PROJET INF443

Création d'une scène polaire 3D

Mai 2021

Xiyao LI
Vu Hoang Anh PHAM

1 Introduction

Nous avons choisi d'animer une scène 3D d'une région polaire en utilisant principalement les modèles d'animation vu en TP. Sous la neige, une petite île flotte sur la mer, couverte d'igloos, de pingouins et de forêt boréale. Des morceaux de banquise de forme aléatoire se trouvent autour de l'île et il y a un iceberg au loin.

Le rendu final de notre scène est ci-dessous. Dans la suite du rapport, nous présentons succinctement les modèles géométriques et l'animation que nous avons mis en place.

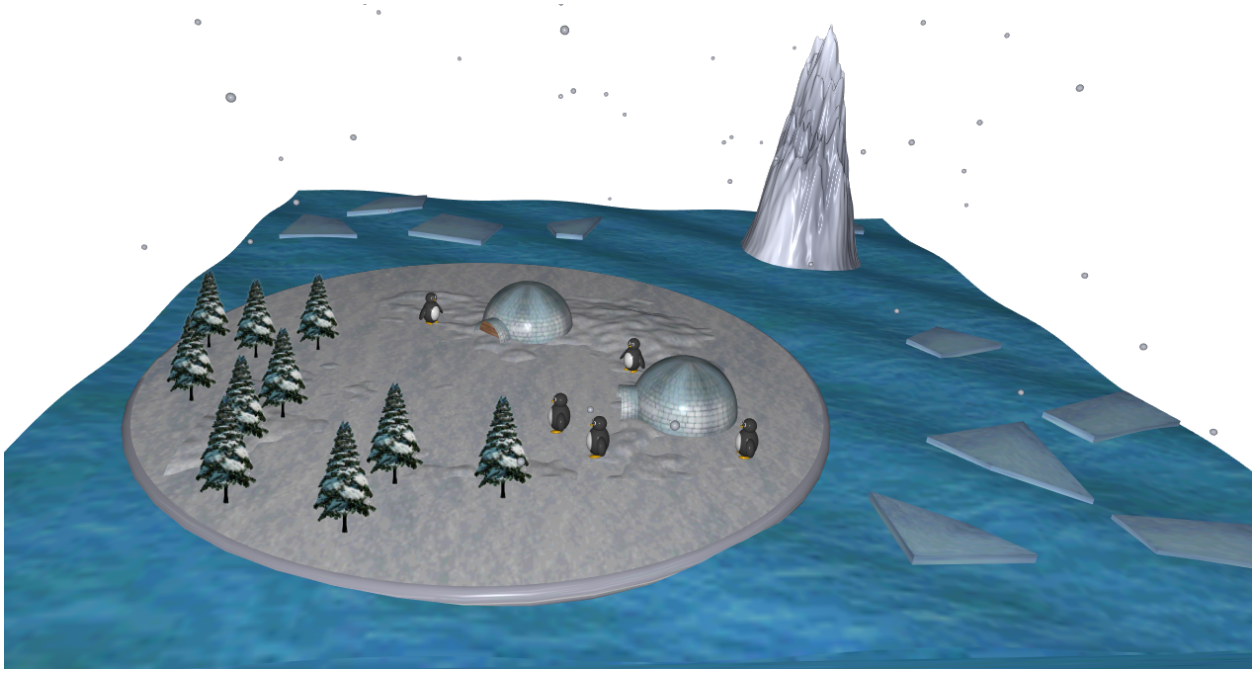

Figure 1 – Rendu final du scène 3D

2 Méthode d'animation

2.1 Modèles vu en cours

— **Igloo** : Nous avons combiné une sphère et un cylindre qui passe par le centre de sphère en utilisant les fonctions de la bibliothèque *VCL* générant des maillages primitifs. Après l'avoir texturé par une image de briquette glacée à motif répétitif, nous avons fait émerger la partie supérieure de cette combinaison pour simuler un igloo.

— **Pingouin** : Nous avons créé des pingouins par une animation hiérarchique qui relie des géométries simples par une relation hiérarchique. Nous n'avons pas rencontré de difficulté dans cette partie du code, mais il a fallu beaucoup de temps pour ajuster la taille, l'orientation et l'amplitude de swing de chaque partie.

— **Pin** : Afin d'ajouter de la végétation sur l'île, nous avons choisi les billboards consistant à mettre une image de pin sur un quadrangle transparent qui fait toujours face à la caméra.

— **Iceberg** : Nous avons commencé par créer une grille avec la fonction *mesh_primitive_grid()*. En ajoutant le bruit de Perlin, nous avons augmenté la hauteur du terrain et rajouté des irrégularités comme les sommets de la montagne.

2.2 Effet de la mer

En commençant avec une grande grille qui est créée par la fonction *mesh_primitive_grid()*, nous avons eu plusieurs options pour simuler les vagues de la mer. Une façon de procéder aurait été d'utiliser la transformation de Fourier rapide, ou encore d'utiliser le bruit de Perlin. Nous avons finalement choisi d'utiliser une fonction périodique du temps. Pour chaque point $P(u, v)$ de la grille, nous mettons à jour sa hauteur avec une fonction sinus périodique $c_3 * \sin(c_1 u + c_2 v + 2\pi t)$ (dont c_1, c_2 pour contrôler la direction des vagues et c_3 pour contrôler l'amplitude) pour animer le mouvement de la mer.

2.3 Création des morceaux de banquise

Nous nous sommes inspiré de la fonction *mesh_primitive_cube()* dans la bibliothèque *VCL* pour créer des quadrilatères de glace. Nous définissons au préalable l'épaisseur h et un rayon maximum r pour le tirage aléatoire. Dans chaque quart de plan délimité par les axes x et y, nous faisons deux tirages aléatoires entre 0 et 1 pour obtenir les coordonnées x et y d'un sommet. Nous copions ensuite leurs coordonnées en x et y et ajoutons une épaisseur h en coordonnée z. Puis, en utilisant la fonction *mesh_primitive_quadrangle()* pour relier les points et générer un maillage fermé, nous créons une banquise de forme aléatoire avec une épaisseur h. Cela se traduit par le pseudo-code suivant :

Algorithm 1: Création des morceaux de banquise

h := height
r := rayon maximal
for *chaque quart de plan* **do**
  $x := r *$ tirage aléatoire entre (0,1)
  $y := r *$ tirage aléatoire entre (0,1)
  $p[i]_{down} := (x, y, 0)$
  $p[i]_{up} := (x, y, h)$
end
Relier les points adjacents pour former les six facettes
return le maillage final

2.4 Trajectoire du pingouin

Le mouvement du pingouin est simulé par animation descriptive. Étant donné les points clés de sa position, une trajectoire lisse est obtenue par l'interpolation spline cardinale de ces points.

Cependant, quand le pingouin arrive à la fin d'un segment de la trajectoire, le changement d'orientation reste encore brusque. Pour éviter ce problème, nous recalculons l'orientation du pingouin à chaque instant. Plus précisément, nous calculons la position du pingouin p_0 avec la fonction d'interpolation à l'instant t et sa position p_1 à l'instant $t + dt$. L'axe de rotation du pingouin à l'instant t est donc donné par $p_1 - p_0$. Cette méthode permet d'animer un mouvement lisse et plus réaliste sans produire de changement brusque.

2.5 Modélisation de la neige

Pour la neige, nous avons utilisé l'animation par modèles physiques : nous modélisons les flocons de neige par des particules qui ont une hauteur initiale très haute et une vitesse initiale nulle. Les flocons tombent alors sous l'effet de la gravité.

Algorithm 2: Modélisation de la neige

$r := 0.1$
$dt := 0.01$
Initialisation des particules par $Z := 20.0$ et $V := 0$
$l =$ demi longueur du plan
for *Chaque particules* **do**
  $V := V + dt * F/(c * m)(cestconstante)$
  $X :=$ tirage aléatoire entre $[-l, l]$
  $Y :=$ tirage aléatoire entre $[-l, l]$
  $Z := Z + V * dt$
end
Retirer les particules sous le plan
Afficher la scène;

3 Aspects à améliorer

Malgré l'ajout de vagues en mouvement sur la mer, nous n'avons pas pu faire bouger la banquise avec les vagues. De plus, les morceaux de banquise ont toujours quatre côtés, nous aurions pu écrire une fonction plus sophistiquée pour générer un nombre aléatoire de côté.

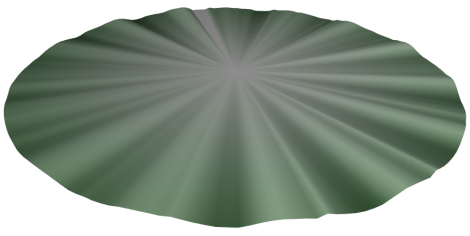

Figure 2 – Relief irréaliste à cause d'un nombre insuffisant de triangle

L'île flottante a été créée par un maillage primitif de disque : on échantillonne des points sur la circonférence du disque et on les relie tous au centre pour générer un disque constitué des triangles. À cause d'un nombre insuffisant de triangles, quand nous rajoutons le bruit de Perlin, le relief est seulement sur les côtés des triangles. Pour éviter ce problème, nous aurions dû échantillonner le rayon du disque pour créer des cercles concentriques, puis les remplir par des triangles. Finalement nous avons préféré superposer une grille à l'emplacement de l'île pour rajouter des reliefs.