Sacha BRAUN, Gaspard PASQUERON DE FOMMERVAULT

I. Idée originale

Nous avons choisi, pour notre projet, de présenter une scène montrant le mythique Orient Express, évoluant dans un paysage montagneux, avec de la neige.

II. Paysage

Nous avons commencé la modélisation de la scène par le paysage montagneux. Pour cela, nous avons entouré notre scène avec une skybox présentant un arrière plan montagneux. Les skybox disponibles en libre accès sur internet étant de qualité médiocre, nous avons décidé de ne considérer que les plans verticaux (4 faces), pour lesquels nous avons répliqué une même image. La contrepartie réside donc dans des « anomalies » à la jonction de 2 faces, présentant un paysage parfaitement symétrique.

Pour le terrain en lui-même, nous avons créé une grille de taille 400 par 400, à laquelle nous avons ensuite appliqué un bruit de Perlin avec des paramètres trouvés empiriquement pour répliquer le plus fidèlement possible le paysage montagneux. Nous avons ajouté par dessus une texture 2D de montagne (rochers et neige) à partir d'une image trouvée sur internet. La qualité de l'image, bien que critère primordial de notre choix de texture, ne permet cependant pas de rendre le terrain suffisamment réaliste lorsque l'on zoome trop.

Nous avons ensuite choisi de complexifier la scène en ajoutant un lac de glace, et des détails comme des igloos, un pêcheur et des mouches.
Le lac consiste en un plan z = constante, sur lequel on a ajouté une texture 2D de glace, et qui coupe le bruit de perlin des montagnes.
Pour les igloos, nous avons créé un mesh spécifique (sphère déformée), avec une texture adaptée, et nous les avons distribués de façon aléatoire sur le lac (sans détection des potentiels contacts ou pénétrations entre deux igloos). Les formes des igloos sont relativement aléatoires dans la mesure ou le rayon et la deformation de la sphere, ainsi que la hauteur et la taille de la porte sont générés aléatoirement.

Concernant le pêcheur, nous avons d'abord créé le trou dans le lac, modélisé par un disque troué ou le centre du disque est légèrement surélevé. Ainsi cela donne l'illusion d'un trou. Le pêcheur se trouve juste au bord du trou, et son bras est animé de façon descriptive. Le fil de pêche tenu dans sa main est animée par mouvement générateur : nous avons choisi un modèle de fil extensible (système masse-ressorts), dont nous contraignons le mouvement du point en haut, le faisant correspondre à celui de la main du pêcheur. Le caractère extensible du fil ne nuit pas au réalisme étant donné que le spectateur ne voir pas l'extrémité basse du fil.
Les mouches autour du pêcheur sont animées par un mouvement de boyds : une mouche « leader » suit une trajectoire circulaire autour du trou de pêche, et les autres mouches tendent à s'en rapprocher ou à s'en éloigner en fonction de la distance qui les en séparent.

Pour la neige qui tombe, nous avions préalablement opté pour des flocons modélisés par des sphères blanches, auxquelles on attribue une masse, soumises aux forces de gravité et à la force de frottement visqueux de l'air. Ce choix posait 2 problèmes majeurs : premièrement, il nécessitait la gestion d'un très grand nombre de sphères en même temps, ce qui affectait le rendu visuel de la scène, le rendant saccadé; deuxièmement, on perdait en réalisme avec l'assimilation des flocons à des simples sphères blanches, surtout pour celles qui tombaient près de la caméra. Nous avons donc opté pour des billboards pour représentas les flocons, ce qui règle nos deux problèmes préalablement cités. Nous avons animés ces billboards par un mouvement uniforme (vitesse constante) depuis une position prise aléatoire à une altitude fixée. En effet, il nous est

apparu que l'accélération des flocons au cours de leur chute n'était pas un détail important pour le réalisme de la scène. En dessous d'une certaine altitude (concrètement une fois que le flocon a dépassé ne niveau zéro de la scène), on le retire de la liste des flocons « actifs » pour éviter que l'algorithme ne diverge, devant gérer une quantité trop grande de flocons.

III. Train

Pour le train, nous avons modélisé une locomotive et des wagons (nous en avons affichés 6 dans notre projet mais on peut facilement en affiché plus). Pour animer le train, nous avons placé des points de contrôle de la trajectoire sur notre terrain. Nous avons ensuite interpolé ces points avec une courbe spline pour décrire la trajectoire. Le terrain étant montagneux, la trajectoire est en 3D dans l'espace et ne peut pas être restreinte à un plan ce qui a crée des problèmes pour les rotations du train.

Le train est animé de manière générative. On lui indique une acceleration et une vitesse initiale et le train avance ensuite tout seul. La vitesse peut alors augmenter, ou même diminuer en fonction de la pente. L'acceleration et la vitesse du train dépend donc du terrain et la vitesse précédente.

L'idée du code est la suivante :

En fonction des données précédentes et du terrain, on met à jour l'accélération, puis la vitesse, pour calculer la distance parcourue. On calcule la position grâce à l'interpolation à cette distance et on met à jour la position de la locomotive. Grâce à la dérivée de la courbe d'interpolation on peut mettre à jour la rotation selon l'axe y. Nous avons eu des soucis liés à des changements brusques de signe de cet angle, nous avons donc codé un système empêchant les changements brusques d'angle. Cette méthode fonctionne beaucoup moins bien pour la rotation qui permet au train de suivre la courbe. Pour celle-ci, nous avons utilisé la différence de hauteur entre deux positions successives.

La distance utilisée est la distance théorique réellement parcourue par le train et non pas le paramètre « t » du code interpolation initialement vu en classe en une distance réelle parcourue. L'interêt est triple :

- Avoir une vitesse cohérente : même si deux point de contrôles sont très rapprochés ou très éloignés le train va à la même vitesse sur les tronçons. La vitesse du train ne dépend pas de l'écart entre les points de contrôle.
- Coder le mouvement des wagons. En effet, pour garder une cohérence dans le mouvement du train nous avons imposé aux wagons de toujours être espacés d'une même distance réelle. Ainsi, le mouvement les wagons sont toujours à la même distance du train.
- Faire tourner les roues avec une certaine cohérence. La vitesse de rotation des roues dépend de la vitesse réelle.

La vitesse de rotation des roues est particulièrement importante dans notre projet dans la mesure où nous avons modélisé un piston pour la locomotive. Toutes les pièces de la roues du piston ont été modélisées en code « dur » et sont disponibles dans le fichier « model_texture ». Le piston entraine les roues et les roues tournent en fonction du piston… du moins théoriquement, car en réalité le mouvement du piston est un mouvement descriptif. Nous avons mis en équation les positions du piston en fonction de la rotation de la roue.

Enfin, pour avancer, le train doit avoir des rails ! Pour faire cela nous avons crée un mesh, là aussi avec un code « dur » en openGL. On donne la trajectoire du train et on la parcourt pas à pas. À chaque pas on ajoute quatre point pour former un carré et obtenir à la fin du processus un rectangle géant très fin. On duplique le mesh et on ajoute enfin une texture acier dessus. Ensuite on procède de manière similaire pour ajouter pas à pas des planches de bois entre les rails et des pilotis qui portent la structure.