Sprint Review

Création de la planète : L'on expérimente avec plusieurs types de sphères comme base pour notre planète. Initialement, pour générer des planètes nous partions de 0 en plaçant les vertices par code pour ensuite subdiviser la mesh crée.

Cela posait de nombreux problèmes comme :

- Devoir générer procéduralement les UV de chaque vertices, les normales etc.. Compliqués sur certaines sphères (ex : Icosahédron).
- Subdiviser efficacement, Compliqué sur certains type de Sphères (ex CubeSphere).

Nos solutions furent d'utiliser :

- Pour la subdivision efficace des outils "built-in" de Godot, le moteur graphique de notre choix qui nous donne accès à une mesh de cube dont la subdivision se facilement par une méthode préexistante.
- Pour l'icosaèdre la solution fu d'exporter un icosaèdre depuis Blender avec des informations tels que les UV et les Normales contenues dedans, il suffisait ensuite d'interpoler chaque valeur pour trouver les valeurs des vertices générées procéduralement, ce qui est plus simple.

Apparence de la planète : L'on a essayé de voir à l'avance comment l'on gèrerait l'apparence de notre planète et l'influence de son évolution sur celle-ci. L'on a implémenté une méthode qui utilise une librairie de bruit "built-in" afin de modifier l'altitude de chaque vertice en fonction de sa position. L'on a aussi expérimenté avec un "niveau d'eau" qui fait que toutes les vertices en dessous de ce niveau se retrouvent remonté au niveau de l'eau et leur couleur serait changé pour la couleur de l'eau. L'information de la couleur fut initialement stockée dans le canal de couleur des vertices.
Mais cette méthode est couteuse en performances car l'on se retrouve à boucler 2 fois sur tout les points en stockant une copie à chaque fois d'un tableau entier référençant les points.

On a donc ensuite essayé de réaliser tout ceci mais à l'aide d'un vertex_shader pour modifier la hauteur de nos vertices et avec un fragment_shader l'on gère ensuite la couleur des triangles en fonction de leur altitude. Les shaders sont bien plus efficaces et rapides en terme de calcul, pour un résultat similaire concernant les taches mentionnées. L'on va donc penser à les utiliser pour gérer l'apparence de notre planète par la suite.

Système de coordonnées : Pour l'icosphère, de base on était parti pour gérer les voisins sur la texture UV. Cela n'était pas une très bonne approche car le système de calcul devenait vite très pénible à adapter pour les différentes tailles de textures (car les coordonnées sont en pourcentages de 0.0 à 1.0).
L'on a donc envisagé de trouver les voisins d'un point en regardant les points avec les quels il formait un triangle, il suffisait de les ajouter de manière unique à un tableau les référençant. Cela nous donne les 5 voisins pour chaque point plus simplement qu'avec les UV.

Pour la cubesphère, on a d'abord du utiliser des couleurs de débogage afin d'observer et établir les différentes transitions entre les différentes faces (Exemple : bleu x=0 a comme coté adjacent vert y=max).
Les coins furent pénibles car notre première approche était de les traiter séparément mais en fusionnant les cas des cotés, puis enlever les coordonnées "invalides" et en doublon, le problème a été réglé et le code était plus concis.
On pense que l'arithmétique des cotés et transitions peut encore être optimisé pour avoir moins de tests.

Sprint Retrospective

Pour le moment il n'y a aucun problème pour travailler en binome, chacun a ses propres idées et entreprend des choses tout en informant l'autre et l'on s'explique régulièrement ce que chacun a fait. L'on réfléchit ensuite à la manière de mettre en commun ce que chacun a fait et l'on s'entre aide.

Product Increment

Numéro du dernier commit correspondant à la version finale du sprint
631043c1066a3124abc8ab3f6853cdf49998ae0a

Sprint Backlog

n°	Tâches	Séance 1	Séance 2	Séance 3	Etats
1	Trouver un sujet	2h	0	-	Terminé
2	Trouver un moteur graphique	1h	1	0	Terminé
3	Génération procédurale d'une planète	3-4h	3	0	Terminé
4	Mettre en place un système de coordonnées afin de placer les agents/objets	3h	3	1	En cours
5	Déterminer les entités à placer sur la planète et les conditions de leur placement (exemple : arbres, végétation; buissons, pierres, eau, ...)	2h			En attente
6	Evolution de la planète et génération d'entités en fonction des ressources	10h			En attente
7	Mettre en place le système de météorites pouvant apporter des ressources	2-3h			En attente
8	Mettre en place une UI et un joueur pouvant se déplacer et contrôler le monde (exemple : faire tomber des météorites, faire apparaitre des agents, ...)	6h			En attente
9	Mettre en place des agents dont les caractéristiques diffèrent (régime alimentaire, conditions de survie, vitesse etc...)	3h			En attente
10	Evolution des agents	8h			En attente
11	Statistiques	2h			En attente
12	Ajouts bonus (exemple : caméra centré sur agents, amélioration de la génération de la planète, optimisation des performances (compute shaders),)	-			En attente