

Rapport du Projet Pratique INF443

Capucine Leroux et Thibaud Verny

Sujet : Forêt entourant un lac, habitée par divers animaux

Plan du rapport :

- I - Démarche du groupe
- II - Conception du terrain et de la végétation
- III - Conception et animation des animaux
- IV - Intégration des animaux au terrain

I – Démarche du groupe

La principale difficulté à laquelle nous avons dû faire face est le travail à distance, imposé par la crise sanitaire et par le confinement. Il a donc rapidement été décidé de séparer le travail de manière à être le plus indépendants possible, et de repousser au dernier moment la mise en commun de nos travaux. Nous avons donc séparé, comme l'indique le plan, la conception de notre scène en deux points : la conception du terrain et de la végétation, et la conception des animaux. Tout en faisant des points réguliers sur les avancées de chacun, nous pouvions ainsi travailler indépendamment l'un de l'autre, et donc diminuer l'impact de la distance sur notre projet. Comme conseillé sur Moodle, nous avons partagé notre code ensemble via Github. Ceci nous a permis de garder un œil sur le travail de l'autre et de partager notre expérience de certains bugs récurrents.

Ce partage de code via Github avait aussi pour objectif de pouvoir accéder à tout moment au code de l'autre, pour faciliter la mise en commun de nos travaux. En effet, notamment pour le mouvement des animaux, il était intéressant de savoir comment le terrain était construit pour que l'adaptation de ce mouvement au terrain soit la plus simple possible.

II – Conception du terrain et de la végétation

La première étape de cette partie a été de modéliser la surface. Nous avons repris la forme globale du terrain du TD mais en adaptant les points et les valeurs de hauteur pour obtenir la forme voulue. Le centre de ce terrain est constitué d'un trou, destiné à accueillir le lac. La première difficulté rencontrée a été de faire disparaître les discontinuités du terrain autour du lac, dues à l'affaissement brutal. L'équation de la hauteur en fonction des deux autres coordonnées a donc été adaptée en fonction de la distance au centre du lac. Nous avons ensuite rendu cette surface un peu plus réaliste en y ajoutant une composante de bruits de Perlin, puis en y plaquant une texture d'herbe.

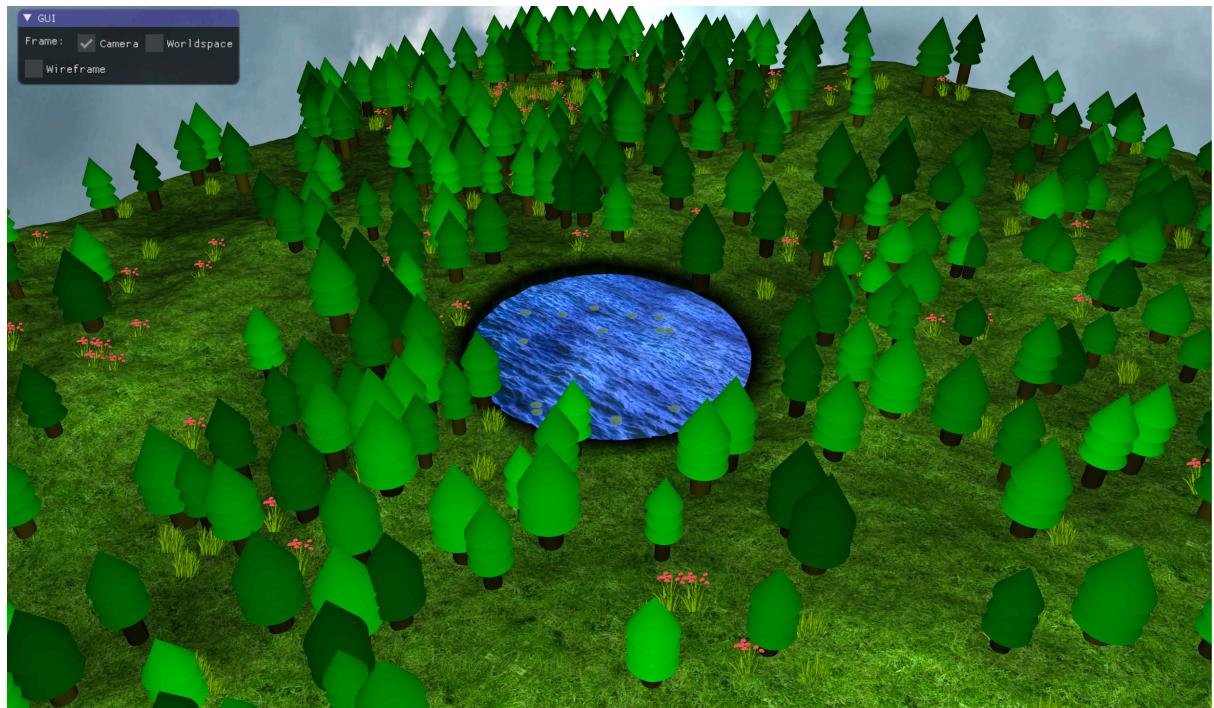
Il a ensuite fallu concevoir le lac. L'idée était de concevoir un terrain qui, au niveau de l'affaissement de terrain, avait une surface plane de lac, et qui s'affaissait sous la terre aux autres endroits. Nous y avons ensuite ajouté un relief sinusoïdal de très faible amplitude, couplé à des bruits de Perlin pour augmenter le réalisme de cette surface, comme si elle était perturbée par du vent. Nous y avons ensuite plaqué une texture d'eau. S'en est suivi une

étape de tâtonnement sur les constantes du terrain, du lac et des bruits, pour obtenir la meilleure cohérence possible, sans discontinuités et sans que le lac ne déborde.

Une fois le terrain conçu, il a fallu le peupler de végétation. Nous avons repris l'architecture des arbres du TD pour peupler notre forêt. Notre première idée était de répartir notre terrain en des zones différentes, entre lesquelles la densité en arbres allait varier. La fonction `update_positions` a donc été séparée en huit sous fonctions, peuplant chacune une partie de notre terrain. Toujours dans l'objectif de donner un aspect plus réaliste à notre scène, nous avons décidé de randomiser la couleur et la taille de nos arbres, ce qui donne à notre forêt un aspect moins monotone. Nous avons donc introduit une fonction `create_tree`, qui ajoute une variable randomisée au rayon du tronc et du feuillage, et aux valeurs RGB de la couleur. Nous avons ensuite peuplé la forêt d'un nombre limité de touffes d'herbe et de fleurs sous forme de billboards, et nous en avons rajouté un nombre important au niveau d'une clairière circulaire conçue dans l'une des zones de notre forêt. Ces billboards font face à la caméra en permanence, ce qui permet d'éviter de remarquer que ce ne sont que des images planes. Afin de diminuer la monotonie du lac, nous avons aussi ajouté des nénuphars à sa surface, mais à orientation constante cette fois, de manière à ce qu'ils soient toujours plaqués sur l'eau.

Enfin, nous avons rajouté une skybox autour de notre terrain, pour donner l'impression que la scène s'inscrit dans un monde plus étendu. Nous avons pour ce fait récupéré une image de ciel sous la forme d'un patron de cube, et créé un `mesh_drawable` à partir de celle-ci. Malgré nos efforts de réalisme, certaine délimitations de ce cube restent visibles sur le rendu final.

Voici un aperçu du rendu final du terrain seul :



III – Conception et animation des animaux

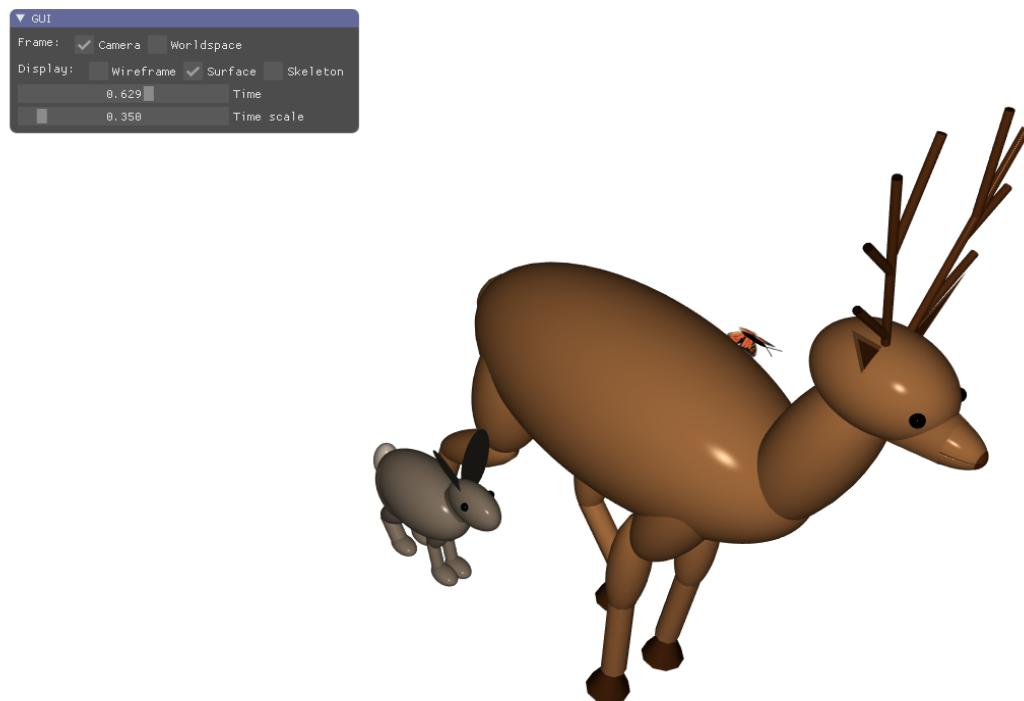
Nous avons conçu trois animaux différents : un cerf, un lapin et un papillon.

Pour concevoir les corps des trois animaux, il a fallu implémenter des fonctions auxiliaires pour les différentes formes de bases dont nous avions besoin pour chacun des membres. Notamment, la sphère pour les yeux, l'ellipsoïde par exemple pour les corps, tête, cou, pattes, le cylindre pour certaines parties des pattes, antennes et moustaches, et des formes légèrement plus complexes comme des cônes ou ellipsoïdes partiels. Pour certaines de ces fonctions, nous avons utilisé les primitive mesh déjà implémentés.

Une fois ces formes de bases implémentées, chacun des corps a été construit grâce à une hiérarchie chaînée, en plaçant à taton au visuel les différents membres avec des tailles, formes et couleurs adaptées. Pour l'instant, les animaux étaient à l'arrêt dans leur posture de base.

Pour le papillon, seul le corps a été conçu de cette manière. Pour les ailes, nous avons détourné une image d'aile de papillon que nous avons collée en texture sur un rectangle transparent. Nous avons créé plusieurs versions de cette image en recoloriant les ailes en orange, bleu, vert ou rose pour avoir plusieurs couleurs de papillons pour la suite.

Voici un aperçu de ces animaux à l'arrêt :



Ensuite, pour animer ces trois animaux sur place, il a fallu tester et calculer les vitesses et amplitudes de variation d'angles de chaque membre l'un par rapport à l'autre, ainsi que le décalage en temps de chaque mouvement pour donner une impression réelle de marche. Pour cela, nous avons fait osciller chaque membre en mouvement autour de sa position de base en testant amplitude, vitesse et décalage en temps par rapport à l'ensemble. Pour faciliter la lisibilité, nous avons implémenté des fonctions auxiliaires qui calculent la matrice de rotation autour des axes x, y et z. Pour le papillon, le mouvement est simple, il suffit de faire battre les ailes autour du corps. Pour le cerf et le lapin, il faut faire attention à garder le pied toujours incliné face au sol pour garder l'impression que l'animal marche. Nous avons également ajouté une légère oscillation du cou et de la tête pour le réalisme. Un problème que nous n'avons pas pu totalement géré est le fait que les pieds des animaux ne restent pas toujours à une hauteur parfaitement constante. Il est compliqué de contrôler la hauteur du pied directement car toute la chaîne de la hiérarchie est rattachée au corps et non au pied. Sachant que les animaux seront dans de l'herbe, l'impression de s'enfoncer légèrement dans le sol ne sera pas très gênante donc nous avons accepté quelques oscillations de niveau.

Une fois l'impression de mouvement à l'arrêt créée, il a fallu gérer la trajectoire des animaux, les faire avancer. Pour le lapin et le papillon, nous voulions une trajectoire aléatoire dans un périmètre fixé. Pour le cerf, nous voulions qu'il enchaîne plusieurs actions : qu'il marche entre les arbres vers le lac, puis qu'il détecte l'approche du lac, s'arrête, et se baisse pour boire, puis remonte et ne bouge plus.

La difficulté majeure que nous avons dû surmonter pour cette étape est que la fonction d'affichage fonctionne de manière cyclique. A chaque étape, le programme relit la fonction d'affichage dans sa totalité, avec possibilité de modifier le mouvement au cours du temps à l'aide d'un timer qui revient à 0 également de manière cyclique. Cela ne se prête pas forcément à programmer un enchaînement de mouvements qui restent aléatoires et autonomes. En effet, la trajectoire du cerf n'est pas décidée à l'avance, il devra détecter seul s'il approche du lac ou s'il doit continuer d'avancer, ou s'il doit éviter un arbre. On ne peut donc pas créer un tableau de keyframes contenant à l'avance tous les points de contrôle de la trajectoire et instants correspondants. La solution que nous avons choisie a été de créer deux variables de temps : le timer qui tourne en boucle entre 0 et 1, et une variable t_reel qui s'update pour faire défiler le temps jusqu'à ce qu'on ferme la fenêtre d'animation. Chaque fois que le timer revient à 0, t_reel passe à l'unité de temps supérieure. De cette manière, on peut calculer un mouvement selon le temps, qui ne revient pas à 0 de manière cyclique, mais qui continue de manière linéaire jusqu'à la fin de l'animation.

Pour le mouvement du lapin et du papillon, nous avons associé à chaque animal un tableau de keyframes contenant 4 points de contrôle de trajectoire, correspondant à t0, t0+1, t0+2, t0+3. La position de l'animal entre t0+1 et t0+2 est une interpolation des deux, puis à t0+2, le tableau de keyframes s'update, crée un nouveau point de contrôle aléatoire et fait avancer les temps correspondants. Nous avons également implémenté une fonction calculant le vecteur dérivé associé à la position de l'animal et qui calcule la matrice de rotation qui permet d'aligner le corps avec le vecteur dérivé.

Pour le cerf, une difficulté de plus se pose. Le cerf avance selon le même principe, jusqu'à ce qu'il détecte qu'il est proche du lac. A ce moment-là, le cerf doit passer à un mouvement différent. Nous avons donc créé plusieurs fonctions auxiliaires, chacune dédiée à un mouvement : la marche, l'arrêt pour se remettre droit, baisser la tête, boire, relever la tête. Pour savoir quelle fonction doit être lue, nous avons créé des variables booléennes qui

retiennent dans quelle fonction on se situe. Si un critère est détecté (à savoir : le cerf est proche du lac, ou le cerf est bien revenu en position de base, ou le cerf a fini de baisser la tête), les valeurs des booléens sont mises à jour. Il faut donc passer et rendre les booléens aux fonctions, ce qui nécessite parfois de rendre plusieurs arguments, il faut alors créer au préalable la structure rendue par la fonction. Puis, pour chacun des mouvements, il a fallu effectuer plusieurs calculs pour faire en sorte que tous les membres s'arrêtent au même moment au bon endroit avec le bon mouvement. Enfin, pour s'assurer de la continuité du mouvement, par exemple que le mouvement pour baisser la tête finisse bien à la position où commence le mouvement pour relever la tête, il a fallu jouer sur le timer pour l'arrêter et le reset aux bons moments. Il y a donc un timer pour chaque animal.

Il faut être également vigilant pour tous les tests d'arrêt car on ne peut pas tester `angle_reel == angle_de_reference`, il faut tester qu'ils soient suffisamment proches. Mais ce seuil doit être ajusté pour éviter qu'il soit trop grand ou trop petit. Si le seuil est trop petit, la condition risque de ne jamais être satisfaite, si le seuil est trop grand, le mouvement ne se finit pas correctement.

Une fois les trajectoires gérées, il a fallu adapter les fonctions pour pouvoir ajouter plusieurs papillons et plusieurs lapins. Nous avons ajouté des fonctions auxiliaires qui créent un tableau contenant la trajectoire aléatoire de chaque papillon et de chaque lapin, et qui crée des nouveaux points aléatoires au cours du temps en évitant les autres papillons ou les autres lapins. Nous avons également randomisé la couleur des papillons entre les 4 couleurs possibles.

IV – Intégration des animaux au terrain

A ce stade, nous avons mis en commun les deux codes. Pour cela, nous avons privilégié l'utilisation de fonctions auxiliaires pour alléger les fonctions de setup des data et d'affichage. Nous avons regroupé au maximum ces fonctions auxiliaires dans des fichiers séparés pour gagner en lisibilité.

Pour mettre les animaux dans le décor, nous avons adapté l'échelle des animaux au décor existant. Puis nous avons adapté les fonctions qui calculent la trajectoire aléatoire en imposant plus de contraintes. Notamment, les lapins et le cerf doivent vérifier dans la liste de positions d'arbres qu'ils ne rencontrent pas un tronc. Les lapins et les papillons sont restreints à bouger dans un périmètre précis du terrain pour être plus visibles dans la forêt et éviter les zones trop denses d'arbres. Le cerf, lui, est contraint de suivre une diagonale allant d'un coin de la forêt au lac tout en évitant les arbres. Enfin, les points de contrôle de trajectoire du cerf et du lapin sont à la hauteur du terrain.

Nous avons dû également ajuster certains éléments du décor, par exemple remonter la hauteur du lac pour s'assurer que le cerf pouvait atteindre l'eau en se baissant.

Néanmoins, certains problèmes perdurent : du fait que la trajectoire est une interpolation entre deux points de contrôle, il peut arriver que la position d'un lapin ne colle pas parfaitement au sol. Cela implique aussi que le cerf et les lapins peuvent se retrouver proches d'un arbre malgré le fait que les points de contrôle en soient loin.