Bonnes pratiques de débogage : logs, détection d'erreurs courantes

Ce cours se concentre sur l'importance du débogage dans le développement de jeux vidéo. Vous apprendrez les bonnes pratiques pour identifier, comprendre et corriger les erreurs courantes dans vos jeux, ainsi que l'utilisation des **logs** pour faciliter le processus de débogage.

---

1. Importance du Débogage dans le Développement de Jeux Vidéo

Le débogage est un élément essentiel du processus de développement. Il permet de localiser et de corriger des erreurs dans le code, ce qui peut éviter des comportements inattendus ou des plantages pendant le jeu. Un bon débogage peut améliorer l'expérience de développement, rendre le code plus robuste, et optimiser les performances.

Les erreurs dans les jeux vidéo peuvent être liées à plusieurs aspects :

- **Erreur de logique :** Des comportements inattendus des objets ou des mécaniques de jeu.
- **Erreur de syntaxe :** Des erreurs dans le code empêchant l'exécution correcte.
- **Problèmes de performance :** Un jeu peut avoir des ralentissements ou une mauvaise réactivité si le code n'est pas optimisé.
- **Problèmes d'interface utilisateur (UI) :** L'UI peut ne pas s'afficher ou réagir comme prévu.

---

2. Utilisation des Logs pour le Débogage

Les **logs** sont un moyen très puissant de suivre l'exécution du code et d'identifier les erreurs. Ils permettent de consigner des informations

importantes sur l'état du jeu pendant son exécution.

Types de Logs :

- **Logs d'information (Info) :** Utilisés pour suivre le déroulement du jeu sans que ce soit une erreur. Par exemple, afficher des informations sur l'état d'un objet, le score, ou la progression d'un niveau.
- **Logs de débogage (Debug) :** Fournissent des informations détaillées pour aider à localiser les erreurs, comme les valeurs de variables, les étapes du programme, ou les comportements des objets.
- **Logs d'erreur (Error) :** Utilisés pour signaler des erreurs qui empêchent l'exécution correcte du programme. Cela peut inclure des exceptions, des erreurs de syntaxe ou des appels à des méthodes qui échouent.
- **Logs critiques (Fatal) :** Indiquent une erreur grave qui empêche le jeu de fonctionner, souvent utilisée pour signaler des plantages ou des problèmes de ressources critiques.

Exemple en C# (Unity) :

```
void Update() {
    if (score > 100) {
        Debug.Log("Score is over 100!");
    }

    if (health <= 0) {
        Debug.LogError("Player health is zero! Game ove
r.");
    }
}
```

Ici, `Debug.Log()` est utilisé pour afficher un message d'information, et `Debug.LogError()` pour signaler une erreur.

Bonnes pratiques pour l’utilisation des logs :

- Utiliser des niveaux de logs appropriés : ne pas inonder le programme de messages inutiles.

- Enlever ou désactiver les logs en production pour éviter la surcharge du système et améliorer les performances.
- Ajouter des informations pertinentes : inclure des détails comme les valeurs des variables, le nom des fonctions ou les étapes du processus où l'erreur se produit.
- Organiser les logs pour qu'ils soient facilement analysables (par exemple, les afficher dans un format structuré).

---

3. Détection des Erreurs Courantes dans les Jeux Vidéo

Les erreurs courantes dans les jeux vidéo incluent des problèmes de logique, de performance et d'interaction entre les éléments du jeu. Voici les types d'erreurs les plus fréquentes et comment les détecter :

3.1 Erreurs Logiques

- **Comportement imprévu d'un objet ou d'une fonctionnalité.**
  - **Exemple :** Un personnage traverse un mur ou un objet se déplace de manière erratique.
  - **Solutions :** Utiliser des logs pour suivre les positions, les entrées utilisateur et les états des objets. Vérifier les conditions qui régissent les déplacements ou les interactions.

3.2 Erreurs de Physique

- **Problèmes avec les collisions, la gravité ou le mouvement.**
  - **Exemple :** Un objet ne tombe pas correctement ou traverse un autre objet.
  - **Solutions :** Assurez-vous que les composants physiques comme les **Rigidbodies** et **Colliders** sont correctement attachés et configurés. Utilisez les logs pour vérifier la vélocité, la position et les forces appliquées aux objets.

3.3 Erreurs d'Interface Utilisateur (UI)

- **Problèmes d'affichage, boutons non réactifs, ou textes incorrects.**
  - **Exemple :** Le score ne s'affiche pas correctement.

- **Solutions :** Vérifiez que les éléments UI sont bien attachés à leurs objets et que les scripts mettent correctement à jour l'UI. Utilisez les logs pour surveiller les valeurs qui doivent être affichées.

3.4 Erreurs de Performance

- **Le jeu tourne trop lentement ou subit des chutes de framerate.**
  - **Exemple :** Un grand nombre d'objets ralentit le jeu.
  - **Solutions :** Utilisez des profils de performance pour identifier les goulots d'étranglement. Analysez les **frames par seconde (FPS)** et identifiez les objets ou scripts qui prennent trop de ressources.

---

4. Outils et Techniques Complémentaires pour le Débogage

- **Débogueur (Debugger) :**
  Utilisez un débogueur pour mettre des
  points d'arrêt dans le code, inspecter les valeurs des variables et suivre l'exécution ligne par ligne.
- **Profils de performance (Profiler) :**
  Utilisez des outils comme le
  Unity Profiler pour surveiller les performances du jeu en temps réel. Cela permet d'analyser l'utilisation du CPU, du GPU, de la mémoire et d'autres ressources.
- **Tests Unitaires et Tests d'Intégration :**
  Implémentez des tests automatisés pour détecter les erreurs avant qu'elles n'apparaissent dans le jeu. Cela peut inclure des tests sur des mécaniques spécifiques ou des interactions entre différentes parties du code.
- **Outils de Visualisation des Collisions :**
  Dans Unity, vous pouvez activer la
  visualisation des colliders pour voir si les objets interagissent correctement pendant le jeu, ce qui peut vous aider à repérer des problèmes de physique.

---

5. Bonnes Pratiques de Débogage

- **Documenter les erreurs :** Créez un journal de bord des erreurs pour comprendre comment elles sont survenues et comment elles ont été résolues. Cela peut être utile pour l'avenir.

- **Déboguer progressivement :** N'essayez pas de corriger plusieurs erreurs en même temps. Isoler chaque problème et résoudre une erreur à la fois.
- **Utiliser des tests automatisés :** Les tests unitaires peuvent être un excellent moyen d’identifier rapidement les erreurs dans votre code sans avoir à tout vérifier manuellement.
- **Collaborer avec l’équipe :** Si vous ne pouvez pas résoudre un problème, demander l'avis d'un collègue peut souvent offrir une nouvelle perspective.

Conclusion

Le débogage est une compétence essentielle pour tout développeur de jeux vidéo. L’utilisation de logs pour suivre l’exécution du code, ainsi que la détection et la correction des erreurs courantes, permet de rendre les jeux plus fiables et performants. En appliquant ces bonnes pratiques et outils, vous serez en mesure de créer des jeux plus solides, tout en réduisant le temps de développement et les bugs.