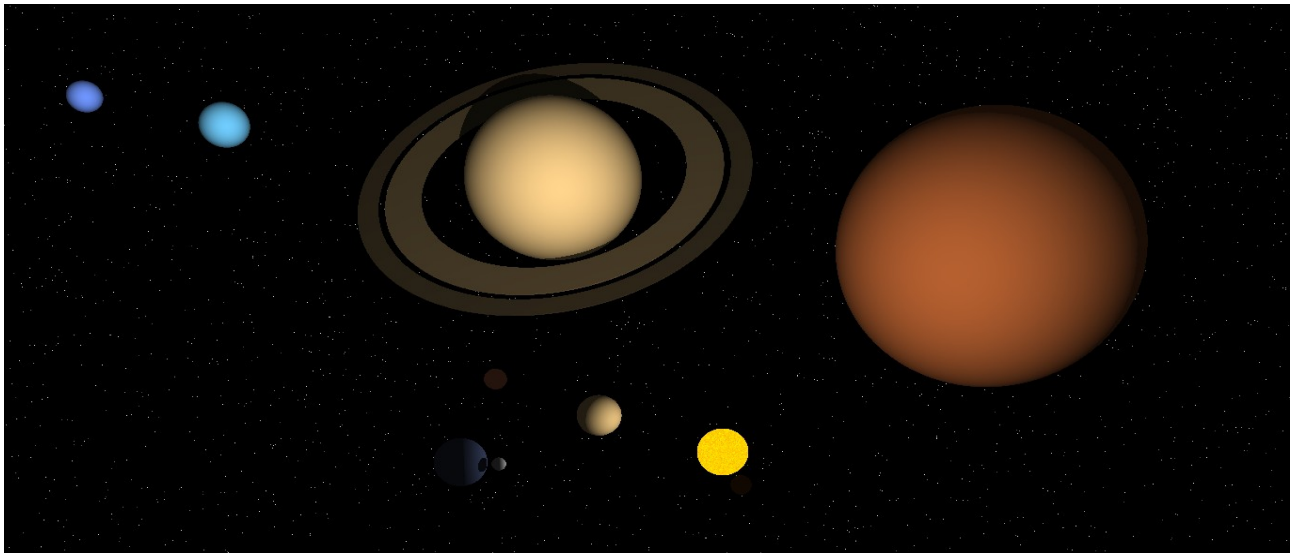
Rapport du projet de cours Web-INFO4 – 3D Dynamique pour le web

Simulation du système solaire



Ce rapport présente le projet réalisé lors des séances du cours de 3D Dynamique pour le web. Le sujet choisi est la simulation simplifiée du système solaire.

Soleil

Le premier astre sur lequel le travail s'est porté est le soleil. Concernant sa forme, il est rendu par une simple sphère de rayon 1 et de centre égal à l'origine du repère de la scène. Etant la source de lumière de celle-ci, sa couleur n'est pas obtenue par le modèle de Phong fourni en cours (il n'est pas irradié par une source de lumière externe). Une fonction spécifique a au contraire été écrite pour sa texture : elle reprend le bruit fractal dont le code a été donné dans ce module. Pour chaque pixel correspondant à cet astre, une valeur du bruit est calculée et permet d'obtenir une couleur comprise entre le jaune et l'orange. Les valeurs utilisées dans cette fonction ont été fixées de façon empirique, pour un rendu satisfaisant.

Principe général de composition de la scène

La seconde étape a consisté en l'ajout de planètes à la scène. Pour des raisons qui tiennent à la progression du projet et à celle de ma compréhension des concepts, c'est une fonction globale de recherche de minimum des valeurs de distances signées aux différents objets qui a dès le début été utilisée pour composer la scène. Ce n'est pas une opération d'union successive qui est effectuée pour chaque objet à ajouter.

Plus tard, une structure de données particulière, nommée Astre, a été créée en remplacement de la structure Surface fournie. Elle contient, en plus des champs de cette dernière, les informations de position du centre de l'astre (donnée à réutiliser dans différentes situations) et de son type (un booléen, égal à true s'il s'agit du soleil, et false pour tout autre astre).

La composition de la scène se fait donc de la façon suivante. Pour chaque astre à représenter, une variable de type Astre est calculée à partir de données géométriques et stockée dans un tableau. Puis ce tableau est fourni à la fonction de calcul de minimum mentionnée précédemment, et celle-ci renvoie l'astre dont la distance signée au point courant est la plus petite. Des fonctions d'union et de soustraction sont tout de même utilisées ponctuellement.

Représentation des planètes

Le principe de composition ayant été posé, huit planètes ont ensuite été représentées : Mercure, Venus, la Terre, Mars, Jupiter, Saturne, Uranus et Neptune. Un seul satellite est affiché : la Lune.

Pour représenter des planètes de façon réaliste, des données réelles ont été utilisées. Elles ont été tirées de la page Wikipédia consacrée au sujet et d'un livre grand public emprunté pour l'occasion. Trois types de données sont nécessaires : le rayon des planètes, le rayon de leur trajectoire circulaire autour du soleil (valeur

moyennée) et la fréquence de révolution correspondante. Ces différentes données sont exprimées relativement à celle de la Terre (donneePlanete = donneeTerre * RapportDonnees). Les valeurs utilisées pour la Terre jouent donc le rôle de facteurs d'échelle et permettent de calibrer l'ensemble du système.

Il faut mentionner qu'il n'est pas possible de représenter correctement les différents types de distances dans le système simulé : les proportions entre les rayons des planètes sont respectées, les proportions entre les distances des planètes au soleil le sont également, mais les proportions entre ces deux types de données ne le sont pas et ne peuvent pas l'être : pour un rayon de la Terre égal à 1, celle-ci devrait être située à environ 25 km du soleil… Cela explique d'ailleurs la taille donnée au soleil.

Mouvement des planètes

Il n'y a pas de simulation physique du mouvement des astres (le problème à n corps n'est pas résolu… mais cela pourrait être une évolution du programme). Un mouvement circulaire est simplement imposé aux planètes, celui-ci étant obtenu par translation circulaire des sphères les représentant. Cette translation est donnée par la formule :
rayonTrajectoire * vec3 (cos(2*PI*frequence*iTime), 0., sin(2*PI*frequence*iTime)

Une autre approche aurait pu être adoptée : réaliser une rotation de l'astre dépendant du temps, puis (dans cet ordre) une translation par ajout d'un vecteur constant.

Couleurs des planètes

Le principe d'obtention de la couleur d'un astre autre que le soleil est le suivant. Une couleur initiale est tout d'abord attribuée à celui-ci lors de la création de la variable de type Astre qui lui correspond. Puis, un astre ayant été atteint par l'algorithme de ray marching, sa couleur initiale est modifiée par la fonction shade réalisant le modèle de Phong.

La formule a cependant été modifiée pour les besoins de la scène. Premièrement, le soleil illuminant les autres astres, le vecteur donnant la direction de la source de lumière (utilisé dans le calcul du produit scalaire des termes spéculaire et diffus) doit être modifié. Ce vecteur doit correspondre au vecteur normalisé partant du point atteint par le tracé de rayon et allant au soleil. Le soleil étant placé à l'origine, ce vecteur correspond directement à celui retourné par l'algorithme de ray marching (ro + d * rd). Ensuite, le terme spéculaire a été supprimé car les reflets sur les sphères représentant les planètes étaient tout à fait incongrus, même avec un exposant faible. Enfin, le terme constant est pris égal à la couleur initiale pondérée par un coefficient réglable (et le terme diffus est pondéré par 1 moins ce coefficient). Ce terme permet de régler la couleur des planètes lorsqu'elles ne sont pas illuminées.

Le soleil est traité à part : si c'est lui qui est atteint par l'algorithme de ray marching, le champ de la structure Astre qui donne le type de l'astre (le booléen « type ») permet de dérouter le rendu de sa couleur vers la fonction qui lui est dédiée, et qui a été décrite dans le premier paragraphe.

Eléments particuliers

Anneaux de Saturne

Ces éléments ont constitué une difficulté non négligeable de la scène. Ils sont finalement créés de la manière suivante. Un anneau est obtenu par soustraction ensembliste d'une sphère (de rayon le rayon minimal de l'anneau) à un cylindre de très faible hauteur (et de rayon le rayon maximum de l'anneau). La fonction de distance signée utilisée pour tracer le cylindre est celle fournie par Inigo Quilez utilisant deux vecteurs : le vecteur a allant de l'origine à la base du cylindre, le vecteur b allant de l'origine à l'autre extrémité du cylindre (le vecteur b-a donne alors la direction de l'axe du cylindre). Pour représenter des anneaux avec une direction constante dans l'espace, il suffit donc de choisir a et b tel que b-a soit constant. Le vecteur a étant paramétré par le temps pour décrire un cercle (la trajectoire de Saturne), b se déduit par b = constante + a, constante étant choisie arbitrairement comme axe des anneaux.

Lune

La Lune est représentée dans la scène comme élément emblématique de notre planète et comme exemple de composition de mouvements. Pour afficher ce satellite, on décrit premièrement son mouvement comme une translation circulaire autour de la Terre, avec la même équation que celle utilisée pour décrire le mouvement d'un astre autour du soleil (positionLuneTerre(t)). Puis, connaissant la position de la Terre par rapport au soleil (positionTerreSoleil(t)), on peut tracer la Lune en utilisant la fonction de distance signée d'une sphère,

avec translation fournie par la somme des deux positions précédentes : sdSphere (p + positionTerreSoleil(t) + positionLuneTerre(t), rayonLune).

Cratères de Mars

Pour illustrer la rotation d'une planète sur son axe, une idée envisagée a consisté à représenter des cratères sur sa surface. Des tests ont été réalisés avec des perturbations de surface en cosinus, mais le résultat n'était pas probant. Une autre façon de procéder a finalement été utilisée : la soustraction (au sens des ensembles) de sphères décalées par rapport à la sphère principale et affleurant celle-ci. Les coordonnées sphériques sont utilisées pour déterminer la position des sphères à soustraire. Des valeurs de rayon, de theta et phi sont tirées aléatoirement pour créer plusieurs cratères. L'angle theta est également incrémenté en fonction du temps pour faire tourner les cratères ensemble et donner l'illusion de la rotation de la planète.

Ombre

Pour donner un peu plus de réalisme encore à la scène et montrer les phénomènes d'éclipse, les ombres ont été mises en œuvre.

Le principe utilisé est identique à celui montré en cours : lorsqu'un astre autre que le soleil est intersecté par l'algorithme de ray marching, un second rayon est tiré depuis le point atteint vers le soleil pour déterminer si une planète est située entre l'astre et celui-ci. Lorsque c'est le cas, la couleur attribuée au pixel est la même que celle des zones d'ombres (voir plus haut).

Deux précautions doivent cependant être prises. Il faut d'abord limiter la distance maximale du second tracé de rayon car sinon un astre est forcément atteint (le soleil) et une ombre sera projetée inutilement. Il convient ensuite de ne pas réaliser ce tracé de rayon si le point atteint est situé dans la zone d'ombre d'une planète : tirer un rayon vers le soleil fera rencontrer un astre, la planète en question, et ajoutera une zone d'ombre supplémentaire à la zone déjà sombre. C'est le produit scalaire entre la normale à la surface au point atteint et le vecteur provenant du soleil qui permet de déterminer cette situation (négatif dans le cas d'une zone d'ombre).

Caméra

La position de la caméra est exprimée en coordonnées sphériques (R,theta,phi) :
R * vec3 (sin(phi)*sin(theta), cos(phi), sin(phi)*cos(theta)

Une correspondance est effectuée entre la position en x de la souris et l'angle theta, pour pouvoir faire varier cet angle de -π à π par déplacement du curseur depuis la partie la plus à gauche de l'écran jusqu'à la partie la plus à droite. En revanche, phi est fixé et ne varie pas en fonction du déplacement de la souris (cette possibilité a été testée mais est moins intéressante et moins « compréhensible »).

Etoiles au loin

Enfin, tous les objets de la scène ayant été rendus, le fond de celle-ci reste noir. Pour une représentation un peu plus réaliste, des étoiles ont été affichées. Le principe est le suivant.

Premièrement, l'algorithme de ray marching permet de déterminer facilement si un objet a été atteint en un pixel donné ou non. En effet, il attribue une distance égale à DIST_MAX dans la structure Astre qu'il renvoie si aucun objet n'est intersecté. Un simple test sur le champ dist de cette structure permet donc de savoir ce qu'il en est. Ensuite, pour un pixel retenu, un nombre aléatoire est tiré grâce à la fonction hash12 fournie en cours, qui renvoie une valeur entre 0 et 1. Un seuil sur cette valeur (qui a dû être fixé très bas) permet de déterminer si l'on colore ce pixel en blanc ou non. Enfin, un coefficient aléatoire variant en fonction du temps est affecté au pixel pour produire un scintillement. La fonction de scintillement correspondante reprend la fonction de bruit de Perlin fournie en cours et l'adapte à une variable d'entrée temporelle. Une précaution doit être prise pour éviter un scintillement identique entre les différents pixels.

Adresse du shader

Le shader que j'ai écrit peut être consulté à l'adresse suivante :
https://www.shadertoy.com/view/cdtXz4