



Groupe 8  
Erwan UMLIL  
Adrien BURQ

Juin 2021

---

# PROJET INF443

## CRÉATION D'UNE SCÈNE ANIMÉE AUTOUR DU NIL

---

### Sommaire

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Démarche</b>	<b>2</b>
<b>3</b>	<b>Modèles géométriques</b>	<b>2</b>
3.1	Principe . . . . .	2
3.2	Items . . . . .	2
<b>4</b>	<b>Textures et plaquage d'environnement</b>	<b>4</b>
<b>5</b>	<b>Animation</b>	<b>4</b>
<b>6</b>	<b>Conclusion</b>	<b>5</b>

### 1 Introduction

Dans le cadre du projet du cours *INF443 - Informatique Graphique 3D*, nous avons choisi de créer une scène animée inspirée des rives du Nil, en Egypte. Une telle scène présente l'avantage de contenir divers éléments modélisables et de natures variées (fleuve, palmiers, pyramides, faune...). Afin de ne pas partir de notre seule imagination, nous avons décidé de nous appuyer sur une image déjà existante (Figure 1), de laquelle nous nous autorisons à dévier au cours de la réalisation du projet.



Figure 1: Image d'inspiration

Ce rapport présente succinctement la démarche adoptée et les modèles utilisés.

## 2 Démarche

Pour parvenir à notre objectif, nous avons choisi une architecture de fichiers par *items* : chaque objet ou groupe d'objet (par exemple la végétation) de la scène est codé sur un fichier du nom de cet objet ou groupe. Nous nous sommes initialement séparé le travail : Adrien a travaillé sur le terrain, le fleuve et le ciel, tandis qu'Erwan a développé les différents items.

## 3 Modèles géométriques

### 3.1 Principe

Pour assurer un confort d'utilisation des fonctions de modélisation, les différents items suivent tous le même format. Ils comportent :

- une fonction *create\_item(parameters)* créant l'objet *mesh* à partir d'une liste de paramètres définissant entièrement l'item;
- une fonction *initialize\_item(mesh\_drawable &item, float size)* qui sera appelée dans le *main* et qui affectera à l'objet *mesh\_drawable* du *main* le *mesh\_drawable* construit à partir de paramètres par défaut et multipliés selon une échelle *size*.

### 3.2 Items

**Terrain et Fleuve** Le terrain est divisible en plusieurs parties :

- l'île (en bas sur la Figure 1)
- la rive gauche va jusqu'au fond de la scène, comme sur la Figure 1, où elle comporte des dunes de sable
- la rive droite
- le fleuve entre ces différentes parcelles surélevées

Chacune de ces portions est représentée par un objet *mesh\_drawable*, ce qui permet de leur appliquer une texture différente. La modélisation de ces objets se fait par un processus de **surface implicite 2D** : partant des équations des formes de rives souhaitées, nous avons défini des fonctions, telles que par exemple *is\_water(position)*, permettant d'en dessiner les contours par balayage.

**Ciel** Pour le ciel, nous avons générée une *skybox* texturée avec une image de ciel étoilé.

**Pyramide** La construction d'une pyramide se fait par **modélisation procédurale** classique, en définissant les coordonnées de chaque point (ainsi que la fonction de plaquage).

Les positions des pyramides sont générées de manière déterministe par le terrain.

**Colonne** La modélisation d'une colonne a été obtenue par un processus plus élaboré de **règle de réécriture**. Partant d'un cylindre de rayon donné, on impose à chaque étape que chaque cylindre présent se divise en sept cylindres plus petits équitablement répartis.

La colonne est ensuite complétée par deux cylindres à rayon plus élevé en haut et en bas.

Les positions des colonnes sont générées de manière déterministe par le terrain.



Figure 2: Colonnes

**Obélisque** Pour l'obélisque, nous employons un processus de **modélisation procédurale** similaire à celui de la pyramide, en ajoutant quatre points formant une base carrée plus grande que celle de la pyramide en-dessous.

La position de l'obélisque est générée de manière déterministe par le terrain.

**Barque** La barque est également définie par *modélisation procédurale*. Elle est définie par l'intersection de deux cercles et une largeur maximale définissant le positionnement de ces cercles, ainsi qu'une hauteur de barque. On y ajoute un fond afin de la rendre plus réaliste.

Le fond est également une astuce afin de simplifier l'animation : l'eau peut désormais pénétrer légèrement dans la barque sans que cela se voie, ne nécessitant pas de gestion des collisions.

**Palmier** Le palmier est représenté par un objet *hierarchy\_mesh\_drawable* contenant deux objets *mesh\_drawable* : le tronc, simplement modélisé par la primitive cylindrique de la bibliothèque *vcl*, et les feuilles, générées par **lancer de particules**.

En effet, la "colonne vertébrale" des feuilles suit la trajectoire d'une particule dont les caractéristiques et la vitesse initiale (ainsi qu'un temps maximal  $t_{max}$ ) sont donnés en entrée. Ensuite, la feuille se voit attribuer une largeur selon un paramètre donné.

Enfin, la vitesse initiale verticale  $v_{0,z}$  se voit attribuer une modification aléatoire afin d'ajouter du réalisme, et chaque palmier est tourné d'une rotation aléatoire pour que tous semblent différents pour l'oeil non averti.

Les positions des palmiers sont générées de manière **aléatoire** par le terrain.



Figure 3: Palmiers

**Fougère** La fougère est constituée d'un objet *mesh\_drawable* créé par recollement de deux objets *mesh*.

Le premier, le tronc, est généré par la même fonction que celle qui génère les colonnes par **règle de réécriture**. Cette fonction se trouve d'ailleurs dans *vegetation.cpp* car elle a d'abord été conçue pour les fougères.

Le deuxième est composé des différentes feuilles, elles-mêmes générées par **règle de réécriture** afin d'obtenir un maillage assez fin, caractéristique des fougères. Un niveau de détail de 2 donne déjà un visuel satisfaisant, bien qu'il ralentisse déjà l'exécution du programme. L'orientation des feuilles plus ou moins "vers le haut" est **aléatoire**.

Les positions des fougères sont générées de manière déterministe par le terrain.



Figure 4: Fougère

**Oiseau** L'oiseau est représenté de manière **procédurale** par un objet *hierarchy\_mesh\_drawable*, contenant une sphère pour la tête, un ellipsoïde pour le corps, deux sphères pour les yeux, un cône pour le bec, et deux quadrangles modifiés pour les ailes.

## 4 Textures et plaquage d'environnement

Le texturage de la plupart des objets a été réalisé par **plaquage d'une image 2D** via une fonction de plaquage.

Pour rendre l'eau du fleuve plus réaliste, lui permettant de refléter le ciel (à la manière de l'image de la Figure 1), nous avons choisi d'implémenter du **plaquage d'environnement**.

## 5 Animation

**Fleuve** Le fleuve est animé de manière descriptive : les vaguelettes sont générées par un **bruit de Perlin** qui donne cette impression de mouvement caractéristique de l'eau. Cette méthode donne des résultats très satisfaisants, couplée avec le plaquage d'environnement, puisque les reflets sont alors très réalistes.

**Barque libre** La barque libre est animée de manière descriptive par **interpolation par une spline cardinale** de points de contrôle de la trajectoire. Ainsi, elle suit simplement une trajectoire donnée en restant à la surface de l'eau.

**Barque attachée** La barque attachée est une barque reliée à la berge droite par une corde (suite de segments). L'animation de ce système est en deux parties.

Premièrement, la barque est animée de manière descriptive, suivant une **trajectoire explicite** calculée à partir d'un sinus, à laquelle on rajoute des éléments aléatoires à plusieurs niveaux (dans le sinus et en dehors du sinus) afin de créer l'illusion légère (et à peine perceptible) d'un mouvement brownien.

Deuxièmement, la barque est reliée à la berge par une corde consistant en une série de masses ponctuelles reliées par des ressorts. On utilise ici une **simulation par modèle physique** afin d'obtenir un comportement de la corde réaliste. En outre, cette corde reste à la surface de l'eau (elle flotte).

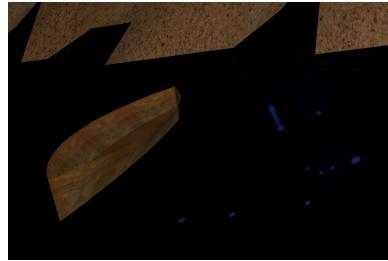


Figure 5: Barque attachée

**Oiseaux** Enfin, les oiseaux sont un système animé assez complexe puisqu'il comporte deux niveaux d'animation.

Premièrement, l'effet de "troupeau" est obtenu à partir d'un **système de particule** avec guide. Nous avons défini un oiseau "leader", non affiché finalement, qui attire les autres oiseaux. Ces autres oiseaux, appelés "followers", sont également soumis à des forces d'attraction à grande distance et de répulsion à faible distance entre eux. On y ajoute une force de frottement visant à aligner les vitesses. Ces forces et leur importance relative sont pilotées par des coefficients données en entrée.

L'oiseau leader suit une trajectoire bouclée définie de manière descriptive par **interpolation par spline cardinale** de points de contrôle. Nous avons fait le choix délibéré de ne pas interpoler les orientations afin de ne pas passer trop de temps sur les oiseaux. Par conséquent, nous orientons les oiseaux de manière moins réaliste selon la direction dans laquelle ils vont à chaque instant, créant une orientation discontinue dans le temps.

Une fois les positions et orientations des oiseaux obtenues par simulation, nous affichons  $n$  fois le même *mesh\_drawable* translaté, où  $n$  représente le nombre d'oiseaux. Nous obtenons ainsi un "troupeau" d'oiseaux cohérents.

## 6 Conclusion

Pour conclure, nous avons profité de ce projet pour appliquer en pratique différentes techniques vues en cours. Nous avons cherché à aller plus loin qu'en travaux pratiques (lancer de particules, règles de réécriture, système de particules), tout en nous restreignant pour ne pas nous égarer et pour être certains de rendre un projet complet et cohérent.



Figure 6: Scène complète