Techniques d'optimisation : réduction des coûts de calcul, batching, LOD

Ce cours se concentre sur les différentes techniques d'optimisation utilisées dans le développement de jeux vidéo pour améliorer les performances. Vous apprendrez des concepts tels que la réduction des coûts de calcul, l'utilisation du **batching** pour optimiser les rendus graphiques, et la technique de **LOD (Level of Detail)** pour adapter la complexité des objets en fonction de leur distance à la caméra.

---

1. Réduction des Coûts de Calcul

Les jeux modernes peuvent être gourmands en ressources, en particulier lorsqu'ils incluent de grandes scènes, des effets visuels complexes ou des calculs physiques intenses. Réduire les coûts de calcul est crucial pour assurer un gameplay fluide et une performance optimale, surtout sur des dispositifs mobiles ou des plateformes avec des ressources limitées.

Stratégies pour réduire les coûts de calcul :

- **Optimisation des algorithmes :**
  - Utiliser des algorithmes plus efficaces pour les calculs, par exemple en remplaçant des algorithmes de tri ou de recherche coûteux par des alternatives plus rapides.
- **Réduction des appels de rendu :**
  - Minimiser le nombre d’objets qui nécessitent des appels au GPU pour le rendu (ex. : en réduisant la complexité des objets ou en utilisant des techniques comme le batching).
- **Cache et pré-calculs :**

- Pré-calculer les résultats des opérations complexes et les stocker pour les réutiliser, plutôt que de les recalculer à chaque image (ex. : pré-calcul de la lumière pour les objets statiques).

- **Utilisation de la parallélisation :**
  - Tirer parti du multithreading ou des unités de calcul spécialisées (GPU) pour répartir les tâches lourdes (comme le calcul de la physique ou le rendu) entre plusieurs cœurs ou processeurs.

---

2. Batching (Groupement des Objets pour le Rendu)

Le **batching** consiste à regrouper plusieurs objets dans un seul appel de rendu, ce qui réduit le nombre d'appels au GPU et améliore ainsi la performance du jeu.

Types de Batching :

- **Static Batching :**
  - Cette technique regroupe des objets statiques qui ne bougent pas. Unity, par exemple, peut combiner plusieurs objets qui ne changent pas dans un seul objet pour réduire les appels de rendu.
  - Cela permet de combiner des meshes simples dans un mesh plus complexe pour que le GPU les traite ensemble.
- **Dynamic Batching :**
  - Cette méthode regroupe des objets dynamiques (qui bougent) pour les rendre plus efficaces. Cependant, elle ne s'applique qu'aux objets ayant des caractéristiques similaires, comme les matériaux ou les shaders.
- **GPU Instancing :**
  - Cela permet de dessiner plusieurs instances du même objet sans générer de nouveaux appels de rendu. Chaque instance peut avoir des positions, rotations, et échelles différentes, mais elles partagent le même mesh et matériau, ce qui optimise les rendus.

Avantages du Batching :

- Réduction du nombre d'appels de rendu.
- Amélioration de la performance, surtout dans des scènes avec de nombreux objets statiques ou similaires.

3. Level of Detail (LOD) : Adaptation de la Complexité des Objets

Le **Level of Detail (LOD)** est une technique qui consiste à réduire la complexité des objets 3D lorsque ceux-ci sont éloignés de la caméra. Cela permet de réduire le nombre de triangles et donc les coûts de calcul pour le rendu.

Comment fonctionne le LOD :

- **LOD0** : Version la plus détaillée de l'objet, utilisée lorsque l'objet est proche de la caméra.
- **LOD1, LOD2...** : Versions de l'objet avec des niveaux de détail de plus en plus faibles à mesure que l'objet s'éloigne.
- Le moteur de jeu choisit dynamiquement quel niveau de détail afficher en fonction de la distance entre l'objet et la caméra.

Avantages du LOD :

- Réduction de la charge sur le GPU, particulièrement dans les scènes avec de nombreux objets à grande échelle.
- Amélioration des performances sans sacrifier la qualité visuelle perçue, car la simplification des objets à distance est généralement imperceptible pour les joueurs.

Mise en œuvre du LOD dans Unity :

- Unity permet de configurer le LOD via le composant `LOD Group`, où vous pouvez définir plusieurs modèles de niveau de détail et leur distance d'activation.
- Les objets peuvent être représentés par différents modèles 3D selon leur distance de la caméra.

---

4. Autres Techniques d'Optimisation

- **Culling (Suppression des objets non visibles) :**
  - Le **frustum culling** permet de ne rendre que les objets qui se trouvent dans le champ de vision de la caméra.

    - Le **occlusion culling** supprime également les objets bloqués par d'autres objets (par exemple, derrière un mur).
- **Optimisation des shaders :**
    - Utiliser des shaders optimisés pour réduire leur coût de calcul, notamment en simplifiant les effets visuels ou en réduisant le nombre de passes de rendu.
- **Réduction de la complexité des scènes :**
    - Utiliser des techniques comme le **terrain streaming** ou la **scène segmentée** pour ne charger que les parties du monde du jeu nécessaires à chaque moment.
- **Compression des textures :**
    - Compresser les textures en utilisant des formats optimisés pour l’affichage sans perdre trop de qualité visuelle, ce qui réduit la quantité de mémoire utilisée et améliore la performance.

Conclusion

Les techniques d’optimisation sont essentielles pour garantir la performance et la fluidité des jeux vidéo. La réduction des coûts de calcul, le batching des objets, et l’utilisation de LOD permettent de traiter efficacement un grand nombre d'éléments dans des environnements complexes. En intégrant ces méthodes dans votre développement, vous pouvez offrir une expérience de jeu plus agréable, même sur des appareils avec des ressources limitées.

Cela permet également de garantir que votre jeu sera performant sur une large gamme de plateformes, en particulier lorsqu'il est destiné à des appareils mobiles, des consoles, ou des systèmes avec des capacités graphiques limitées.