R-Type

Documentation utilisation Game Engine

EPITECH
LEARN DIFFERENT*

Sommaire:

Introduction:

Un moteur de jeu utilisant la méthode ECS ou Entity-Component-System est conçu pour rationaliser et optimiser le développement de jeux vidéo en séparant les différents aspects du jeu en entités, composants et systèmes.

Dans ce modèle, une entité représente un objet dans le jeu, un composant est une caractéristique ou un aspect spécifique de cette entité, et un système est une logique de traitement qui agit sur des entités possédant certains composants.

L'utilisation de l'ECS apporte une modularité accrue au développement du jeu, permettant aux développeurs de gérer plus efficacement la complexité croissante des projets.

Les avantages majeurs incluent la réutilisation des composants, la facilité d'ajout de nouvelles fonctionnalités, et des performances optimisées. L'ECS facilite également la parallélisation des opérations, offrant ainsi des gains significatifs en termes d'efficacité et de rapidité d'exécution.

En résumé, un moteur de jeu basé sur la méthode ECS offre une approche plus flexible et efficace pour structurer et développer des jeux vidéo complexes, en permettant aux développeurs de se concentrer sur des éléments spécifiques du gameplay sans sacrifier les performances.

Fonctionnement du Game Engine:

Notre Game Engine est construit de telle manière à réaliser une bibliothèque statique qui est ensuite utilisée tant par le serveur que par le client dans un jeu vidéo offre une architecture robuste pour la gestion des entités et des interactions entre le serveur et les clients.

Du côté du serveur, il agit en tant qu'autorité centrale, gérant la logique du jeu, les calculs critiques et la synchronisation de l'état du monde de jeu. Le moteur ECS est utilisé pour maintenir la vérité absolue du jeu, gérer les collisions, effectuer la simulation physique, et traiter les événements globaux.

Les entités du jeu, représentant des objets et des joueurs, sont mises à jour de manière centralisée par le serveur. Les composants spécifiques tels que la position, la santé, ou d'autres propriétés sont gérés dans le système du serveur, garantissant une cohérence globale du jeu.

De l'autre côté, chaque client utilise également son propre moteur ECS pour afficher via la bibliothèque SFML le jeu actualisé par les données serveur reçues.

Les clients reçoivent des mises à jour régulières du serveur pour synchroniser leurs instances locales avec l'état global du jeu.

Les événements générés localement par le joueur, tels que les mouvements ou les actions, sont ensuite transmis au serveur pour validation et propagation à l'ensemble des clients. Cela permet une réactivité instantanée aux actions des joueurs.

En résumé, l'utilisation d'un moteur de jeu ECS partagé entre le serveur et les clients dans un jeu vidéo offre une architecture distribuée et cohérente, permettant une interaction fluide entre les différentes parties tout en maintenant la synchronisation et l'intégrité du jeu.

Utilisation du Game Engine:

Pour utiliser le Game Engine efficacement il faut se servir de la classe “Registry” qui joue un rôle central dans un moteur de jeu basé sur le modèle Entity-Component-System.

La classe “Registry” agit comme un gestionnaire global qui stocke et organise les entités du jeu.

Elle offre des fonctionnalités essentielles telles que la création et la destruction d'entités, l'ajout et la suppression de composants, ainsi que la récupération de composants spécifiques associés à une entité.

En facilitant la gestion des entités et de leurs composants, le “Registry” permet au moteur de jeu d'appliquer des systèmes de manière efficace, contribuant ainsi à une conception modulaire, flexible, et extensible du code.

Son rôle central dans un moteur ECS en fait un composant clé pour la gestion dynamique des objets et la mise en œuvre cohérente de la logique du jeu.

Registry Méthodes Publiques:

Les méthodes publiques exposent des fonctionnalités essentielles pour l'interaction externe avec le Registry.

La méthode CreateEntity permet la création d'une nouvelle entité, attribuant un identifiant unique à chaque entité générée.

La fonction DestroyEntity offre la possibilité de retirer une entité spécifique du registre, participant ainsi à la gestion dynamique des objets du jeu.

Les méthodes template publiques, telles que addComponent, removeComponent, getComponent, et hasComponent, offrent une flexibilité considérable pour manipuler les composants associés aux entités.

Ces méthodes permettent d'ajouter, de supprimer, de récupérer et de vérifier la présence de composants de différents types.

Ces fonctionnalités sont fondamentales pour la construction et la modification dynamique des entités, offrant une approche modulaire pour la définition des caractéristiques individuelles des objets du jeu.

Registry Méthodes Privées:

Les méthodes privées, telles que addComponent, removeComponent, getComponent, et hasComponent, encapsulent la logique interne nécessaire pour effectuer ces opérations sur les entités et leurs composants.

Ces méthodes privées sont utilisées pour garantir une mise à jour cohérente du registre interne lors de l'ajout ou de la suppression de composants, contribuant ainsi à la stabilité et à la cohérence du système ECS dans le moteur de jeu.