Moteur de jeu

Vertical-scrolling - Rail shooters

Arnaud Sanchez, Rémi Soulier et Thomas Rampin

Plan

1. Présentation générale
2. Sources d'inspiration
3. Outils et bibliographie
4. Aspects technique
    a. Génération du terrain
    b. Gestion de la scène
    c. Gestion des modèles
    d. Gestion des entrées / Contrôle du joueur
    e. Système de particules
    f. Lumière
5. Améliorations possibles et difficultés rencontrées
6. Conclusion

Présentation générale

Moteur de jeu basé sur le défilement de la carte

Défilement infini en se basant sur un bruit de perlin cohérent

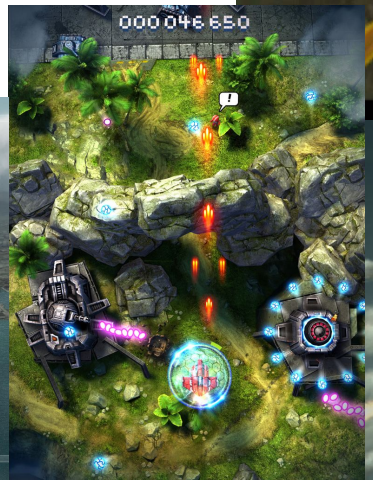
Chargement de modèle 3D (formats .3ds et .obj) avec Assimp

Modélisation arborescente de la scène

Moteur sous forme d’API

Sources d’inspiration

- Sky Force
- Spyro
- Ace Combat
- Tom Clancy’s H.A.W.X



Outils et bibliographie

Outils:

- Qt 5, C++ 11
- OpenGl
- libnoise
- Assimp

Bibliographie:

- Realtime Procedural Terrain Generation, J. OLSEN, 2004

Aspects technique

Génération du terrain avec la librairie `libnoise`

- Combinaison de plusieurs bruit de perlin
  - Un bruit pour les montagnes
  - Un pour les plaines
  - Un pour la répartition
- Modélisation en chunks
- Application de textures suivant l'élévation
- Ajout d'une skybox

Aspects technique

Gestion de la scène

- Arborescence
- Caméra relative à l'objet
- Transformation dans l'espace relatif au père

Aspects technique

Gestion des modèles

- Chargement des modèles avec Assimp
- Affichage de chaque mesh du modèle
- Objet de jeu
    - Manipulation dans l'espace
    - Gestion du rendu

Aspects technique

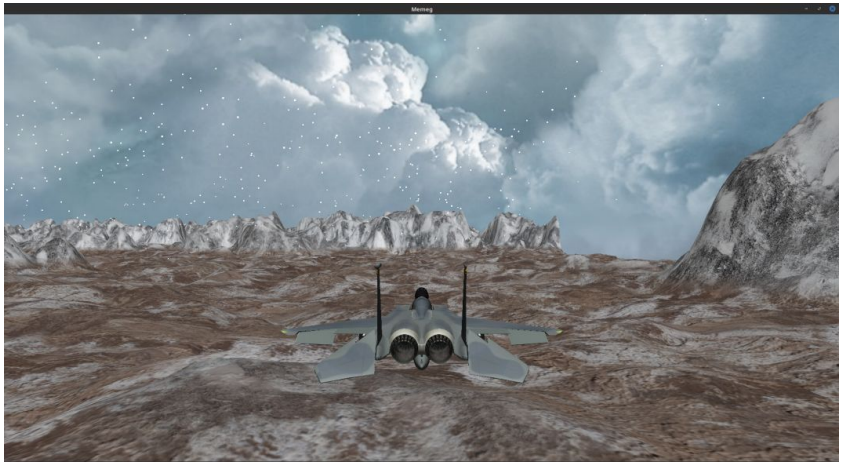
Contrôle du joueur

- Gestion des inputs clavier
- Tableau de booléen pour chaque touche
- Déplacements effectués dans l'update
- Permet de gérer plusieurs inputs à la fois
- Rotation des objets en fonction du virage
- Touche de tir

Aspects technique

Système de particules

- Système de saison
- Génération des particules de façon homogène
- Plusieurs types de particules possibles à générer
- Gestion dans les shaders
    - Apparition
    - Disparition



Aspects technique

Lumière

- Lumière directionnelle pour simuler le soleil
- Normal mapping pour le terrain
  - Meilleur rendu des détails des textures
- Modèle phong
  - Diffuse
  - Ambient
  - Specular

Difficultés rencontrées

- Chunker le terrain avec le bruit adéquate
- Jonction invisible entre deux chunks
- Manipulation des shaders
- Intégration de Assimp
- Normal mapping
- Différences entre les repères orthonormés
- Collisions

Améliorations possibles

- Gestion des collisions
- Ajouter des ennemis avec une IA
- Améliorer les bordures du terrain
- Étendre le moteur aux objets roulants
- Éviter le lag lors du chargement du prochain chunk
- Ajout des ombres
- Réduire l’impact en mémoire vive

Conclusion

- Moteur 3D simple
- Intégration d'une génération procédurale cohérente avec `libnoise`
- Intégration de la librairie Assimp pour les modèles 3D
- Utilisation du moteur sous forme d'API
- Possibilités d'amélioration multiples

Merci et place à la démo !