Open World Game Prototype

Sujet proposé par Simon Guggisberg, IL

Introduction et contexte

Les applications en temps réel sont un sous domaine d'applications présentant bon nombre de défis techniques. L'entreprise du jeu vidéo, une de celles brassant le plus de revenus, est définitivement familière avec ces défis, puisque couplé à ceux-ci s'ajoutent les maints problèmes de rendu graphique, bien souvent 3D. Avec le temps, néanmoins, bon nombre de techniques ont vu le jour afin d'optimiser les performances des programmes. Mais une catégorie de jeu représente encore le summum de la complexité des applications en temps réel, tant en termes de rendu graphique que d'architecture : les jeux dit en monde ouvert.

Définition et présentation d'un jeu en monde ouvert

Un jeu monde ouvert a souvent comme caractéristique :

1. Vaste environnement dans lequel les joueurs et autres agents peuvent évoluer : Cela implique une complexité quant à charger différents éléments et garder en mémoire uniquement ceux faisant sens.
2. Environnement qui laisse percevoir jusqu'à une longue distance : Les techniques telle que l'occlusion culling qui allégeraient les problèmes d'affichage ne peuvent que difficilement être appliquées.
3. Gérer une large quantité d'objets : Que ce soit des objets interactibles, d'autres agents, ou des purements décoratifs, il convient de gérer ceux-ci efficacement.
4. Cycle jour-nuit : La technique pour alléger le calcul graphique de la lumière via le lightmap baking ne peut pas être appliquée. De plus toute source de lumière dynamique complexifie le rendu.
5. Progression non linéaire : Cela implique une communication entre les éléments pour régler les différentes situations plutôt que de passer par un manager en charge de l'exécution d'un niveau.
6. Adaptabilité aux changements : que ce soient un changement de météos, ou même de saisons. Avoir une architecture permettant de facilement représenter ceux-ci est souhaitable.
7. Modifications du monde : Au travers de l'édition du terrain, de la construction d'éléments. Ceci ajoute une complexité supplémentaire tant au niveau de la gestion des ressources des éléments ajoutés que via l'édition du terrain en soit, par la création de faces supplémentaires de manière dynamique.
8. Multijoueur : Un environnement en monde ouvert est souvent une excuse pour que plusieurs joueurs interagissent ensemble. L'architecture à choisir pour supporter des interactions physiques ou une dirigée par un serveur demande une réflexion particulière.
9. Génération procédurale : certains jeux en monde ouvert sont générés de manière procédurale. Cette génération peut utiliser différents algorithmes afin d'aboutir au résultat souhaité, tels que Cellular Automata, Perlin Noise, Voronoi Tesselation, Binary Space Partitioning, etc.

Présentation du travail de bachelor

Création d'un prototype de jeu vidéo en monde ouvert

La réalisation de jeux vidéo se passe bien souvent de manière itérative, avec une phase prototype afin de tester diverses idées et de se confronter aux problèmes pouvant survenir. Réaliser un prototype de jeu vidéo se concentrant sur les aspects et difficultés d'un monde ouvert en implémentant le state of the art d'un moteur de jeu vidéo utilisé dans l'industrie est le cadre de ce travail de bachelor.

Résumé du travail à effectuer

Définition du prototype

Le travail consistera en la réalisation d'un prototype de jeu vidéo en monde ouvert en 3D. Ce prototype contiendra un environnement dans lequel le joueur pourra se déplacer. Ce prototype servira de vertical slice

En outre, les points suivants définis en tant que composante d'un jeu en monde ouvert seront abordés dans les fonctionnalités :

- Vaste environnement :
  - Assets Loading
  - World Loading
  - Float Approximation
- Longue distance d'affichage :
  - LOD
  - Imposteurs
- Gestion d'une large quantité d'objets :
  - Optimisation par shader

Fonctionnalités

Cette liste de fonctionnalités est là à titre de cadre général. Celle-ci sera affinée lors de la rédaction du cahier des charges.

Required

- Assets et World Loading : Le fait de charger les ressources locales et les prochaines parties du monde requises par le jeu de manière asynchrone afin d'éviter temps de chargement à la moindre nouvelle ressource ou parcelle du monde rencontrée.
- Float approximation : Les moteurs de jeu utilisent des float en lieu de double pour accélérer les calculs. Avec de grandes distances, des erreurs d'approximation peuvent se produire. Une solution standard consiste à centrer l'origine du monde sur le joueur en tout temps.
- Contrôle de la caméra et/ou d'un avatar : Ne serait-ce que pour explorer l'environnement et tester celui-ci, le choix d'implémentation de vue, première ou troisième personne, est possible.
- Performances acceptables : Sujet sensible au vu la diversité des ordinateurs et des architectures/drivers offerts par certains. Un compromis de métrique serait un ordre de grandeur à respecter, plus de 30 frames par seconde tout en évitant les chutes de framerate hors d'écran de chargement.

Essential

- LOD et Imposteurs : Afin d'améliorer la performance en substituant des modèles complexes distants de la caméra par des moins détaillés ou des images. Requière un comportement spécifique pour certains éléments pouvant nécessiter un changement visuel malgré la distance. Un moulin à vent doit continuer de tourner, même s'il s'agit d'un imposteur.

Nice to have

- Optimisation par shader : Pour un élément simple répétable, n'ayant qu'un impact visuel, tel que l'herbe. Cet type d'élément peut aisément être représenté par un shader afin d'améliorer les performances en découplant la logique visuelle de celle de l'objet. Ici, implémenter une interaction avec l'herbe est en dehors du scope de ce projet.

Technologies utilisées

Chaque moteur de jeu contient des outils spécifiques pour certaines situations. Le moteur de jeu Unity, le plus populaire au niveau de l'industrie indie, conserve une grande part du marché malgré certaines décisions marketing. C'est avec ce moteur de jeu que le prototype sera réalisé en utilisant le state of the art de celui-ci.

Moteurs de jeu	**Sortie initiale**	**Rendu graphique**	**Langage**	**Open Source**
GameMaker	2007	2D	GML	Non
Godot	2014	2D/3D	GDScript/C#	Oui
Unity	2005	2D/3D	C#	Non
Unreal Engine	2014	3D	C++	Non

Ressources supplémentaires et mots clés

- AAA : Qualitatif pour des grandes productions de jeux vidéo nécessitant de grandes équipes, une grande coordination et un budget tout aussi conséquent
- Indie : Qualitatif pour des productions de jeux vidéo dites indépendantes, à équipe réduite. Originellement vient du fait que ces jeux étaient autopubliés, sans passer par un éditeur tiers, mais des publishers visant le public indie ont vu le jour depuis.
- Game jam : Événement se déroulant souvent sur un weekend, où programmeurs et artistes collaborent pour créer un jeu vidéo sur un thème donné. Réalisé en présentiel ou en ligne.
- Gameplay : La manière dont on expérimente et interagit avec un jeu vidéo, au travers des contrôles, de leur ergonomie, et des systèmes les gouvernant.
- Lightmap baking : Technique consistant à rendre, lors de l'édition d'un niveau, tous les effets de lumière sur une texture qui sera ensuite utilisée afin d'illuminer une scène, sans qu'aucune lumière ne soit utilisée en temps réel
- LOD : Level of Detail, technique consistant à substituer des modèles complexes par des moins détaillés en fonction de la distance à la caméra.
- Occlusion culling : Technique consistant à cacher les éléments invisibles à la caméra, afin de gagner du temps de rendu. À ne pas confondre avec le Frustum Culling qui consiste à cacher tous les éléments ne se trouvant pas dans le champ de la caméra.
- Shader : Programme spécialisé en charge du rendu graphique. Ils sont souvent écrits dans un langage différent que celui du moteur de jeu. HLSL pour Unity et Unreal Engine, GLSL pour Godot.

Sources

Popularité des moteurs de jeu :

- The Little Engines That Could