Lancer de Rayon

Guillaume LEMONNIER
Erin LE DREAU
Harisson KOBYLT
Antoine LEMAITRE
Logan VIVEN

L2 info, 2021-2022
Groupe 4A

24 février 2022

Table des matières

1 Introduction

Le lancer de rayon est une technique de visualisation 3D reoposant sur le principe de calculer la distance de l'écran avec un objet afin de connaitre sa taille et donc ça représentation sur l'écran. Cette technique de visualisation 3D est très utilisé dans le domaine de l'informatique, elle est connue sous le nom de Ray Tracing.

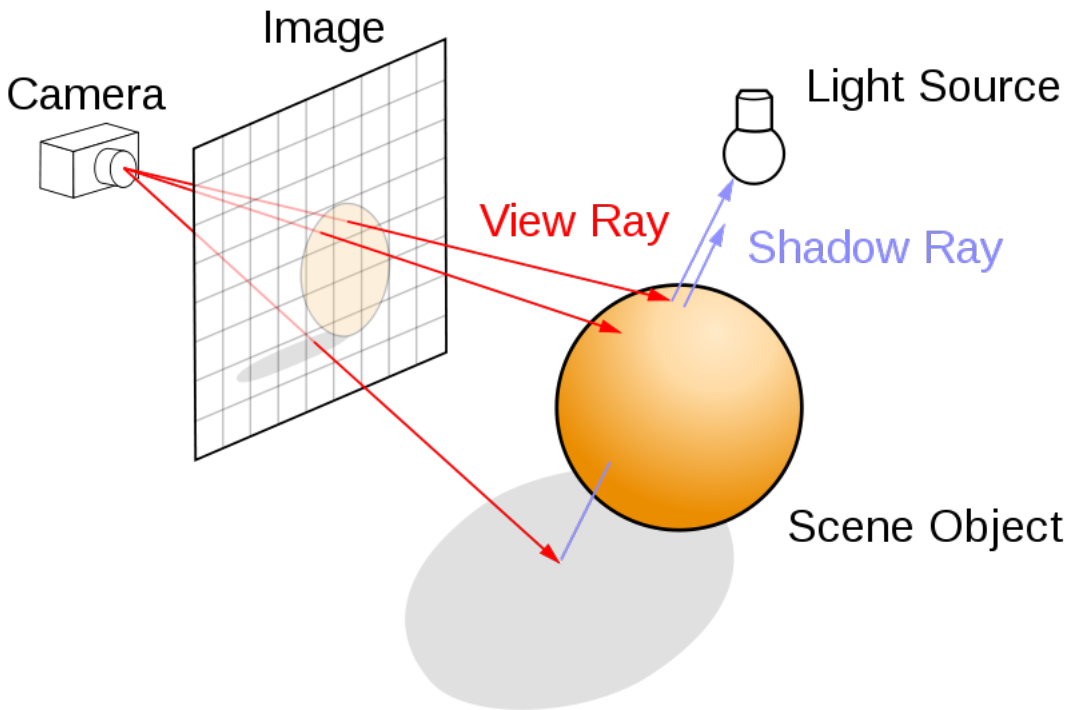
1.1 Objectif

Lors de notre selection de projet nous avons vite pensé à choisir ce projet. En effet il s'agissait du projet qui nous inspiré le plus parmis tout ceux proposé. Nous nous sommes répartit le travail comme suit :

Guillaume LEMONNIER	Model 3D et Rayons
Antoine LEMAITRE	Lumières et Ombres
Harisson KOBYLT	Interface
Logan VIVEN	Parser
Erin LE DREAU	Interface

De ce fait nous nous somme répartit le travail équitablement. Cependant comme il est possible de le voir. Erin et Harisson se sont retrouvé sur la même partie. La raison est que Erin n'avait à l'origine pas de groupe. Elle nous a rejoint au bout du cours numéro 3 et nous avons donc du la mettre avec quelqu'un, car le travail été déjà équitablement répartit.

L'objectif était donc de programmer le principe du schema suivant :



2 Partie Guillaume

2.1 ray

Lors de la première séance nous avons mis sur le papier nos rôles. Guillaume, ayant déjà une bonne idée de ce qu'il devait faire, a commencé les Vector2D, Vector3D et Ray qu'il a fini pendant la première séance sans trop de difficultées.

2.1.1 Vector2D

Algorithme 1 : constructor

```
1 this.(0, 0);
```

Algorithme 2 : constructor

Entrées : Int x, y

```
1 this.setX(x);
2 this.setY(y);
```

Algorithme 3 : constructor

Entrées : Vector2D $vector$

```
1 si vector == null alors
2 |  Exception
3 fin
4 this.setX(vector.getX());
5 this.setY(vector.getY());
```

Algorithme 4 : getX

Output : Int x

1 **retourner** $this.x$

Algorithme 5 : getY

Output : Int y

1 **retourner** $this.y$

Algorithme 6 : setX

Entrées : Int x

1 $this.x \leftarrow x$

Algorithme 7 : setY

Entrées : Int y

1 $this.y \leftarrow y$

Algorithme 8 : add

Entrées : Int x, y

1 $this.setX(this.getX() + x)$;

2 $this.setY(this.getY() + y)$;

Algorithme 9 : add

Entrées : Vector2D $vector$

1 **si** $vector == null$ **alors**

2 | Exception

3 **fin**

4 $this.add(vector.getX(), vector.getY())$;

Algorithme 10 : multiply

Entrées : Int x, y

1 $this.setX(this.getX() * x)$;

2 $this.setY(this.getY() * y)$;

Algorithme 11 : multiply

Entrées : Vector2D $vector$

1 **si** $vector == null$ **alors**

2 | Exception

3 **fin**

4 $this.multiply(vector.getX(), vector.getY())$;

Algorithme 12 : toString

Output : String

1 **retourner** $"(" + this.x + "," + this.y + ")"$;

Algorithme 13 : rotate

Entrées : Double $angle$
Données : Int x, y

```
1 x ← (Math.cos(angle) * this.x − Math.sin(angle) * this.y);
2 y ← (Math.sin(angle) * this.x + Math.cos(angle) * this.y);
3 this.setX(x);
4 this.setY(y);
```

Algorithme 14 : normalize

Output : Vector2D
Données : Double $length$

```
1 length ← Math.sqrt(this.x * this.x + this.y * this.y);
2 si length == 0 alors
3 |  Exception
4 fin
5 retourner newVector2D(this.x/length, this.y/length);
```

2.1.2 Vector3D

Algorithme 15 : constructor

```
1 super(0, 0);
2 this.setZ(0);
```

2.1.3 Ray

2.2 Models 3D

Lors de la deuxième séance il a commencer à creer un cube en 3D avec juste la position de ses sommets comme référence. Il a aussi commencer à faire des graphs afin de mieux représenter les modeles 3D. Les models 3D sont uniquements un Point avec une liste de Points voisin.

Algorithme 16 : constructor

Entrées : Int x, y, z

1 $super(x, y)$;
2 $this.setZ(z)$;

Algorithme 17 : constructor

Entrées : Double x, y, z

1 $super(x, y)$;
2 $this.setZ(z)$;

Algorithme 18 : constructor

Entrées : Vector2D $vector$

1 $super(vector.getX(), vector.getY())$;
2 $this.setZ(0)$;

Algorithme 19 : constructor

Entrées : Vector3D $vector$

1 $super(vector.getX(), vector.getY())$;
2 $this.setZ(vector.getZ())$;