Rapport de projet

WebGL

Rendu interactif de scènes virtuelles



# Sommaire

[Sommaire 2](#_Toc27323024)

[Introduction 3](#_Toc27323025)

[Descriptions des effets réalisés 4](#_Toc27323026)

[a. Textures et implantation des objets 3D 4](#_Toc27323027)

[b. Déplacement du train 5](#_Toc27323028)

[c. Fumée 6](#_Toc27323029)

[d. Ciel et soleil 7](#_Toc27323030)

[e. Cycle jour/nuit 8](#_Toc27323031)

[f. Les options du GUI 9](#_Toc27323032)

[Conclusion 10](#_Toc27323033)

# Introduction

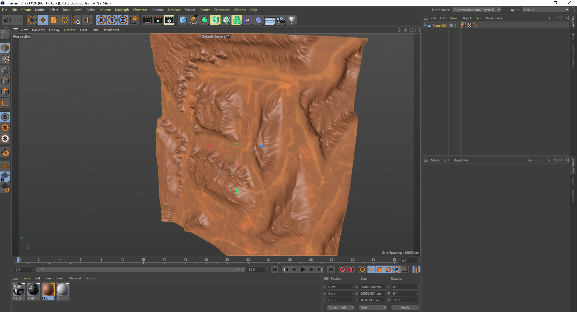
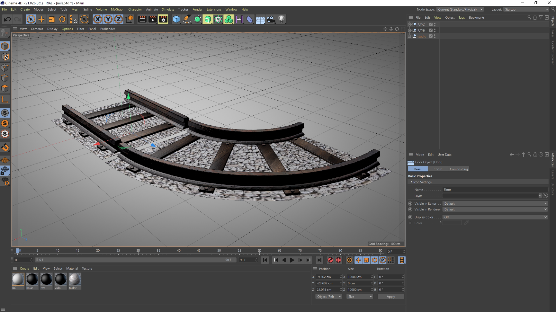
Le but de ce projet fut de réaliser une application interactive reposant sur les principes de l'informatique graphique vus en cours au travers de la technologie WebGL et de la librairie Three.js. Notre objectif a été d'améliorer la scène WebGL mise à notre disposition.

En voici une présentation où nous décrirons les effets réalisés et nous expliquerons nos choix de conception ainsi que les sources et inspirations utilisées.

# Descriptions des effets réalisés

## Textures et implantation des objets 3D

Avant toute chose, nous nous sommes rendus compteque nous ne pouvions récupérer les couleurs ou textures avec les fichiers fournis. De ce fait, nous avons recherché et trouvé les mêmes fichiers sur le site <https://www.cgtrader.com/free-3d-models> cependant, ils n’étaient pas tous en .obj et .mtl. Pour remédier à cela, nous avons utilisé les logiciels Blenders et Cinema4D afin de convertir les fichiers.

A partir de ces derniers, nous avons utilisé un OBJLoader et un MTLLoader pour charger les textures sur les objets.

Tout d’abord, le terrain utilise seulement l’OBJLoader puis nous chargeons les différentes textures avec un TextureLoader afin de l’appliquer sur toutes les Mesh de terrain (ici il n’y en a qu’un) à l’aide de la fonction traverse(). Nous avons choisi d’appliquer l’image « hight.png » en guise de texture d’émission de la lumière et ainsi voir ce canyon plus comme une vallée avec des montagnes enneigées.

Ensuite, pour les rails, nous chargeons d’abord les rails droits et affectons leur géométrie et leur matériau dans des variables globales afin de les réutiliser dans l’initialisation de chaque rail. Pour les rails tournants, nous attendons que les rails droits soient terminés pour les charger de la même manière car leur initialisation dépend des rails droits. Nous avons ajouté une bumpMap et specularMap sur les graviers afin de leur donner un peu de relief.

Enfin, la locomotive utilise les couleurs définies dans le fichier mtl. De plus, nous avons rajouter des phares qui s’allument en fonction du cycle jour/nuit (voir ci-dessous) positionnés dans le repère de la locomotive et de la fumée (voir ci-dessous) pour donner un peu de réalisme. De même, nous avons ajouté une deuxième caméra afin d’avoir une vue depuis la cabine.



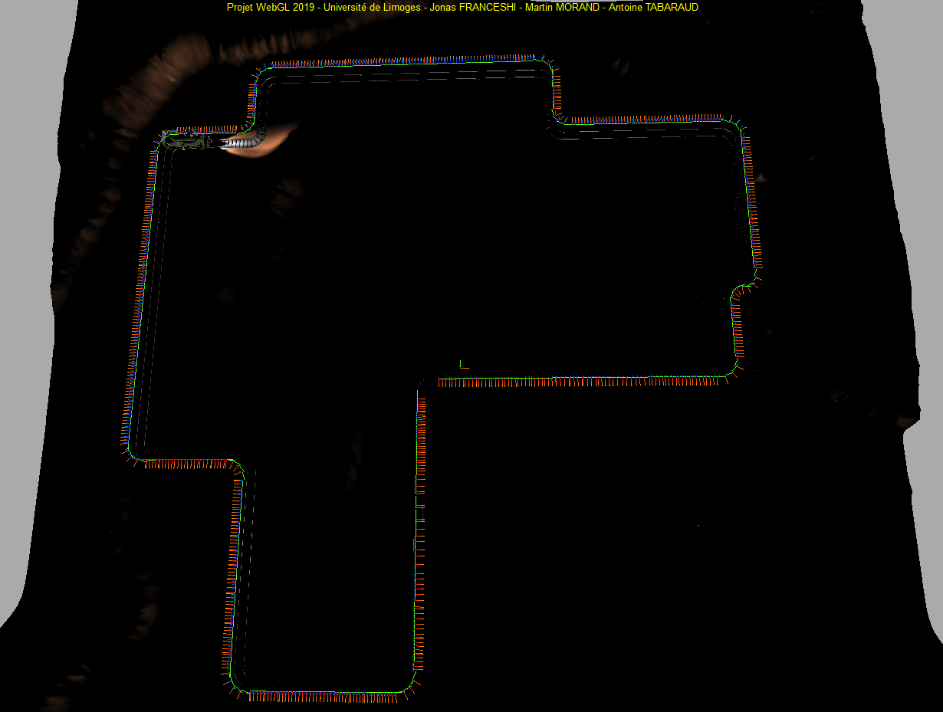
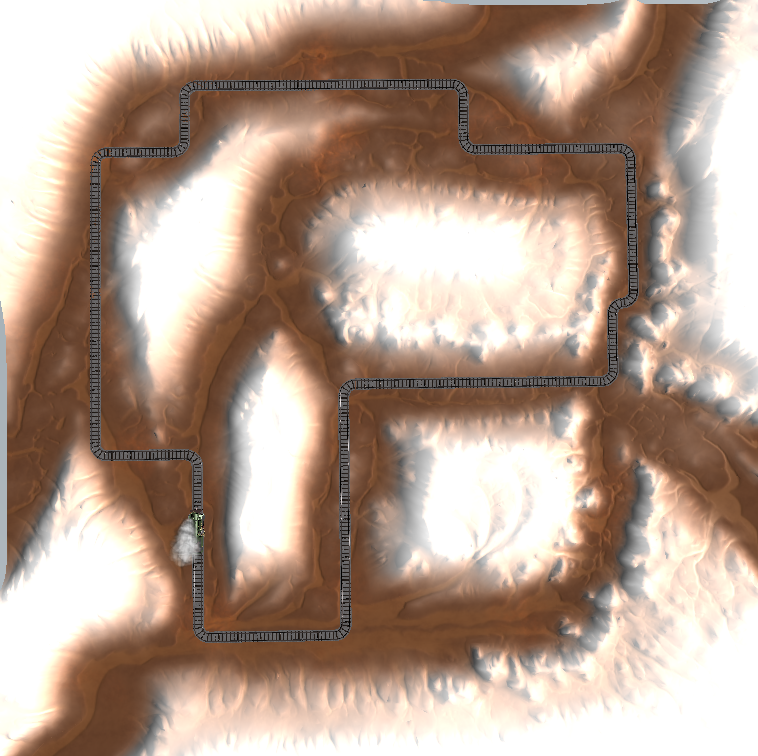
## Déplacement du train

Maintenant que tous les objets sont implémentés, intéressons-nous à la locomotive et son animation.

En effet, la partie la plus complexe du projet fut de déplacer le train sur les rails de façon réaliste.

Pour ce faire, nous avons tout d’abord placé les rails droits dans un tableau et les virages dans un autre. Chacun d’entre eux est positionné en fonction du rail précédent. Pour ce faire, nous avons utilisé des boucles for dans les lignes droites, et chaque virage est positionné en fonction du dernier rail droit.

Cependant, pour déplacer la locomotive, nous avons utilisé la librairie « Tween.js » qui permet de faire suivre un objet du point de départ vers un point d’arrivée. Cependant, avec ce système, lors d’un virage, la locomotive effectue une trajectoire en ligne droite et non courbée. Pour remédier à cela, il nous a fallu avoir un point intermédiaire et utiliser des interpolations. En effet, pour avoir une belle courbe dans les virages, nous avons utilisé une interpolation de Bézier et un calcul pour déterminer le point supplémentaire surtout dans les virages. Le calcul des points intermédiaires a été un vrai casse-tête pour les virages, d’autant plus qu’il fallait aussi une rotation intermédiaire. C’est la fonction animateLocomotive() qui se charge de calculer tous les cas possibles. Tant que le prochain rail n’est pas atteint c’est le TWEEN qui se met à jour tout seul. De plus, à chaque tour, il a fallu incrémenter de 2 \* PI la rotation du train par rapport aux rails car c’est un système qui tourne sur lui-même.

L’image ci-dessus, à gauche, montre les différentes positions du centre de la locomotive tout au long de son circuit. On peut donc voir une trajectoire qui suis bien celle des rails (image de droite).

## Fumée

Après le déplacement réaliste, nous avons ajouté de la fumée en nous inspirant d’une vidéo dont le but était de la simuler sur le logiciel blender ( <https://www.youtube.com/watch?v=tL0G5oOjdtk> ).

Nous avons donc 30 objets composés de 3 plans afin de voir la fumée quel que soit l’angle.

Chaque objet part de l’emplacement de la cheminée dans le repère général (la scène). Ils ont juste une trajectoire vers le haut et un changement de proportion.

En effet, la proportion des images permet d’avoir un effet de dissipation. D’autre part, la vitesse de montée change au fur et à mesure de la hauteur (en fonction du temps depuis leur sortie de la cheminée) ce qui donne un effet de courbe.

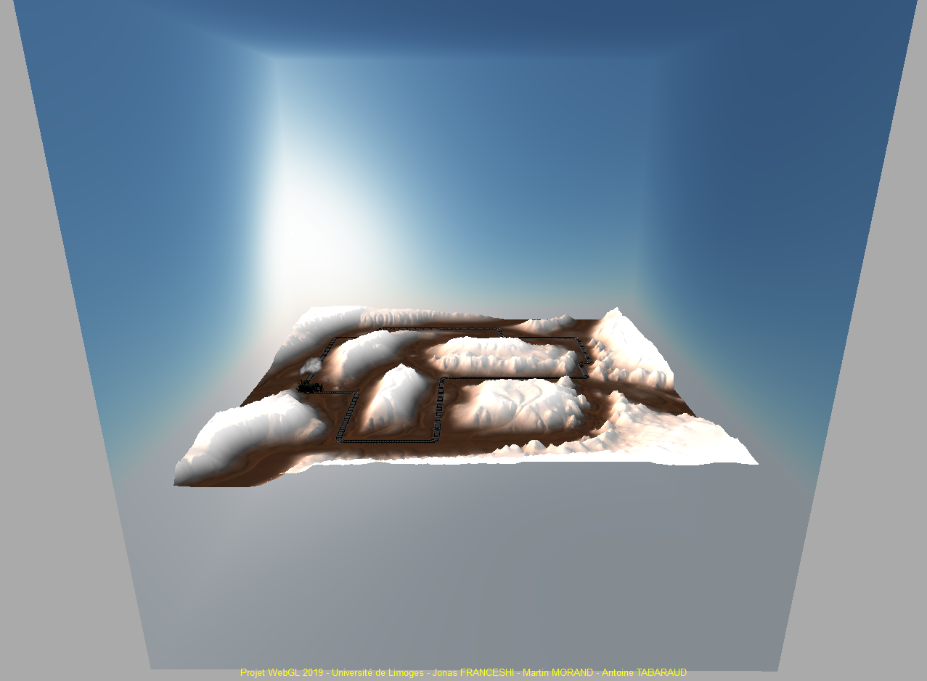
L’opacité joue aussi un rôle majeur afin d’avoir une fumée la plus réaliste possible, de même que l’ajout d’une texture de relief et de brillance. Bien entendu, plus la caméra se rapproche plus la qualité est médiocre (peu aussi laguer si votre processeur ou carte graphique n’est pas assez puissant).



## Ciel et soleil

Pour donner un peu de profondeur, nous avons intégré une skyBox sous forme de shader à l’aide de la librairie Sky.js ( <https://threejs.org/examples/jsm/objects/Sky.js> ) et d’un exemple disponible sur le site de Three.js ( <https://threejs.org/examples/webgl_shaders_sky.html> ). Cependant, comme l’orientation n’était pas bonne, nous avons changé le vertexShader en ajoutant des fonctions de rotation pour les axes X, Y et Z ( <https://gist.github.com/onedayitwillmake/3288507> ) puis nous les avons appelés dans le main() de ce vertexShader pour changer le vecteur worldPosition, tout comme nous pourrions le faire sur ShaderToy.

Ensuite, nous avons gardé les paramètres d’initialisation (sauf le GUI) en rajoutant une lumière au même emplacement que le soleil de la librairie. (Voir ci-dessous pour le changement de position du soleil).



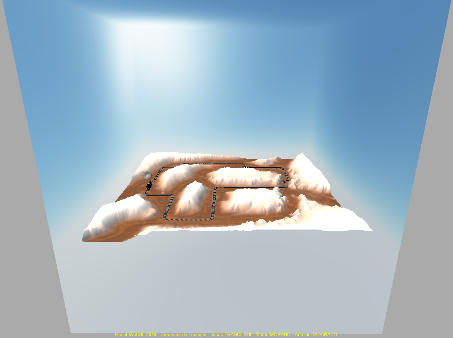
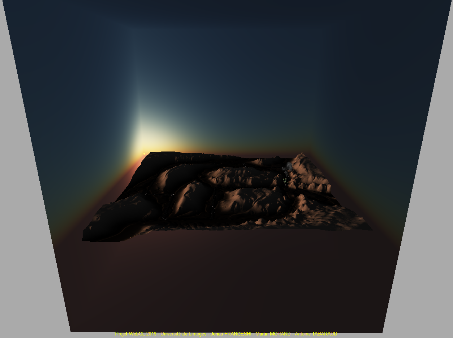
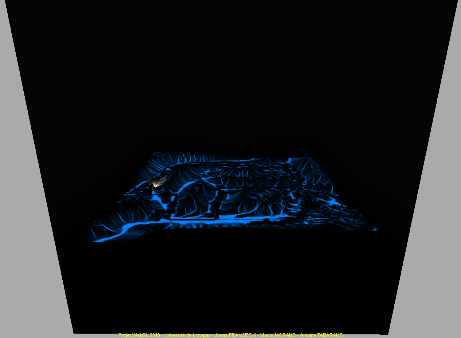
*SkyBox shader*

## Cycle jour/nuit

Afin de faire varier les ombres et le soleil, nous avons changé la position du soleil du shader et de la lumière en fonction du temps.

Pour donner un contraste entre le jour et la nuit, nous avons tout d’abord ajouté les lumières sur le train qui s’allument quand la hauteur du soleil passe sous l’horizon et inverse pour le jour. D’autre part, nous avons choisi de faire ressortir les rivières du terrain la nuit dans un soucis de luminosité. Pour cela, nous changeons progressivement l’intensité de la texture émissive.

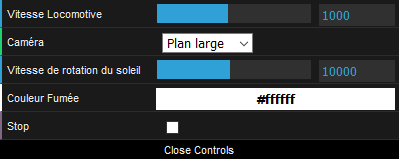
De plus, pour éviter de voir la fumée la nuit, nous mettons à zéro cette même valeur.

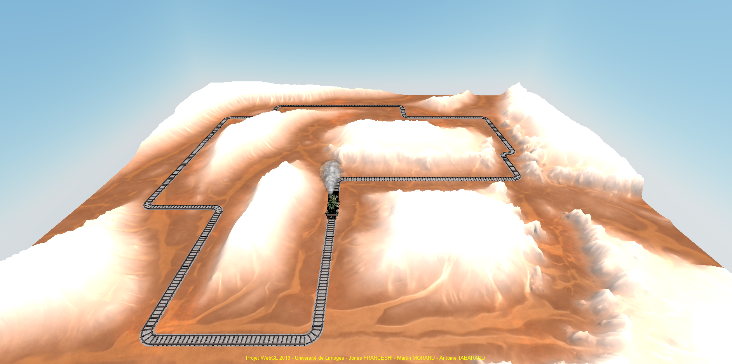
*Changement du soleil et de la texture du terrain.*

## Les options du GUI

Nous avons rajouté un GUI afin d’interagir avec les différents objets.

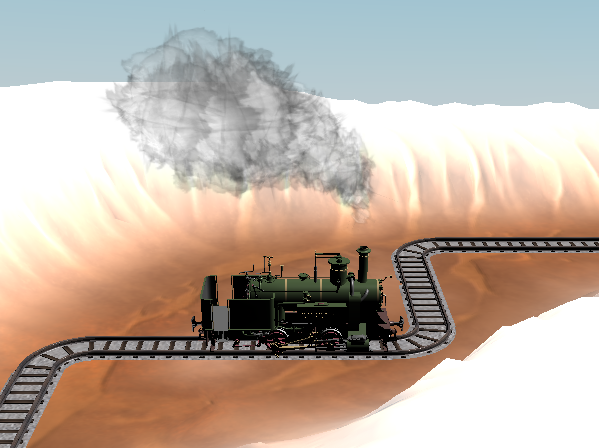
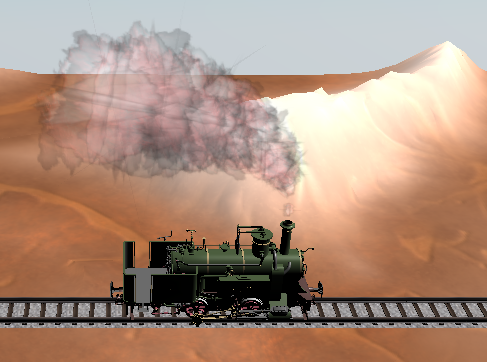
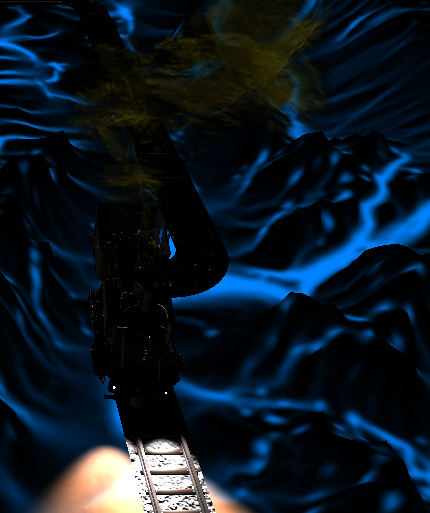
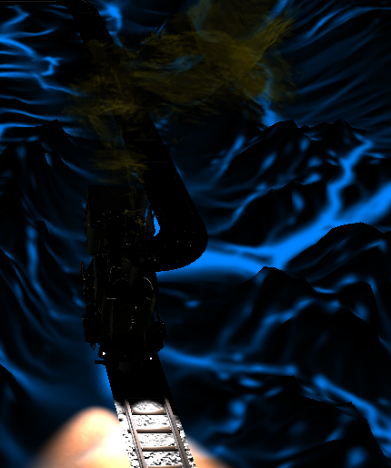


En effet, nous pouvons changer la vitesse de la locomotive. Mais aussi de caméra comme précisé plus haut :

*Camera plan large Camera locomotive*

Ainsi que la couleur de la fumée :

Enfin, en plus du changement de vitesse de la rotation du soleil, nous avons mis un bouton qui permet d’arrêter l’animation (cependant, il ne stoppe pas le temps).

# Conclusion

En guise de conclusion, ce projet fut très intéressant, même si nous n'avons pas eu une très bonne coordination concernant la répartition des tâches. Chacun de nous a, cependant, essayé d’appliquer au mieux les connaissances apprises lors des cours et TP.

Pour finir, nous sommes tout à fait conscients que nous pouvons améliorer notre projet avec des ajouts d’objets par exemple puis revoir la structure de la locomotive afin de faire tourner les roues ou encore utiliser un autre shader afin de faire circuler l’eau des rivières qui apparaissent la nuit.