_can_sum_matrix() | 4 ms |
| ✓ test_sum_matrix_matrix() | 1 ms |
| ✓ test_sub_matrix_matrix() | 3 ms |
| ✓ test_can_sub_matrix() | 3 ms |

Tests fichiers

| | |
|--|--------|
| ✓ Test Results | 128 ms |
| ✓ PlyReaderTest | 128 ms |
| ✓ set_FileInResourceWithFileName_Test() | 99 ms |
| ✓ set_NotExistingFileWithFileObject_Test() | 17 ms |
| ✓ set_FileInResourceWithFileObject_Test() | 5 ms |
| ✓ set_NotExistingFileWithFileName_Test() | 7 ms |

Méthodes calculs matriciels

```
public void multiplyMatrix(Matrix other) {
    if(!this.canMultiply(other)) {
        return;
    }
    double[][] vals;
    int l1 = this.getRowCount();
    int c1 = this.getColumnCount();

    int c2 = other.getColumnCount();

    vals = new double[l1][c2];
    for(int row = 0; row < l1; row++){
        for(int col = 0; col < c2; col++){
            for(int k = 0; k < c1; k++)
                vals[row][col] += this.getValues()[row][k] * other.getValues()[k][col];
        }
    }
    this.values = vals;
}
```

Méthodes rotation

```
public void rotation(Rotation r, double degre) {  
    double[][] vals;  
    if(r.equals(Rotation.X)) {  
        vals = xRotationMatrix(degre);  
    } else if(r.equals(Rotation.Y)) {  
        vals = yRotationMatrix(degre);  
    } else if(r.equals(Rotation.Z)) {  
        vals = zRotationMatrix(degre);  
    } else {  
        throw new IllegalArgumentException("Le type de rotation n'est pas valable.");  
    }  
  
    Matrix matrixVal = new Matrix(vals);  
    matrixVal.multiplyMatrix( other: this);  
  
    this.values = matrixVal.values;  
}
```

« enumeration »

Rotation

+ X

+ Y

+ Z

+ getRotation() : Rotation

Méthodes rotation

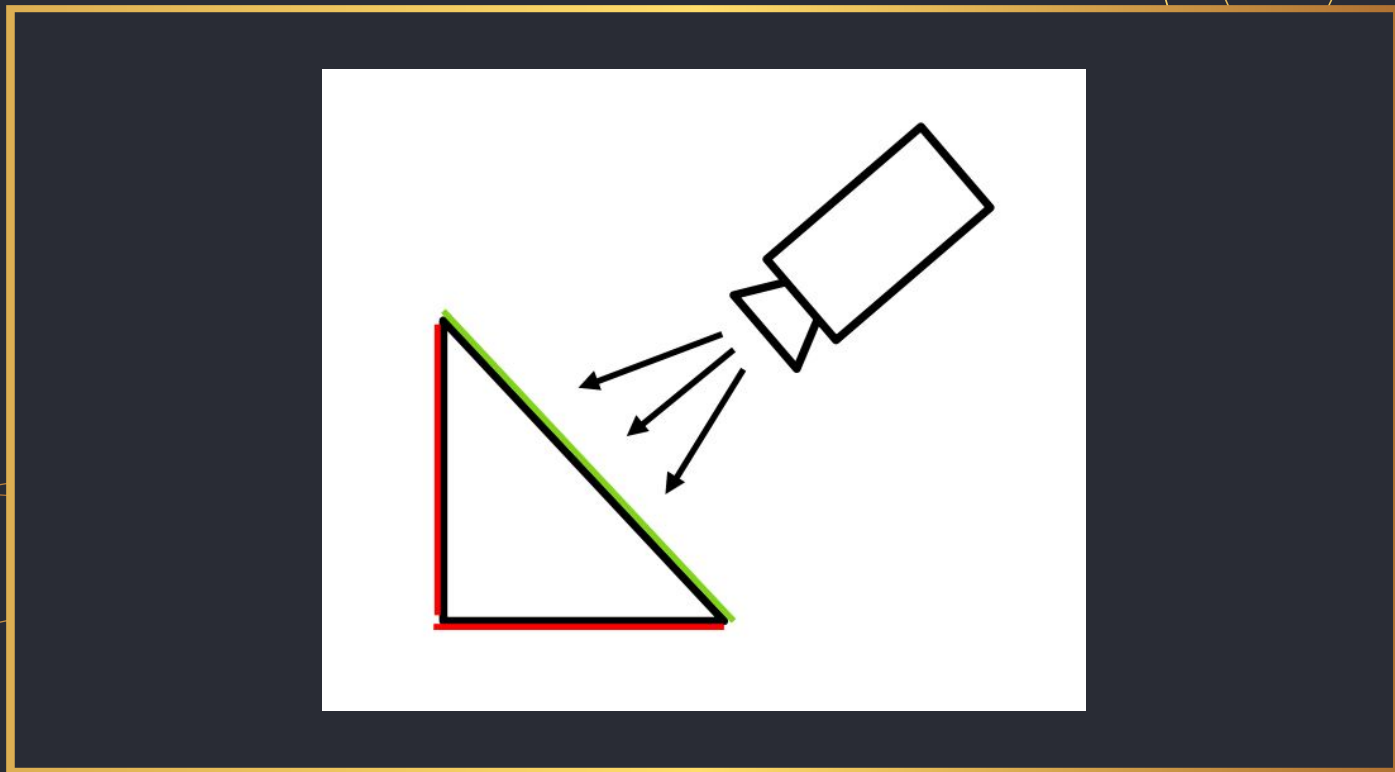
```
public double[][] xRotationMatrix(double degree){  
    return new double[][] {  
        {1, 0, 0, 0},  
        {0, Math.cos(Math.toRadians(degree)), -Math.sin(Math.toRadians(degree)), 0},  
        {0, Math.sin(Math.toRadians(degree)), Math.cos(Math.toRadians(degree)), 0},  
        {0, 0, 0, 1},  
    };  
}  
  
public double[][] yRotationMatrix(double degree){  
    return new double[][] {  
        {Math.cos(Math.toRadians(degree)), 0, Math.sin(Math.toRadians(degree)), 0},  
        {0, 1, 0, 0},  
        {-Math.sin(Math.toRadians(degree)), 0, Math.cos(Math.toRadians(degree)), 0},  
        {0, 0, 0, 1},  
    };  
}  
  
public double[][] zRotationMatrix(double degree) {  
    return new double[][] {  
        {Math.cos(Math.toRadians(degree)), -Math.sin(Math.toRadians(degree)), 0, 0},  
        {Math.sin(Math.toRadians(degree)), Math.cos(Math.toRadians(degree)), 0, 0},  
        {0, 0, 1, 0},  
        {0, 0, 0, 1},  
    };  
}
```

Méthodes normales aux faces : Ombres

```
if(canvasDrawHandler.isDrawLight()) {  
    Vector vectorLumos = new Vector( anX: 0, anY: 0, anZ: -1);  
    double coeffLumos = (Math.cos((vectorLumos.normalisation()).produitScalaire(face.vecteurFace())));  
  
    Color colorLumos = null;  
  
    if(this.facesColor == null) {  
        colorLumos = Color.rgb(face.getColor()[0], face.getColor()[1], face.getColor()[2]);  
    } else {  
        colorLumos = this.facesColor;  
    }  
  
    gc.setFill(  
        Color.rgb((int)((colorLumos.getRed()*255)*coeffLumos),  
            (int)((colorLumos.getGreen()*255)*coeffLumos),  
            (int)((colorLumos.getBlue()*255)*coeffLumos));  
} else {  
    gc.setFill(getFaceColor(face));  
}
```


Méthodes normales aux faces : "Ray casting"

```
private final static Vertex CAMERA = new Vertex( x: 0, y: 0, z: 1);  
public void draw() {  
    setupDrawStuff();  
    for(Face face : this.model.getFaces() ) {  
        if(!this.canvasDrawHandler.isDrawFaces() || face.vecteurFace().dot(CAMERA) > 0) {  
            drawFace(face);  
        }  
    }  
}  
  
public double dot(Vector other) {  
    return this.getX() * other.getX() + this.getY() * other.getY() + this.getZ() * other.getZ();  
}
```



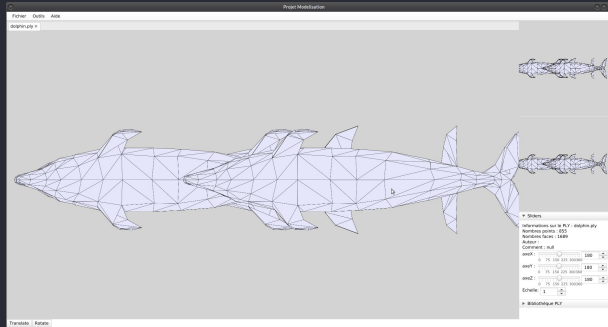
Éléments d'organisation

Implémentations abouties

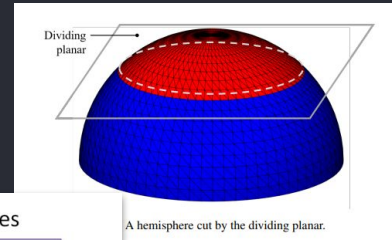
- Documentation @
- Calcul Matriciel N P
- Observer/Observé N
- Affichage (Éclairage, N P)
Algorithme peintre, N
"RayCasting") N
- Contrôleur horloge N M
- Librairie de fichiers N P M
- Interface @
- Thèmes N
- API N P
- Test @
- Gestion dynamique des fichiers N
- Clean code @
- Reader S
- Barre de recherche S
- Exporter en PNG N
- Vidéo de présentation M
- Canvas Réactif N P
- PMD + UML M
- Boutons Échelle P
- Sliders rotations M

Implémentations non abouties

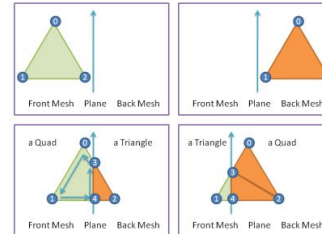
- Tentative d'implémentation des 3 vues **P** **M** **N**



- Tentative d'implémentation d'affichage en tranche **P**



Triangle Slicing – 4 Cases



* There are more degenerate cases (the plane 'falls' on one of the original vertices - not quad generated).

Conclusion

- Pousser nos connaissances en JavaFx
 - Travail en groupe
 - GitLab/GitHub
 - PMD
 - Modélisation 3D