Projet d’option Réalité Virtuelle n°10

Création d’un TP WebGL pour l’option

Encadrant : Jean-Marie Normand

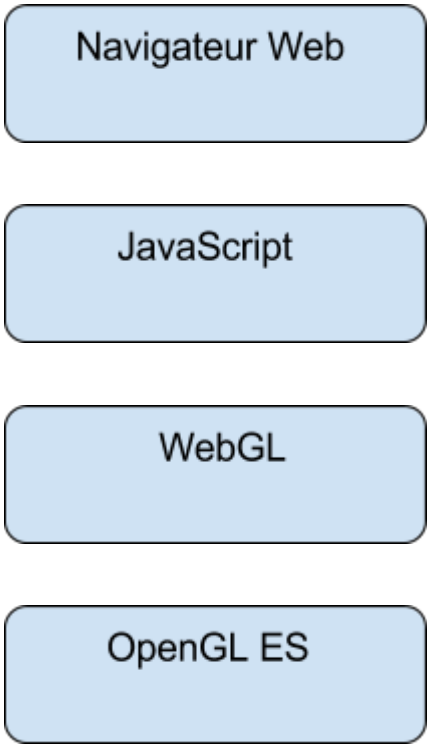
Alexandre Anchyse
Pierre-Louis Antonsanti

Table des matières

Présentation du projet

WebGL est une spécification d'interface de programmation de 3D dynamique pour les pages et applications HTML5. Il permet d'utiliser le standard OpenGL ES au sein d'une page web notamment. On peut ainsi d'afficher et gérer dynamiquement des objets 3D complexes dans la fenêtre du navigateur client. Il est implémenté sur la plupart des grands navigateurs (Chrome, Mozilla ou encore Safari par exemple) (1).

Les applications sont nombreuses, et vont du jeu vidéo en 3D à l'affichage du corps humain directement sur le web en passant par des applications de test d'animations physiques, de gestion des fluides ...



Il existe de plus de nombreuses librairies comme surcouches à WebGL permettant la facilitation de l'écriture JavaScript : Three.JS, babylon.JS, PlayCanvas… Le but de ce projet est d'utiliser l'un de ces framework pour faire découvrir les bases de l'implémentation graphique JavaScript avec l'interface WebGL. Le TP doit durer environ 8h, et. l'introduction à JavaScript doit être concise : elle pourrait faire l'objet d'un cours préalable.

Nous proposons d'utiliser le biais d'un petit jeu de plateforme pour apprendre à gérer les différents éléments de WebGL.

I/ Javascript, WebGL et babylonJS

Comme présenté dans l'introduction, la solution que nous devons proposer se base sur une surcouche de WebGL. Cette surcouche fait l'intermédiaire entre WebGL et Javascript. Nous recherchons donc un framework qui permette de créer une scène de jeu de plateforme. En effet, le WebGL en lui-même reste un langage de bas niveau peu propice à la mise en place d'applications directes. Les frameworks permettent de s'affranchir de ces considérations et laissent à l'utilisateur la possibilité de se concentrer sur la logique d'application du besoin.

Voici un tableau non-exhaustif reprenant différents aspects des principaux framework existants (2) :

Nom	Modélisation	Animation	Audio	Physique
BabylonJS	Non	Oui	Oui	Oui
Blender4Web	Oui	Oui	Oui	Oui
ThreeJS	Non	Oui	Oui	Non

Nous avons choisi de travailler avec BabylonJS. L'idée étant que le TP doit avoir un prétexte pour manipuler les éléments de webGL à partir d'une surcouche, nous avons décidé d'utiliser babylonJS qui est un framework accé jeu en 3D afin d'élaborer le sujet. Ainsi nous pouvons réaliser le jeu de plateforme tout en validant un certain nombre de critères à définir (cf II/).

En plus de son orientation jeu 3D, babylonJS présente l'avantage d'avoir une documentation détaillée et des exemples pour la plupart des cas de figure que nous pouvons rencontrer. Le système de "playground" et "sandbox" permet de visualiser en direct des scripts BabylonJS ou des objets .babylon sur le navigateur web. Une série de tutos vient compléter cette documentation (3) et fournit des outils supplémentaires pour les élèves.

Toutefois, puisque WebGL se base sur OpenGL ES, certaines fonctionnalités d'OpenGL ne seront pas disponibles : nous disposerons du vertex shader et du fragment shader mais pas du geometry shader, et il n'y a pas de texture en 3D.

II/ Les objectifs du TP

Nous nous inspirons des Travaux Pratiques suivis lors des premiers mois de Réalité Virtuelle pour déterminer les points à travailler au cours du TP WebGL. En particulier, les TP OpenGL et OpenSceneGraph nous ont paru intéressants puisqu'ils consistent également à prendre en main des interfaces de programmation de 3D dynamiques.

1) La création de mesh simples

https://www.babylonjs-playground.com/#3QW4J1#1

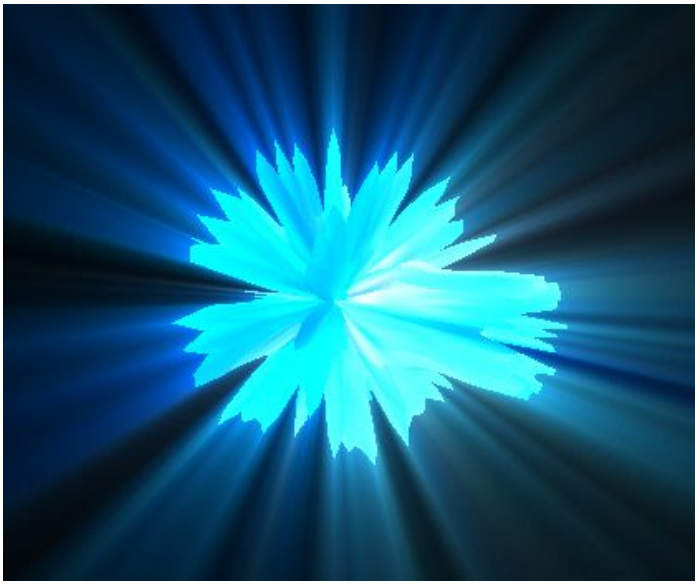
La première étape pour la création d'une scène 3D est d'arriver à afficher des objets. Le but étant, à terme d'être capable d'associer ces objets dans une même scène afin d'obtenir un jeu de plateformes cohérent. On va découvrir les différentes formes élémentaires (cubes, plans, sphères, cylindres ou encore prismes…) et leurs différents paramètres.

Voici une liste non-exhaustive des manipulations possibles sur les mesh :

- Changer leur position,
- Changer leurs dimensions,
- Leur appliquer des matériaux,
- Hiérarchiser des objets afin d'obtenir une structure complexe,
- Les animer,
- Créer des interactions,
- Leur appliquer une physique.

Le prétexte du jeu de plateforme est donc propice à la manipulation des mesh puisqu'il permettra de changer leur structure lors de la création de la scène, ainsi que de de leur appliquer les animations, interactions et physiques lors de l'animation du jeu.

2) Les lumières



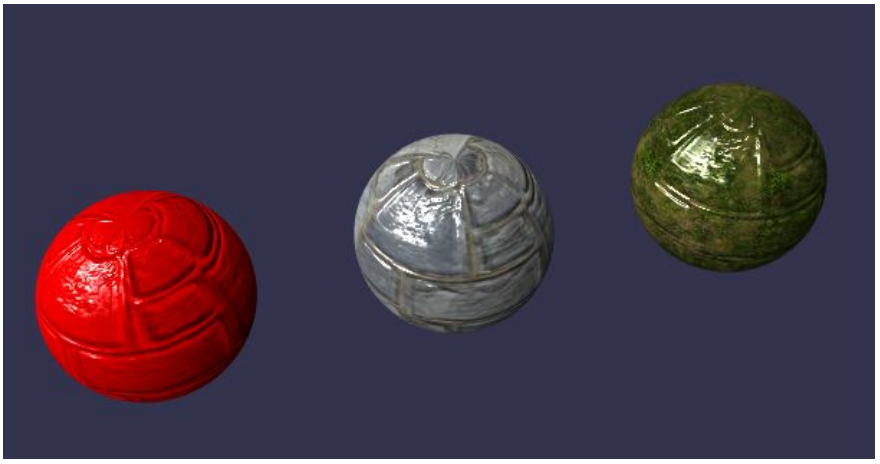
https://www.babylonjs-playground.com/#HYFQJ

Pour ce qui est des lumières, on peut les représenter de quatre façons différentes :

- La lumière ponctuelle, qui émet dans toutes les directions;
- La lumière directionnelle -comme son nom l'indique, émet dans une direction-,
- La lumière de type "spot lumineux" éclairant dans un cône,
- La lumière hémisphérique, qui peut être associée à la lumière ambiante.

Evidemment pour le projet, nous n'allons pas modéliser la lumière ci-dessus, mais il sera néanmoins intéressant de manipuler les différents types de lumières et leurs interactions avec les matériaux.

3) Les matériaux et textures



http://www.babylonjs-playground.com/#20OAV9#23

Les matériaux permettent aux objets d'interagir avec la lumière. Ils définissent la manière dont elle est réfléchie, émise… On va pouvoir jouer avec les couleurs (R,V,B,**α**) pour donner aux matériaux des aspects différent. Le quatrième canal définit la transparence de l'objet.

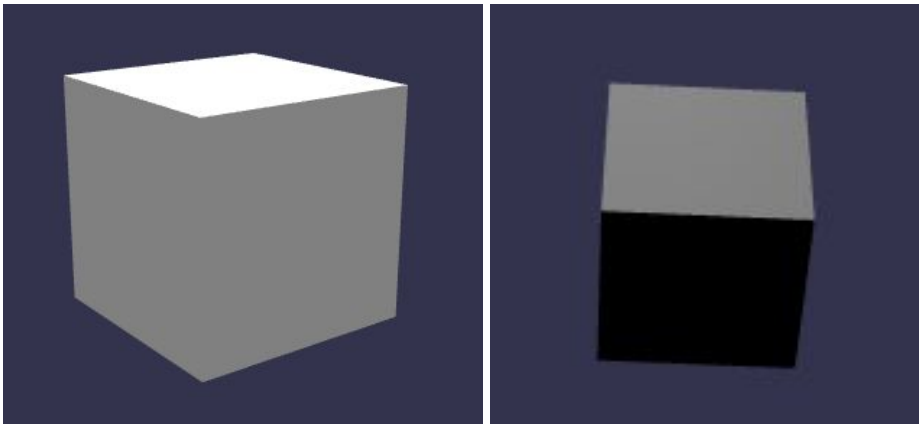
A un matériau donné on peut également ajouter une texture. Les possibilités sont nombreuses et dépendent essentiellement de l'imagination du designer. On peut tout de même relever plusieurs points à balayer :

- L'utilisation d'une texture simple,
- La répétition des motifs,
- L'ajout de reliefs via le bump-mapping ou le normal mapping.

Il existe aussi des méthodes de multi-texture, d' "environment mapping" … Ces aspects seront laissés à l'appréciation des élèves..

Il s'agit donc d'éléments esthétiques mais ils confèrent une réalité à la scène qui rend l'immersion et la plausibilité plus fortes. Ils seront donc exigés au cours du TP.

4) Les caméras



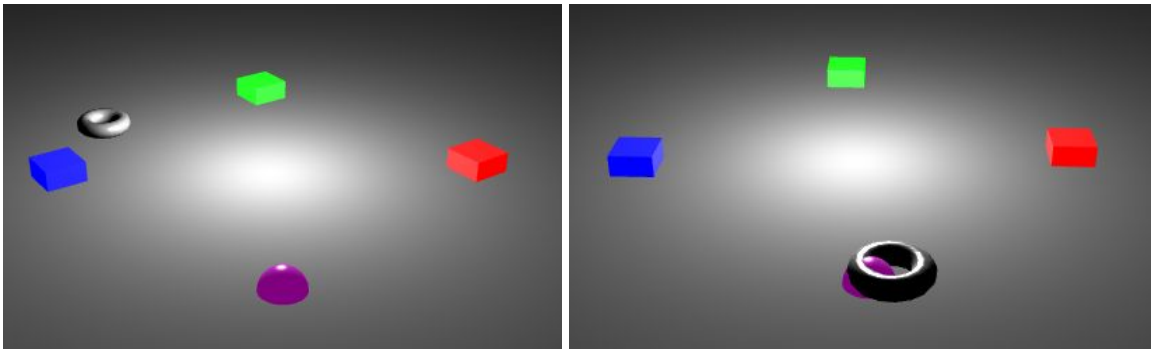
https://www.babylonjs-playground.com/#6FBD14#2

Les caméras sont essentielles à la mise en place de la scène. Elles permettent tout simplement à l'utilisateur de voir se scène, et éventuellement de naviguer au sein de celle-ci. Leur implémentation sera plus simple que pour OpenGL dans la mesure où les déplacements sont déjà gérés par la surcouche de WebGL. La seule difficulté pour la caméra dans notre projet sera liée à son bon appel dans les scripts : dans l'optique d'un jeu de plateforme, la caméra devrait être liée au personnage.

5) Les animations

Comme leur nom l'indique, il faudra animer la scène afin d'obtenir des mouvements. Nous ne recherchons pas des animations particulièrement complexes. Il faudra simplement prendre en main le système d'animations de la surcouche, et comprendre leur fonctionnement. On peut par exemple imprimer un mouvement circulaire à un objet, des translations, des rotations… Mais les animations s'appliquent également aux dimensions par exemple : il est possible d'appliquer des changements d'échelles ou des rotations selon les axes de l'objet.

6) Les interactions



http://www.babylonjs-playground.com/?17

L'exemple ci-dessus illustre l'interaction du tore avec l'objet violet de la scène : celui-ci grossit lorsque sa structure croise la demi-sphère.

Dans le cadre d'un jeu de plateforme les interactions peuvent être multiples, mais seront nécessaires. Elles peuvent être de la détection de collisions, des évènements liés à la souris ou au clavier, ou encore des lancers de rayons.

III/ Le TP webGL

La solution que l'on propose reprend donc les critères établis dans la section précédente pour former un TP de huit heures fournissant les bases du framework BabylonJS exploitant WebGL.
Pour le mettre en oeuvre nous nous sommes appuyés sur le cours OpenClassroom (4) qui fournit un panel assez complet de ce qui peut-être réalisé avec BabylonJS. Ce cours a pour but de créer un FPS multi-joueurs sur le navigateur. Pour la durée du TP nous n'avons évidemment pas gardé la totalité du cours mais nous en avons sélectionné quelques parties.

A partir de cette présentation, nous avons fixé la forme du dossier à fournir avec les scripts déjà préparés. Nous avons également conservé une grande partie du script weapon.js qui programme l'arme des joueurs dans le FPS, mais qui n'est pas le coeur de notre TP : il s'agit d'un plus pour les étudiants qui auront terminé le reste.

1) Introduction JavaScript

Dans une première partie du TP nous décrivons brièvement les points JavaScript qui vont permettre de réaliser le code dans la suite :

- La syntaxe, élément fondamental du code…
- La déclaration de variables
- La déclaration de fonctions
- La forme des objets et l'orientation prototypée
- Les callbacks

Ces points seront importants pour la suite de la réalisation, mais il n'y a pas besoin de beaucoup plus de compréhension pour aborder la création de la scène et l'exploitation de BabylonJS. La plupart des fonctions sont en effet implémentées dans le framework.

De plus, il n'y aura pas à se préoccuper de la partie HTML qui sera déjà écrite. Les scripts JavaScript seront appelés correctement et la seule condition de fonctionnement sera de ne pas les déplacer…

Une fois que le dossier fourni est en place, il suffit de double-cliquer sur index.html pour que la page web s'ouvre dans le navigateur. Il est possible également de l'ouvrir depuis le navigateur.

2) La création de la scène

Dans un second temps nous demandons aux élèves de se focaliser sur les points 1 à 5 de la partie précédente. Par soucis de simplicité, il n'y a pas trop de communication entre les différents scripts pour le moment. Nous nous focalisons sur Game.js et Arena.js

Game.js centralise tous les autres scripts en mettant en place les ponts entre ceux-ci.
Arena.js quant à lui va effectuer la création de l'arène complète, avec ses lumières, objets, matériaux, sons et animations… Cela semble court dit comme ça mais le passage d'un simple objet à une scène totale et des animations peut-être long.

Le but est d'obtenir une scène avec des objets animés permettant de réaliser un jeu de plateforme. Il faut donc au minimum contenu pour continuer le TP.

Nous avon pris le parti de mettre dans le sujet du TP le moins d'informations possibles pour l'élève. Ce sera à lui de chercher les fonctions à appeler dans la documentation de BabylonJS. Nous nous appuyons sur ce qui a déjà été fait pour prendre cette : cela force l'étudiant à effectuer des recherches dans la documentation -fournie- de BabylonJS. Nous donnons des indications directement dans les scripts sur 'emplacement des fonctions à appeler.

3) L'animation du personnage et les interactions

Lorsqu'une scène de jeu de plateforme est obtenue, il faut encore programmer les déplacements et actions du joueur. Pas de panique pour les personnes n'ayant pas fini la première partie, ou n'étant pas satisfaits par leur scène, une scène intermédiaire sera proposée au début de la deuxième partie.

Cette partie est un peu plus ardue car elle demande de mettre en place le contrôle de la caméra en style FPS, ainsi que l'animation d'un saut. Ceci requiert l'utilisation :

- Du callback addEventListener et de ses attributs,
- Des méthodes de déplacements en FPS avec des collisions,
- De lancer de rayons.

Une fois que nous avons ces déplacements il faut également ajouter les interactions du personnage avec l'environnement. Cela peut-être des collisions entre le personnage et des objets, des actions effectuées au clic…

Nous proposons une version "pour aller plus loin" où les élèves doivent programmer un tir dans un quatrième script Weapon.js.

4) Pistes d'amélioration

Au cours du projet nous avons rencontré quelques difficultés, notamment au niveau des imports et exports. Ceci a pour conséquence de limiter notre scène en ce qui concerne les objets et les flux de particules.
En effet, nous ne pouvons pas importer d'objets -même des .babylon- car nous recevons une erreur : "XML mal formé" dans la console du navigateur. Cette erreur semble venir du serveur en lui même et une solution avancée sur internet est d'utiliser une MIMEMAP pour définir ce que le serveur peut télécharger. Cependant malgré ces suggestions rencontrées sur différents sites, nous n'avons pas réussi à charger d'objet pré-fabriqué.

Pour faire fonctionner BabylonJS nous n'avons pas besoin d'installer quoi que ce soit. Les flux de particules sont alors gérés par un module complémentaire à BabylonJS qu'il faut appeler dans le .html. Lorsqu'on essaie de passer cet appel nous obtenons une erreur 404 et le module ne se charge pas.

Conclusion

Nous avons mis en place un projet de TP de 8h pour faire découvrir webGL aux élèves de l'option RV. Le choix du framework n'étant pas fixé, Jean-Marie Normand a accepté que nous travaillions avec BabylonJS. Nous disposons donc : du sujet de TP pour les élèves, du fichier de départ pour commencer le TP et d'une scène finale, comportant un petit jeu de plateforme, un système de saut et de contrôle de la caméra et d'un système de tirs.
La marche à suivre est indiquée dans le sujet mais les consignes restent vagues pour la bonne raison que le supplément d'informations nécessaires se situe dans les scripts. Les noms des fonctions, et des méthodes sont à trouver directement dans la documentation. Nous balayons les fondamentaux de ce qui peut-être fait avec WebGL, mais la durée du TP ne permet pas la création d'un jeu complet plus approfondi.

A partir de cette solution les élèves devraient être en mesure de créer leur propre scène dynamique 3D, de gérer des interactions et des évènements liés à l'utilisateur. En théorie -puisque nous n'avons pas réussi à les télécharger- on peut exporter des modèles d'objets 3D depuis Blender par exemple, et enrichir les éléments de la scène.

Il manque un test réel du TP mais nous pourrons certainement le mettre en oeuvre avec nos camarades cette année...

Sources

(1) https://fr.wikipedia.org/wiki/WebGL

(2) *https://fr.wikipedia.org/wiki/Liste_de_frameworks_WebGL*

(3) https://doc.babylonjs.com/how_to

(4) https://openclassrooms.com/courses/creez-votre-propre-fps-en-webgl/familiarisez-vous-avec-le-webgl-et-babylonjs