

**h e g**

Haute école de gestion  
Genève

# Intégration de la 3D sur un site Web grâce à WebGL



**Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES**

par :

**Ivo DE CARVALHO MATOSINHOS**

Conseiller au travail de Bachelor :

**Sonia PERROTTE, chargée d'enseignement**

**Genève, le 28 juin 2019**

**Haute École de Gestion de Genève (HEG-GE)**

**Filière Informatique de Gestion**

## Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre Bachelor of Science en Informatique de Gestion.

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : [http://www.orkund.fr/student\\_gorsahar.asp](http://www.orkund.fr/student_gorsahar.asp).

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 28 juin 2019

Ivo DE CARVALHO MATOSINHOS

## Remerciements

Je souhaite remercier la Haute École de Gestion ainsi que les professeurs m'ayant soutenu durant ma formation.

J'aimerais particulièrement remercier ma directrice de mémoire, Madame Perrotte Sonia qui s'est montrée d'une aide précieuse grâce à ses conseils et qui a su se rendre disponible pour m'aider.

Je remercie également tous mes proches, ma famille et mes amis, qui ont sû me soutenir pendant la réalisation de ce travail.

## Résumé

Internet a été une réelle révolution qui a permis de relier le monde grâce à un réseau informatique mondial. Puis est venu le World Wide Web qui tire profit d'Internet pour afficher des pages Web, qui sont des sources d'informations accessibles partout dans le monde.

À travers les années et grâce à ses évolutions, le Web est devenu tel que nous le connaissons aujourd'hui : une source d'informations gigantesque et interactive. Ces avancées ont pris du temps à façonner le Web sous sa forme actuelle et de nombreuses technologies en sont responsables. Le JavaScript, l'AJAX et le PHP font partie de ces technologies qui ont permis l'évolution du Web.

Aujourd'hui encore, de nouvelles technologies font surface comme le WebGL qui a été créé pour permettre de faire de la 3D sur une page Web. Ce travail a pour but d'explorer plusieurs aspects de WebGL pour mieux comprendre ce que cette technologie pourrait apporter au Web.

Premièrement, les technologies qui ont précédé le WebGL seront passées en revue afin de mieux cerner comment le WebGL est arrivé sur le marché. Cela permettra de comprendre ce qu'il a apporté par rapport à d'autres solutions.

Deuxièmement, une présentation plus approfondie de WebGL sera faite en abordant plusieurs points. Son aspect technique, ses avantages et inconvénients et ses alternatives actuelles et futures seront présentées afin de mieux connaître le WebGL.

Troisièmement, la place actuelle de WebGL sera analysée avec ses différentes évolutions et son taux d'utilisation afin de pouvoir déterminer quel sera le futur de WebGL.

Quatrièmement, différentes bibliothèques et frameworks seront abordés afin de présenter des solutions qui permettent de simplifier le développement. Une présentation spécifique de chaque solution permettra de savoir lesquelles sont les plus utilisées ou les plus adaptées à certains cas d'utilisation.

Cinquièmement, des cas concrets d'utilisation de WebGL seront présentés afin de démontrer que le WebGL peut être utilisé à des fins très poussées.

Dernièrement, une partie pratique sera réalisée afin de mettre en pratique les connaissances qui ont été acquises à travers ce travail de recherche.

# Table des matières

<b>Déclaration</b> .....	<b>i</b>
<b>Remerciements</b> .....	<b>ii</b>
<b>Résumé</b> .....	<b>iii</b>
<b>Table des matières</b> .....	<b>iv</b>
<b>Liste des tableaux</b> .....	<b>vi</b>
<b>Liste des figures</b> .....	<b>vi</b>
<b>1. Introduction</b> .....	<b>1</b>
<b>2. Les solutions existantes avant le WebGL</b> .....	<b>2</b>
<b>2.1 VRML</b> .....	<b>2</b>
2.1.1 Historique .....	2
2.1.2 Fonctionnement .....	2
<b>2.2 X3D</b> .....	<b>2</b>
2.2.1 Historique .....	2
2.2.2 Fonctionnement .....	3
<b>2.3 O3D</b> .....	<b>4</b>
2.3.1 Historique .....	4
2.3.2 Fonctionnement .....	5
<b>3. Le standard Web3D actuel</b> .....	<b>6</b>
<b>3.1 Introduction</b> .....	<b>6</b>
<b>3.2 Élément HTML</b> .....	<b>6</b>
<b>3.3 Architecture</b> .....	<b>6</b>
<b>3.4 Fonctionnement d'un rendu 3D</b> .....	<b>7</b>
3.4.1 Système de coordonnées 3D .....	7
3.4.2 Étapes de calcul du rendu 3D .....	9
<b>3.5 Avantages et inconvénients</b> .....	<b>10</b>
3.5.1 Avantages .....	10
3.5.2 Inconvénients .....	11
<b>3.6 Les alternatives actuelles à WebGL</b> .....	<b>11</b>
<b>3.7 Les futures alternatives possibles à WebGL</b> .....	<b>12</b>
3.7.1 WebGPU .....	12
3.7.2 Vulkan .....	13
<b>3.8 Quel est le futur du Web3D ?</b> .....	<b>14</b>
<b>3.9 Sécurité de WebGL</b> .....	<b>15</b>
<b>4. Evolution de l'utilisation de WebGL</b> .....	<b>18</b>
<b>4.1 Compatibilité des navigateurs d'ordinateur de bureau</b> .....	<b>18</b>
<b>4.2 De WebGL 1.0 à WebGL 2.0</b> .....	<b>18</b>

4.3	Evolution du taux d'utilisation .....	19
<b>5.</b>	<b>Les bibliothèques WebGL .....</b>	<b>21</b>
5.1	Pourquoi faire des bibliothèques ? .....	21
5.2	Différences entre une bibliothèque et un framework .....	22
5.3	Les bibliothèques/frameworks les plus utilisées.....	23
5.4	Les bibliothèques 2D .....	24
5.4.1	PixiJS .....	24
5.5	Les bibliothèques/frameworks 3D .....	25
5.5.1	Three.js .....	26
5.5.2	SceneJS.....	29
5.5.3	Blend4Web.....	31
5.5.4	BabylonJS .....	33
<b>6.</b>	<b>Cas concrets d'utilisations de la 3D .....</b>	<b>36</b>
6.1	Personnalisation de produits .....	36
6.2	Visualisation de données statistiques .....	38
6.3	Cartographie 3D sur Google maps .....	39
6.4	Visualisation de données économiques .....	41
<b>7.</b>	<b>Réalisation pratique .....</b>	<b>43</b>
7.1	Three.js.....	43
7.2	SceneJS.....	43
<b>8.</b>	<b>Conclusion .....</b>	<b>45</b>
	<b>Bibliographie .....</b>	<b>46</b>
	<b>Annexe 1 : Compatibilité des premières versions des navigateurs d'ordinateur de bureau .....</b>	<b>51</b>
	<b>Annexe 2 : Compatibilité des premières versions des navigateurs pour mobiles .....</b>	<b>52</b>
	<b>Annexe 3 : Partie pratique – fichier « index.html ».....</b>	<b>53</b>
	<b>Annexe 4 : Partie pratique – fichier « houseViewerSatigny_3D.html » .....</b>	<b>56</b>
	<b>Annexe 5 : Partie pratique – fichier « houseViewerMaquette.html » .....</b>	<b>59</b>
	<b>Annexe 6 : Partie pratique – fichier « houseViewerCologne_3D.html » .....</b>	<b>63</b>
	<b>Annexe 7 : Partie pratique – fichier « houseViewerCologne_2D_RDC.html ».....</b>	<b>66</b>
	<b>Annexe 8 : Partie pratique – fichier « houseViewerCologne_2D_1er.html » .....</b>	<b>69</b>
	<b>Annexe 9 : Partie pratique – fichier « style.css » .....</b>	<b>72</b>

## Liste des tableaux

Tableau 1 : Avantages de WebGL .....	10
Tableau 2 : Inconvénients de WebGL.....	11

## Liste des figures

Figure 1 : Schéma du fonctionnement de X3D .....	4
Figure 2 : Schéma du fonctionnement de O3D.....	5
Figure 3 : Schéma de fonctionnement de WebGL.....	7
Figure 4 : Système de coordonnées 3D.....	8
Figure 5 : Représentation d'un cube à l'aide de vertices .....	8
Figure 6 : Processus de calcul du rendu 3D .....	9
Figure 7 : Visionneur 3D de Volkswagen .....	14
Figure 8 : Fonctionnement d'une attaque via WebGL .....	16
Figure 9 : Pourcentage d'utilisateurs compatibles avec WebGL 1.0 .....	20
Figure 10 : Pourcentage d'utilisateurs compatibles avec WebGL 2.0 .....	20
Figure 11 : Exemple basique de code WebGL .....	21
Figure 12 : Différences entre une librairie et un framework.....	22
Figure 13 : Popularité des bibliothèques et frameworks.....	23
Figure 14 : Différents filtres de PixiJS .....	25
Figure 15 : Exemple de code basique de Three.js .....	27
Figure 16 : Structure d'une rendu 3D Three.js.....	27
Figure 17 : Application d'une texture à base d'une image.....	28
Figure 18 : Matériaux de base disponibles .....	28
Figure 19 : Modèle 3D interactif d'un cerveau .....	30
Figure 20 : Exemple de code SceneJS.....	30
Figure 21 : Interface graphique de Blend4Web .....	32
Figure 22 : Editeur en ligne de BabylonJS .....	34
Figure 23 : Autre utilisation de BabylonJS .....	35
Figure 24 : Système de personnalisation de chaussures Nike.....	37
Figure 25 : Système de personnalisation de chaussures Kenzo.....	38
Figure 26 : Représentation graphique interactive en 3D .....	39
Figure 27 : Google Street View sans et avec WebGL .....	40
Figure 28 : Vue satellite à 45° en 3D.....	41
Figure 29 : Liens d'exportations de la Suisse .....	42

# 1. Introduction

Depuis l'arrivée du Web 2.0 en 2007, le Web s'est orienté vers une utilisation plus accessible grâce à des interfaces simplifiées. Il a également apporté la notion d'interactivité et de partage entre les internautes, sans qu'ils n'aient besoin de notions particulières pour utiliser ces nouvelles fonctionnalités. Depuis, le Web n'est plus une source d'information figée mais une plateforme de partage où chaque utilisateur peut contribuer à l'échange d'informations. Grâce à ce changement tendanciel qui veut orienter le Web vers une plateforme plus participative, de nombreuses nouveautés ont vu le jour. Les réseaux sociaux et le commerce en ligne font partie de ces avancées nécessitant de nouvelles technologies pour les appuyer.

Ces avancées ont permis d'un côté l'amélioration de l'expérience des utilisateurs, mais ont de l'autre côté complexifié le fonctionnement interne du Web. Pour enrichir l'expérience des utilisateurs, différentes technologies ont fait leur apparition comme le JavaScript. Cette technologie est considérée comme un pilier fondamental du World Wide Web qui a permis de rendre les pages Web interactives. Le JavaScript n'étant qu'une technologie parmi tant d'autres qui ont apporté leur lot de nouveautés.

Toujours dans l'optique de faire du Web une expérience riche, la question de rendre des éléments graphiques plus dynamiques et interactifs s'est posée. La 3D sur le Web est donc devenue un enjeu important pour permettre à certains domaines comme la science, les jeux vidéo et la visualisation de données d'évoluer. Plusieurs technologies ont vu le jour comme le VRML, qui a été l'une des premières solutions utilisées pour faire de la 3D sur le Web, ainsi que bien d'autres comme X3D ou encore O3D.

Finalement en 2011 est arrivé le WebGL 1.0, une API créé par le Khronos Group permettant de faire de la 3D dynamique sur des pages qui utilisent l'HTML 5. Le fonctionnement de WebGL est possible grâce à plusieurs éléments comme le JavaScript et l'accélération matérielle pour faire le rendu 3D à l'aide d'OpenGL ES. Le WebGL n'a pas été une révolution à proprement parler, mais il a apporté certains avantages que d'autres solutions n'avaient pas. Un de ses plus grand avantage étant le fait de ne pas nécessiter de plug-in additionnel pour assurer son fonctionnement. En partie grâce à ces avantages, le WebGL s'est imposé comme standard et est compatible avec la plupart des navigateurs depuis 2012 et depuis 2013 pour Internet Explorer.



## **2. Les solutions existantes avant le WebGL**

### **2.1 VRML**

#### **2.1.1 Historique**

Le VRML ou « virtual reality modeling language » a été créé en 1994 par des employés de Silicon Graphics et présenté à la convention du World Wide Web en 1994 à Genève. Le VRML a pour but de permettre de décrire des environnements 3D. Dans sa première version, ses spécificités étaient les suivantes :

- Indépendance au niveau de la plateforme.
- Extensibilité.
- Possibilité de fonctionner avec des connexions à la bande passante réduite.

Le but principal était donc de rendre le VRML exploitable sur le Web, les possibilités 3D offertes par le VRML étaient par conséquent moins au centre des préoccupations. Il était possible de créer des formes de base (cube, sphère, cylindre, cône) qui pouvaient ensuite être combinées entre elles. Cependant, il n'était pas possible d'animer ces formes, les environnements étaient uniquement statiques et non interactifs.

La deuxième version de VRML a vu le jour sous l'appellation de VRML97 ou encore VRML 2. Elle apporta de nouvelles fonctionnalités comme les animations et l'interaction des utilisateurs avec la scène. Le VRML n'a pas eu un grand succès, sans doute dû au fait que la problématique n'était pas assez présente dans ces années pour en faire un langage très exploité. Il a donc été peu à peu remplacé par X3D qui est arrivé un peu plus tard.

#### **2.1.2 Fonctionnement**

Le VRML n'est pas un langage de programmation mais un langage de description. Comme l'HTML, le VRML n'est pas constitué des formants algorithmiques classiques mais plutôt d'une suite d'informations qui décrivent les éléments qui seront affichés. Grâce à cette description, les objets peuvent être situés dans l'espace et il est possible de définir leur forme, couleur, texture et d'ajouter des sources de lumière. La description se fait dans un fichier texte utilisant l'extension « wrl » puis est interprétée pour afficher des éléments graphiques à l'écran. Tout ceci se fait sans avoir besoin de gérer la manière dont est interprété notre fichier.

### **2.2 X3D**

#### **2.2.1 Historique**

Le X3D ou « extensible 3D » a été créé en 1999 par un groupe de travail appartenant au Web3D Consortium qui s'occupe de l'évolution de ce format. Il est le successeur de

VRML et reste compatible avec celui-ci. Il reste un langage descriptif qui existe sous diverses syntaxes mais est aussi un format de fichier. Le X3D a été normalisé en 2005 par l'ISO pour devenir un standard du Web3D et se développe actuellement sous sa quatrième version. Il reprend les spécifications principales de VRML tout en évoluant pour répondre aux problématiques actuelles. Voici quelques avancées que le X3D a apporté :

- Possibilité d'utiliser différentes syntaxes (syntaxe basée sur l'XML, VRML ou en binaire).
- Amélioration du rendu graphique et audio.
- Possibilité de faire du rendu multi-textures.
- Gestion des ombres en temps réel.
- Gestion des réflexions en temps réel.

Le X3D est une solution parmi tant d'autres pour faire de la représentation 3D sur le Web. En effet, une partie des grandes entreprises faisant parti du Web3D Consortium ont sorti leur propre technologie 3D.

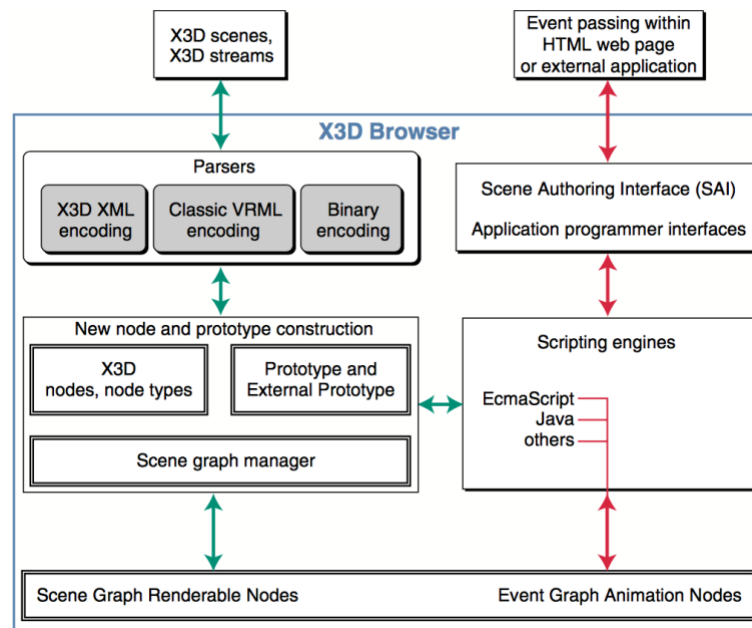
- Intel : U3D.
- Microsoft : XAML.
- Dassault Systèmes : 3D XML.

Mais l'avantage de X3D par rapport à ses concurrents est le fait que c'est une solution gratuite, accessible et bien documentée. X3D est un format toujours utilisé de nos jours, notamment pour exporter des modèles 3D faits sur des logiciels comme Blender. Malgré toutes ces avancées, les solutions de cette époque ont toutes une particularité qui est aujourd'hui considérée comme un défaut. Il était nécessaire d'installer un plug-in additionnel au navigateur Web client pour afficher les scènes 3D produites et les rendre interactives.

### **2.2.2 Fonctionnement**

Comme introduit plus haut, le navigateur Web du client doit être équipé d'un plug-in pour interpréter le X3D. Voici l'architecture d'un navigateur disposant d'un plug-in X3D :

Figure 1 : Schéma du fonctionnement de X3D



[https://x3dgraphics.com/examples/X3dForWebAuthors/Chapter01-TechnicalOverview/Chapter01Technical\\_Overview.pdf](https://x3dgraphics.com/examples/X3dForWebAuthors/Chapter01-TechnicalOverview/Chapter01Technical_Overview.pdf)

L'architecture de X3D est composée d'une première couche qui contient des « parsers » permettant d'interpréter la scène X3D sous ses différents formats disponibles (X3D XML, VRML et binaire). Le résultat est ensuite produit grâce au « scene graph manager » qui parcourt les nœuds X3D. Ces nœuds sont la description des éléments sous forme de balises XML avec les éléments et leurs attributs. Le « *scene graph manager* » gère tous les paramètres des objets comme leur forme, apparence, localisation et orientation. Ces informations permettront de dessiner la scène sur le navigateur client et gérer les interactions du client avec celle-ci. La scène peut être modifiée via les interactions du client grâce au navigateur mais également via du code JavaScript ou Java. Le résultat final est ensuite affiché comme un élément graphique interactif sur un navigateur Web équipé du plug-in X3D ou sur une application externe compatible.

## 2.3 O3D

### 2.3.1 Historique

O3D est une API open source créée par Google en 2009 permettant de créer des applications 3D sur un navigateur Web. Contrairement à VRML et X3D, O3D est un langage de programmation et non un langage de description, les applications 3D sont produites via une API en JavaScript. O3D nécessite l'installation d'un plug-in qui ajoute des fonctionnalités au navigateur client. Le plug-in permet d'utiliser l'accélération matérielle pour que les rendus 3D soient produits par la carte graphique et non le micro-

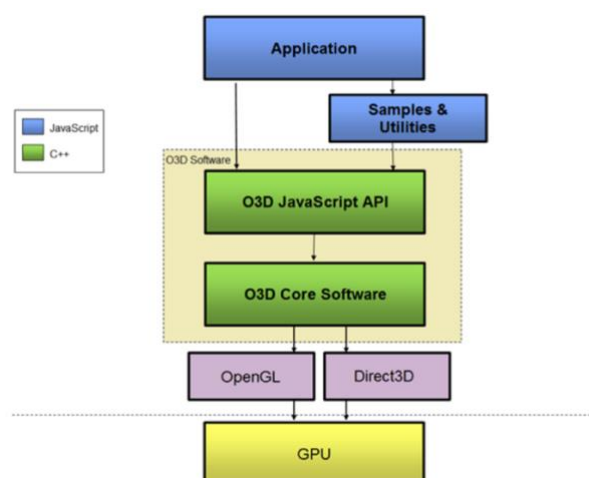
processeur. Cela permet de soulager le micro-processeur et d'améliorer la puissance de calcul pour les rendus 2D et 3D. O3D a été créé de manière à répondre à plusieurs problématiques et peut donc être utilisé dans différents domaines. Les jeux vidéo, les visionneurs de modèles 3D et les applications d'ingénierie faisant partie de ces domaines. Un des avantages de O3D pour en faire un standard est son accessibilité : puisque l'API est en JavaScript, les développeurs Web connaissent déjà bien ce langage. Ce qui permet donc d'éviter aux développeurs de se former à un nouveau langage. Du point de vue technique, O3D fait tous les rendus de manière dynamique et ne nécessite donc pas de ressources externes.

Malgré la volonté de Google de faire de leur solution un standard, le Khronos Group, appuyé par la plupart des géants de l'industrie, s'est imposé avec sa solution Canvas 3D, qui est plus tard devenu WebGL. Afin de ne pas voir son travail rendu inutile, Google a rejoint le Khronos Group pour fusionner O3D et Canvas 3D. Finalement, Google annonça en 2010 que O3D ne sera plus un plug-in mais une librairie dans WebGL.

### 2.3.2 Fonctionnement

Les applications 3D sont produites grâce à l'API écrite en JavaScript qui est parfaitement intégrée au navigateur Web une fois le plug-in installé. Les navigateurs Web disposent d'un moteur JavaScript suffisamment puissant pour coder les applications 3D et ce uniquement dans ce langage. Le cœur de O3D est quant à lui écrit en C++ pour faciliter l'accès au matériel graphique et pour obtenir de très bonnes performances sur les rendus 2D et 3D. L'accès à la carte graphique se fait via différentes API en fonction de l'OS utilisé. Windows utilise l'API Direct3D tandis que MacOS et Linux utilisent OpenGL pour gérer les rendus 2D et 3D via l'accélération matérielle.

Figure 2 : Schéma du fonctionnement de O3D



<https://www.synergeek.fr/google-o3d/>

## 3. Le standard Web3D actuel

### 3.1 Introduction

La 3D étant un enjeu important dans l'évolution du Web, les solutions proposées depuis plus de 20 ans par les géants de l'informatique ont été nombreuses et très variées. Les solutions ont été de mieux en mieux intégrées par les navigateurs et leur utilisation plus poussée. Les possibilités sont passées de simples rendus 3D non interactifs à des jeux vidéo directement exécutables dans un navigateur Web.

Pour arriver à ce résultat, nombreuses ont été les technologies qui ont chacune apporté leurs innovations, mais il subsistait un problème de standardisation avec toutes ces solutions différentes. C'est pourquoi les principaux acteurs de l'informatique ont formé un consortium nommé « Khronos Group » afin de proposer leur solution et d'en faire le standard du Web3D. Ainsi, WebGL a vu le jour en 2011 sous sa version 1.0, en licence open source et a pour but d'exploiter le standard OpenGL. Ce standard a été mis en place par les fabricants de matériel graphique afin d'utiliser l'accélération graphique. WebGL est une API JavaScript qui est compatible avec la majeure partie des navigateurs Web et mobiles et ne nécessite aucun plug-in additionnel. La popularité de WebGL est telle que de nombreux frameworks et bibliothèques ont vu le jour afin de faciliter le développement de projets en WebGL.

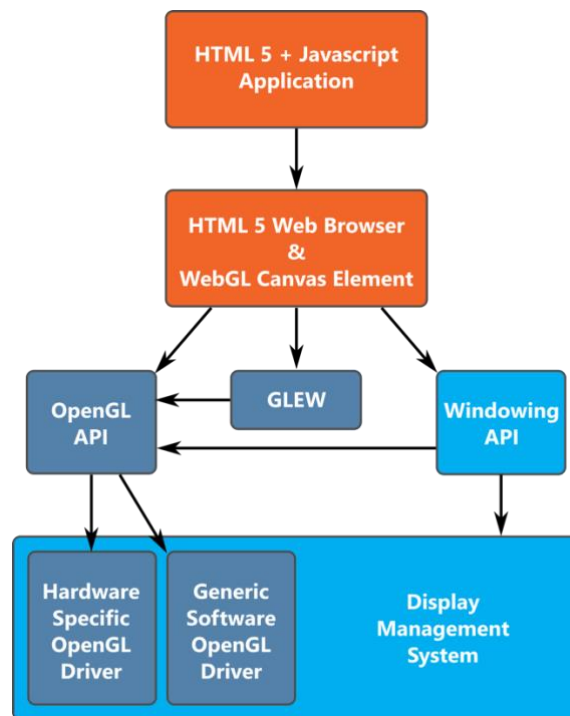
### 3.2 Élément HTML

WebGL fait partie des technologies HTML 5 sans pour autant faire officiellement partie des spécificités HTML 5. Il est destiné aux pages Web et applications pour exploiter l'élément « canvas » proposé par HTML 5 pour réaliser des rendus graphiques 2D ou 3D. L'élément canvas est une balise HTML comme les autres, elle a donc un début et une fin, ce qui permet de faire des rendus graphiques sur l'entièreté de la page Web ou uniquement sur une partie. C'est en partie grâce à l'utilisation de l'élément canvas que propose HTML 5 que WebGL n'a pas besoin de plug-in additionnel pour fonctionner. L'élément canvas est supporté par tous les navigateurs récents mais peut ne pas être supporté par les anciens navigateurs ou les pages Web non HTML 5.

### 3.3 Architecture

L'architecture de WebGL est ici sous sa forme idéale, des variations peuvent être observées au niveau des drivers utilisés par Windows, qui utilise son propre pilote Direct3D au lieu d'OpenGL comme sur MacOS et Linux.

Figure 3 : Schéma de fonctionnement de WebGL



<http://www.cs.uregina.ca/Links/class-info/315/WebGL/Lab1/wip.html>

L'architecture est composée d'une partie « HTML 5 + JavaScript Application », qui est l'application en cours de développement. La partie « HTML 5 Web Browser & WebGL Canvas Element » permet l'utilisation de l'élément canvas pour créer un rendu graphique sur une page Web. La partie « Windowing API » permet de gérer les événements faisant partie du « rendering context » afin de gérer l'initialisation, l'arrêt et les interactions avec la scène OpenGL. La partie « GLEW » est une bibliothèque d'OpenGL qui simplifie la gestion des extensions d'OpenGL. En effet, la gestion des extensions compatibles avec l'ordinateur du client peut être compliquée et GLEW permet, à l'exécution, de charger les extensions adéquates. La partie « OpenGL API » va permettre de créer du contenu graphique sur le navigateur. Les parties « Hardware Specific OpenGL Driver » et « Generic Software OpenGL Driver » sont les drivers dont OpenGL a besoin pour traiter les données. Elles peuvent être traitées de manière classique avec le micro-processeur, ou en combinaison avec la carte graphique pour les opérations plus nécessiteuses en ressources.

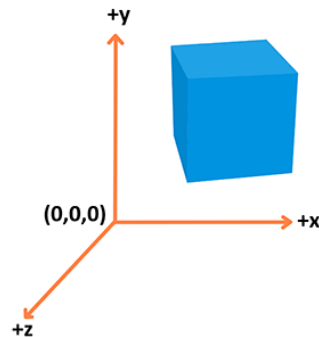
## 3.4 Fonctionnement d'un rendu 3D

### 3.4.1 Système de coordonnées 3D

Pour passer du code JavaScript à une représentation graphique, WebGL utilise un système de coordonnées et de points placés sur celui-ci. Le système varie en fonction

du type de représentation que nous réalisons, si la scène est en 2D il y aura deux axes (x, y) tandis qu'il y aura trois axes (x, y, z) pour une scène en 3D.

Figure 4 : Système de coordonnées 3D



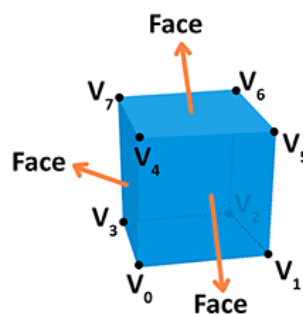
[https://developer.mozilla.org/fr/docs/Games/Techniques/3D\\_on\\_the\\_web/Basic\\_theory](https://developer.mozilla.org/fr/docs/Games/Techniques/3D_on_the_web/Basic_theory)

Les points placés dans ce système de coordonnées sont appelés « vertex », on parle aussi de « vertices » quand il y en a plusieurs. Les vertices possèdent chacun leurs propriétés qui serviront au rendu graphique :

- Position : définit la position du point dans le système de coordonnées 3D.
- Couleur : définit la couleur en valeur RVBA (rouge, vert, bleu, opacité).
- Normal : définit dans quelle direction le vertex fait face.
- Texture : définit une texture 2D qui remplace la couleur de base.

Une fois les vertices reliés, des figures géométriques de bases ou beaucoup plus complexes peuvent être produites. Dans l'exemple suivant, 8 vertices ont été placés et reliés pour donner un cube à 6 faces avec 4 sommets pour chaque face.

Figure 5 : Représentation d'un cube à l'aide de vertices



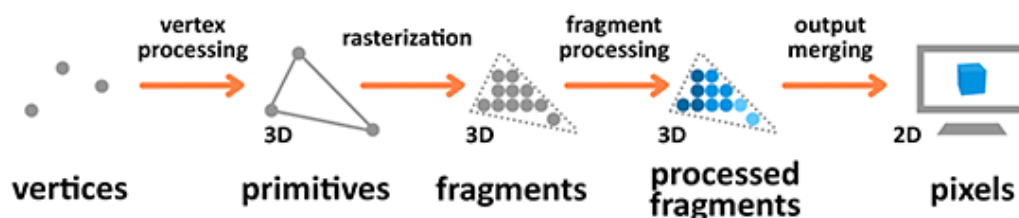
[https://developer.mozilla.org/fr/docs/Games/Techniques/3D\\_on\\_the\\_web/Basic\\_theory](https://developer.mozilla.org/fr/docs/Games/Techniques/3D_on_the_web/Basic_theory)

Les modèles 3D plus complexes qui sont réalisés dans des logiciels de modélisation 3D restent composés de formes géométrique simples qui sont liées entre elles afin de donner le résultat désiré.

Ce modèle 3D complexe est appelé « mesh » en anglais, il est stocké dans une matrice mathématique composée d'un tableau de valeurs qui contient les vertices. Le stockage dans une matrice présente l'avantage de pouvoir déplacer le modèle 3D dans l'espace 3D en une seule instruction plutôt que de changer les vertices un par un. La manipulation des matrices est un exercice qui demande un minimum de notions mathématiques. Heureusement, la majeure partie des bibliothèques 3D permettent de gérer les matrices comme des boîtes noires sans s'occuper du côté mathématique.

### 3.4.2 Étapes de calcul du rendu 3D

Figure 6 : Processus de calcul du rendu 3D



[https://developer.mozilla.org/fr/docs/Games/Techniques/3D\\_on\\_the\\_web/Basic\\_theory](https://developer.mozilla.org/fr/docs/Games/Techniques/3D_on_the_web/Basic_theory)

La première étape consiste à passer de plusieurs vertices à la forme primitive qui est composée des vertices reliés entre eux. Cela permet de former les faces de la figure géométrique et définir leur emplacement dans l'environnement 3D. Cette étape implique aussi le positionnement de la caméra et son orientation.

La deuxième étape permet de passer de la forme primitive à un ensemble de fragments. C'est à cette étape que les formes géométriques produites par des sommets reliés entre eux sont pixélisées pour les afficher sous leur forme la plus simple.

La troisième étape consiste à traiter les textures et lumières qui sont calculées en fonction des paramètres donnés. L'application d'une texture sur une forme en 3D se fait grâce à des images en 2D qui sont combinées entre elles grâce à des pixels voisins. Un système de filtrage est mis en place afin de permettre de gérer la différence de résolution entre la texture et la forme à laquelle elle doit être appliquée en réduisant ou agrandissant la résolution. La lumière agit sur les couleurs de la forme 3D afin de donner l'illusion d'une source de lumière qui peut être réfléchiée ou absorbée sur la surface de la forme 3D.



La dernière étape gère l’affichage à l’écran grâce à la transformation des formes 3D sur une grille de pixels 2D. À cette étape, certaines informations inutiles ne sont pas traitées afin d’améliorer les performances et ainsi, les parties non visibles d’un élément 3D sont ignorées.

## 3.5 Avantages et inconvénients

### 3.5.1 Avantages

Tableau 1 : Avantages de WebGL

Fonctionne sans plug-in	Pas besoin d’un plug-in pour que les rendus 3D fonctionnent. Cela facilite l’usage pour les débutants qui n’ont ni d’installation de plug-in ni de mise à jour de celui-ci à faire.
Basé sur HTML 5	WebGL étant basé sur HTML 5, qui est une norme Web standard, cela contribue à la pérennité du langage dans le temps. Il est également possible d’interagir avec les autres éléments HTML de la page.
Moteur 3D performant	Le moteur 3D embarqué de WebGL permet de créer des applications poussées comme des jeux vidéo. Ceci grâce à OpenGL ES 2.0 qui est spécialement adapté pour les navigateurs Web mobiles et classiques.
Optimisation des ressources	Grâce à l’utilisation de l’accélération matérielle, WebGL permet de soulager le micro-processeur en utilisant la carte graphique, qui permet aussi d’avoir de très bonnes performances 3D.
Simplicité de développement	WebGL est une API en JavaScript, un langage bien connu des développeurs Web. Cela permet de ne pas avoir besoin de se former à un nouveau langage. Beaucoup de bibliothèques et frameworks sont à disposition et simplifient beaucoup le développement.
Open source	L’utilisation de WebGL est gratuite et il est possible d’accéder au code source pour comprendre son fonctionnement.
Compatibilité	WebGL fonctionne sur la plupart des navigateurs Web classiques et mobiles.

### 3.5.2 Inconvénients

Tableau 2 : Inconvénients de WebGL

Basé sur OpenGL ES 2.0	OpenGL ES 2.0 est une version adaptée aux navigateurs mobiles et supports embarqués. Cependant, elle ne dispose pas de toutes les fonctionnalités offertes par OpenGL dû à son adaptation.
Sécurité	WebGL ajoute des fonctionnalités supplémentaires et donc des failles potentielles qui sont critiqués par Microsoft. Les attaques pourraient directement cibler le matériel de la machine, mais un mécanisme de vérification du code est mis en place afin de limiter les attaques. Cela reste cependant une ouverture supplémentaire aux attaques comme le sont les lecteurs vidéo et les bibliothèques de rendus d'images, qui sont exposés au même problème. À chaque ajout de fonctionnalités, l'exposition à de nouvelles failles est inévitable et l'ajout de WebGL ne fait pas exception.
Compatibilité	Certains prérequis sont nécessaires au niveau de la carte graphique pour la prise en charges des pilotes OpenGL. Ainsi les cartes graphiques qui ne supportent pas OpenGL ne pourront pas afficher les rendus 3D.

### 3.6 Les alternatives actuelles à WebGL

Adobe propose sa propre solution nommée Flash Player pour permettre de faire de la 3D sur une page Web. Flash Player est compatible avec une grande partie des navigateurs Web. Cependant, il est nécessaire d'installer un plugin et c'est bien là le désavantage face à WebGL, qui a poussé à l'abandon de cette technologie. Adobe continue à soutenir sa solution car elle n'est pas uniquement utilisée pour le Web3D, mais ce n'est plus une technologie qui est utilisée pour ce genre d'application.

Il existe une autre alternative proposée par Unity qui est principalement un développeur de moteurs pour jeux vidéo. La solution qu'Unity propose fonctionne de la même manière que la solution d'Adobe, c'est-à-dire qu'il faut installer un plugin pour en assurer son bon fonctionnement. Cependant, Unity n'assure plus le développement de leur solution et recommande désormais WebGL pour le même style d'application.

Le constat est plutôt rapide, il existe très peu de technologies qui font concurrence à WebGL. Celles qui étaient sur le marché avant l'arrivée de WebGL, comme Adobe Flash Player, ne font plus partie de la course et sont délaissés au profit de WebGL. WebGL a donc très peu de concurrence sérieuse, mais dans le futur la position de leader de WebGL pourrait bien changer.

### **3.7 Les futures alternatives possibles à WebGL**

WebGL a rapidement été adopté par une grande partie des développeurs Web3D, ses avantages étant divers par rapport à d'autres solutions. Cependant, le fait de devenir un standard demande bien plus que des qualités de performance, de compatibilité et autres. Ce qui a fait de WebGL le standard actuel, est le fait que plusieurs grandes entreprises se sont réunies pour se mettre d'accord pour en faire une solution commune qui convienne à tous. Mais pour rendre cette technologie viable à travers le temps, il fallait surtout penser à l'incorporation de WebGL aux technologies Web actuelles comme l'HTML 5 qui lui, est un standard. Le fait de penser à ces aspects de pérennité et d'adaptation rapide est ce qui a permis son adoption majoritaire par rapport à d'autres solutions.

Malgré ses points forts, le WebGL n'est tout de même pas à l'abri d'autres technologies qui pourraient la remplacer dans un futur proche. Des inconvénients existent au cœur même de WebGL, c'est pourquoi d'autres alternatives pourraient voir le jour et pourraient être de sérieux concurrents.

#### **3.7.1 WebGPU**

WebGPU est une API sur laquelle plusieurs géants comme Apple, W3C, Mozilla, Microsoft et Google travaillent pour créer un nouveau standard de Web3D. WebGPU est une API qui a pour but d'exploiter la puissance des cartes graphiques modernes. Son langage de programmation bas-niveau permet un accès à toutes les fonctionnalités des cartes graphiques pour de meilleures performances. Les performances sont un point important qui est récurrent, mais la raison pour ça est simple. La technologie évolue rapidement et l'architecture des processeurs graphiques a beaucoup changée. D'anciennes technologies comme OpenGL ne sont plus optimisées pour les architectures actuelles. Par exemple, OpenGL ne prend pas en compte les calculs faits en parallèle qui sont apparus avec les processeurs graphiques modernes. Un point négatif quand les performances sont au centre des qualités demandées aux nouvelles API.

Malgré la sortie récente de WebGL 2, cette technologie montre déjà des faiblesses dues à l'utilisation d'OpenGL ES. C'est certes une solution adaptée pour tous les supports,

mais qui reste à un niveau d'abstraction trop haut pour avoir des performances optimales. C'est à cette problématique que WebGPU veut répondre, c'est-à-dire rester à un langage bas-niveau pour être au plus proche du processeur graphique, pour au final améliorer les performances. Le fait d'en faire un langage bas-niveau apporte un autre avantage qui est la portabilité. Ainsi les applications développées en WebGPU fonctionneront sur tous les systèmes d'exploitation sans aucun changement nécessaire. Ce qui n'est pas le cas avec WebGL qui nécessite une architecture différente sur Windows à cause de l'utilisation du pilote Direct 3D de Microsoft au lieu d'OpenGL.

WebGPU est donc une solution prometteuse qui doit encore réussir à s'imposer, comme l'a fait WebGL, mais ses avantages en font une solution qui pourrait bien détrôner WebGL dans le futur.

### **3.7.2 Vulkan**

Vulkan est une API créée par le Khronos Group qui a vu le jour le 16 février 2016 et qui permet de faire des rendus 3D. Cette API est proposée par le même groupe qui a créé le WebGL et elle a pour but de remplacer OpenGL sur le long terme. WebGL n'est donc pour l'instant pas directement menacé, mais plutôt OpenGL. Vulkan se veut donc le successeur d'OpenGL selon le Khronos Group, cependant, il serait possible de voir arriver une version Web de Vulkan. Le fait d'adapter cette technologie bas-niveau au Web représenterait une réelle concurrence à WebGL.

Vulkan a pour but de mieux exploiter les architectures informatiques modernes afin d'offrir de meilleures performances. Voici quelques avantages apportés par Vulkan :

- Meilleure gestion des processeurs multi-cœurs.
- Meilleure adaptation aux plateformes mobiles.
- Simplification du pilote pour permettre au microprocesseur et au processeur graphique une meilleure collaboration.
- Simplification du développement grâce à un langage unifié et compatible d'un constructeur de matériel graphique à un autre.

Vulkan n'est donc pas une menace directe pour WebGL, c'est une technologie récente qui doit encore réussir à remplacer OpenGL avant de pouvoir menacer WebGL. Mais vu ses avantages et le succès des solutions proposées par le Khronos Group, il est fortement possible que Vulkan prenne le dessus sur OpenGL. OpenGL reste tout de même une technologie vieillissante dont les limites sont déjà atteintes. Il ne reste plus qu'à attendre de voir si Vulkan se développera en une version Web. Il reste malgré tout la question de l'avenir de WebGL, qui est également une création du Khronos Group. Le Khronos Group pourrait donc faire une nouvelle version de WebGL qui prendrait en

charge Vulkan ou une nouvelle API pourrait être créée pour utiliser Vulkan et cela signerait sûrement la fin de WebGL à long terme.

### 3.8 Quel est le futur du Web3D ?

Depuis plus de 20 ans, différentes solutions ont permis de faire de la 3D sur le Web un enjeu important pour enrichir l'expérience du Web. La 3D a permis une évolution positive du Web en de nombreux points comme l'augmentation de la créativité. En effet, la 3D permet de casser le schéma que suivent la plupart des sites Web avec des interfaces qui se ressemblent. Par exemple, une entreprise innovante dont le site est un enjeu majeur peut se démarquer grâce à un visuel différent et surtout une expérience qui pousse l'interaction à son maximum. Il est aussi possible de mettre en avant un produit en proposant une interactivité avec celui-ci, comme la visualisation d'un modèle 3D sous tous ses angles. La personnalisation est également un aspect qui permet au client de projeter son imagination et permet à l'entreprise de mettre en avant son produit. Ce point a été bien compris par les constructeurs automobiles qui proposent quasiment tous des configurateurs automobiles 3D.

Figure 7 : Visionneur 3D de Volkswagen



<https://www.volkswagen.ch/app/configurator/vw-ch/fr/la-arteon/31505/36561/r-line/3H74QT-GPJWPJW-GPXZPXZ-GWLHWHLH-GWQ3WQ3-GWS4WS4-GW12W12-GW17W17-GW97W97-GYOZYOZ-GZMFZMF-MEPH7X5-MCDR77G-MSPU79D-MMFA9S8/2019/0/F14%205K5K/F56%20%20%20%20OH/+?page=exterior>

La 3D sur le Web a également permis l'apparition des jeux vidéo qui fonctionnent sur les navigateurs Web. Le nombre de joueurs a beaucoup augmenté et les profils se sont

diversifié grâce à l'apparition de jeux sur smartphones. Il y a donc un marché à conquérir et certaines bibliothèques sont créées exclusivement pour répondre à ce besoin.

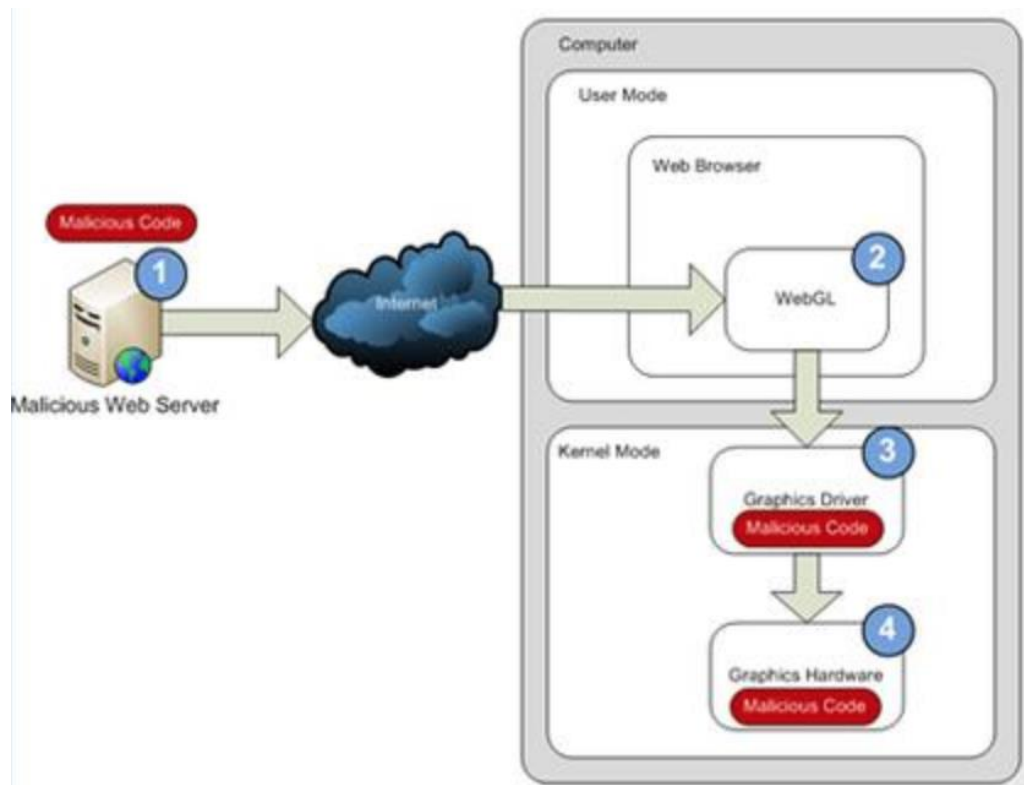
Malgré cette volonté, le constat reste mitigé car il faut avouer que la 3D sur le Web n'est pas très répandue. Auparavant en avance sur son temps, la 3D ne disposait pas d'usages concrets et utiles. Cela a changé maintenant que la technologie est bien implémentée et qu'elle apporte une plus-value à certaines applications. Les usages très poussés de la 3D sur le Web sont donc limités aux entreprises qui y voient une réelle avancée et qui ont surtout les moyens de supporter ces coûts supplémentaires. C'est sûrement un point qui sera déterminant pour l'avenir de cette technologie mais le coût devrait diminuer grâce à différentes technologies. Certaines technologies permettent de réaliser des modèles 3D à partir de photos, ce qui évite de créer le modèle 3D sur des logiciels de modélisation 3D. Il existe aussi des bibliothèques WebGL qui permettent de faciliter le développement et ainsi de réduire le temps de celui-ci.

Le futur de la 3D est assez difficile à prédire. Pour certains usages comme les jeux vidéo et la cartographie 3D, il est nécessaire d'utiliser cette technologie pour offrir une expérience complète et moderne. Pour d'autres usages, la 3D est plus difficile à justifier à cause de ses coûts dus au travail supplémentaire et à la complexification du code que cela engendre. La 3D a donc de beaux jours devant elle et ne risque pas de s'éteindre, mais elle ne sera peut-être pas non plus utilisée partout dans un futur proche.

### **3.9 Sécurité de WebGL**

Différentes technologies ont fait leur apparition sur les navigateurs Web pour enrichir l'expérience des utilisateurs du Web. Les lecteurs vidéo ont par exemple permis de lire des vidéos, chose qui n'a pas toujours été possible. Ces avancées ont permis de répondre à un besoin mais ont nécessité plus de code et l'accès à certaines fonctionnalités avancées afin de fonctionner. Malheureusement, ces avancées ont également amené des vulnérabilités, et des problèmes de sécurité ont rapidement été détectés. WebGL ne fait pas exception, le fait d'utiliser les processeurs graphiques pour le traitement des rendus 3D a amené son lot de complications. Une vulnérabilité a été exposée par Microsoft et l'a poussé à ne pas utiliser WebGL avant d'avoir une solution pour sécuriser le code.

Figure 8 : Fonctionnement d'une attaque via WebGL



<http://igm.univ-mlv.fr/~dr/XPOSE2012/Introduction%20au%20WebGL/inconvenients.html>

Microsoft critique le fonctionnement de WebGL parce que l'utilisation de l'accélération matérielle permet des attaques en accédant directement aux fonctionnalités du matériel de l'ordinateur. L'attaque se déroule en plusieurs étapes :

1. Un visiteur se rend sur un site qui comporte un script WebGL malveillant.
2. WebGL traite le code JavaScript pour le transmettre au pilote graphique qui va faire le rendu 3D.
3. Le code malveillant exploite des failles du pilote graphique pour s'exécuter sur la carte graphique.
4. La carte graphique exécute le code et peut causer un blocage du système voire un arrêt complet.

Cette attaque consiste donc à exploiter les failles d'OpenGL pour accéder à la carte graphique et lui faire exécuter du code infecté. Les navigateurs ne sont pas en cause car la faille se trouve au niveau d'OpenGL, les développeurs des navigateurs ne peuvent donc pas faire grand-chose. En effet, ce sont les constructeurs de cartes graphiques qui prennent en charge la gestion du pilote OpenGL sur leurs produits.

Cette attaque pose problème et ne peut pas être résolue par une simple mise à jour d'OpenGL. Le mécanisme est toujours le même : une fois qu'une faille est trouvée dans OpenGL, cela permet d'injecter du code malveillant. OpenGL aura beau corriger ses

failles actuelles, des nouvelles seront découvertes et le même mécanisme permettant d'exécuter du code infecté sera réutilisé. C'est ce qui a poussé Microsoft à utiliser Direct 3D, son propre pilote qui remplace OpenGL.

Pour contrer ce problème et ne pas être dépendant des constructeurs de matériel graphique, Mozilla a mis en place un système de vérification du code avant son exécution. Le chargement de certains éléments, comme les textures et les ombres qui peuvent également être infectés, sont restreints au domaine dont le code JavaScript provient. Autrement dit, les textures et les ombres doivent être stockées sur le serveur qui a le code WebGL. Ce système de vérification a été adopté par tous les navigateurs et a permis d'écarter cette faille.



## 4. Evolution de l'utilisation de WebGL

### 4.1 Compatibilité des navigateurs d'ordinateur de bureau

L'adoption de WebGL par les navigateurs Web ne s'est pas faite très rapidement et de manière uniforme. Certains navigateurs Web ont pris plus de temps à cause de problèmes de sécurité, comme Microsoft. D'autres ne faisant pas parti du Web Consortium ont adopté cette technologie après que ses fondateurs l'intègrent. Les premiers à intégrer le WebGL ont été Mozilla Firefox, suivis de Google Chrome puis d'Apple avec Safari. Dans l'annexe n°1, un tableau est disponible indiquant les premières versions des navigateurs Web compatibles avec les différentes versions de WebGL. Dans l'annexe n°2, la même tendance peut être observée pour les navigateurs Web mobiles. Les premiers à intégrer cette technologie ont été les fondateurs, puis les autres navigateurs Web ont suivi le mouvement grâce aux avantages que WebGL peut apporter sur mobile.

### 4.2 De WebGL 1.0 à WebGL 2.0

La première version de WebGL a été un succès dans son adoption en tant que standard du Web3D. La majeure partie des navigateurs ont adopté cette nouvelle technologie et son utilisation devient de plus en plus variée. De nombreuses bibliothèques ont été créées afin de simplifier son utilisation. Certaines se sont orientées dans des domaines bien spécifiques, comme la 2D et les jeux vidéo, afin de répondre à un besoin. Le passage vers une nouvelle version a été nécessaire afin de mieux exploiter les possibilités offertes par les cartes graphiques. La version 2.0 permet d'utiliser des techniques de rendu 3D plus poussées et moins gourmandes en ressources, en partie grâce à l'utilisation d'OpenGL ES 3.0. Les nouveautés techniques que la version 2.0 apporte sont diverses comme :

- Le SSAO : permet d'afficher les ombres des objets qui découlent de leur forme, mais surtout en prenant en compte leur relief, ce qui permet un rendu plus réaliste.
- L'instancing : permet d'accélérer le rendu d'objets répétitifs en les représentant sous forme d'une instance unique, ainsi, 1000 instances de 1000 formes identiques sera remplacé par une unique instance.
- Textures 3D : permettent un rendu plus réaliste de certains matériaux comme le bois, le marbre ou tout autre texture avec du relief. Les textures 3D offrent également des effets de fumée plus réalistes.

Les nouveautés apportées par WebGL 2.0 sont nombreuses, comme l'adoption de certaines extensions de WebGL 1.0 pour les intégrer de base avec WebGL 2.0. Les possibilités offertes de base par la version 2.0 sont donc plus importantes. Les navigateurs qui désirent être compatibles WebGL 2.0 doivent désormais intégrer

certaines extensions qui ont pu être mises de côté à cause de certaines failles ou incompatibilités avec le navigateur. Le système d'extensions reste néanmoins en place pour permettre aux différents navigateurs de choisir s'ils veulent intégrer ou non une certaine extension.

La transition à la version 2.0 a été faite par la plupart des bibliothèques et ce passage au nouveau standard ne nécessite aucune modification du code. Si l'application ne présente aucune erreur en version 1.0 elle devrait fonctionner sans modification. Cependant, les applications contenant des erreurs mais qui fonctionnent en version 1.0 peuvent ne plus fonctionner dans la nouvelle version.

Cette nouvelle version est donc la bienvenue et permet de rattraper le retard que le Web3D a pris par rapport à la 3D classique. Malgré ça, il reste des améliorations à faire car certains éléments limitent le WebGL, comme l'utilisation d'OpenGL en version ES.

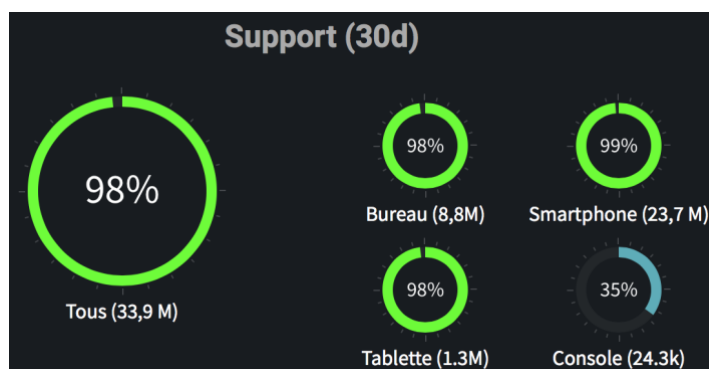
### **4.3 Evolution du taux d'utilisation**

WebGL a connu quelques incidents qui ont freiné l'avancement de cette technologie, tout d'abord, Microsoft a été la dernière grande entreprise à être compatible avec son navigateur Internet Explorer. Cela est dû aux failles de sécurité critiques qui ont été soulevées par Microsoft dans le standard même. Sous sa première version, WebGL a également eu du mal à proposer des fonctionnalités assez poussées pour des applications 3D complexes. Cependant, cela a rapidement été rattrapé par la possibilité d'ajouter des extensions et ainsi adapter WebGL au mieux à son usage personnel.

Les statistiques suivantes viennent de sites qui contribuent volontairement à la collecte des données. Elles ne reflètent donc pas forcément la réalité, mais sont les meilleures statistiques mises à disposition. La récolte de données se fait grâce à une balise que les collaborateurs peuvent mettre dans leur code source.

À ce jour, la version 1.0 est compatible avec 98% des navigateurs Web toutes plateformes confondues et compte presque 44 millions d'utilisateurs. La plupart des utilisateurs utilisent WebGL via smartphone, viennent ensuite les ordinateurs de bureau, les tablettes puis les consoles. Les consoles sont compatibles avec WebGL 1.0 à hauteur de 35%, ce qui explique cette grande différence avec les autres supports est le fait que les consoles possèdent déjà des moteurs 3D plus adaptés et performants.

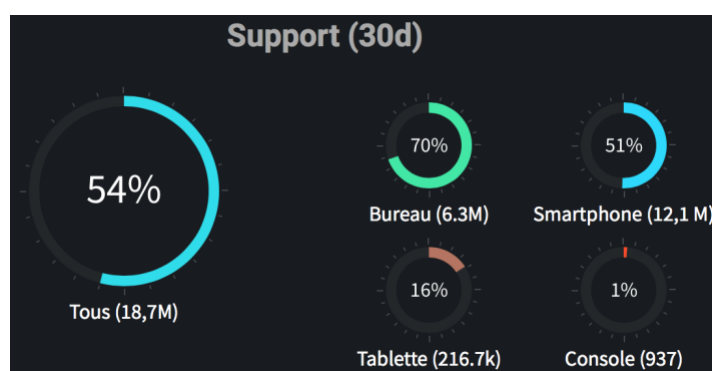
Figure 9 : Pourcentage d'utilisateurs compatibles avec WebGL 1.0



<https://webglstats.com/webgl>

À ce jour, la version 1.0 de WebGL se démocratise rapidement grâce à un passage de la version 1.0 des applications à la version 2.0 sans modification du code nécessaire. Les nouvelles fonctionnalités apportées étaient aussi très attendues et nécessaires ce qui n'a que simplifié la transition à cette nouvelle version. Avec 54% de compatibilité générale et plus de 18 millions d'utilisateurs, le support le plus utilisé reste les smartphones, puis les ordinateurs de bureau, les tablettes et les consoles.

Figure 10 : Pourcentage d'utilisateurs compatibles avec WebGL 2.0



<https://webglstats.com/webgl2>

La nouvelle version connaît un meilleur lancement malgré quelques navigateurs qui se contentent de la version 1.0 avec quelques extensions qui permettent au final de se rapprocher de la version 2.0. Il est cependant intéressant de se demander si ce standard va perdurer. Dans sa dernière version, WebGL garde quelques désavantages qui font de ce standard ce qu'il est, comme l'utilisation du JavaScript qui peut limiter son application. Il est aussi difficile de trouver des cas où la 3D sur le Web est réellement nécessaire, même si cette technologie tend à être de plus en plus utilisée dans les domaines de la vente en ligne, de la visualisation de données scientifique, des jeux vidéo et d'autres domaines.

## 5. Les bibliothèques WebGL

### 5.1 Pourquoi faire des bibliothèques ?

WebGL est certes une technologie qui est simple à aborder grâce à l'utilisation du langage JavaScript. Mais la courbe de difficulté n'en reste pas moins abrupte à cause des notions spécifiques qu'il faut apprendre. Les applications 3D plus complexes demandent des notions variées en graphisme : il faut comprendre le fonctionnement des ombres, textures, formes et autres composants d'un environnement 3D. WebGL demande beaucoup de code pour réaliser des rendus basiques. Parmi les fondamentaux de WebGL qui consistent, par exemple, à afficher un triangle rose, il faut compter pas moins de 80 lignes de code.

Figure 11 : Exemple basique de code WebGL

```
81 // Turn on the attribute
82 gl.enableVertexAttribArray(positionAttributeLocation);
83
84 // Bind the position buffer.
85 gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
86
87 // Tell the attribute how to get data out of positionBuffer (ARRAY_BUFFER)
88 var size = 2; // 2 components per iteration
89 var type = gl.FLOAT; // the data is 32bit floats
90 var normalize = false; // don't normalize the data
91 var stride = 0; // 0 = move forward size * sizeof(type) each iteration
92 var offset = 0; // start at the beginning of the buffer
93 gl.vertexAttribPointer(
94     positionAttributeLocation, size, type, normalize, stride, offset);
95
96 // draw
97 var primitiveType = gl.TRIANGLES;
98 var offset = 0;
99 var count = 3;
100 gl.drawArrays(primitiveType, offset, count);
101 }
102
103 main();
```



<https://webglfundamentals.org/webgl/lessons/webgl-fundamentals.html>

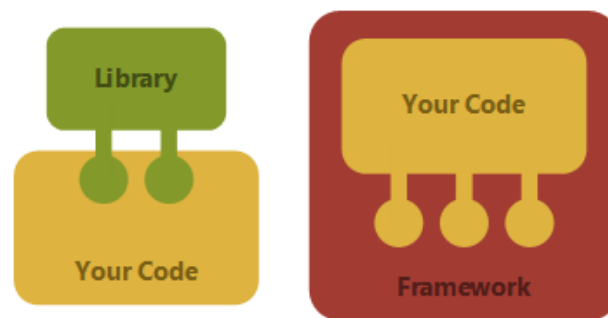
C'est en partie à cause de cette complexité que de nombreuses bibliothèques ont vu le jour et que WebGL est rarement utilisé directement, mais plutôt via des bibliothèques. La simplification du code ainsi que la réduction de sa taille étaient les objectifs principaux mais il fallait aussi orienter certaines bibliothèques pour répondre à une problématique précise.

Les bibliothèques peuvent être divisées en deux catégories principales : les bibliothèques 2D et les bibliothèques 3D. Parmi ces deux catégories il existe de nombreuses sous catégories. Pour les bibliothèques 2D, nous allons retrouver des bibliothèques destinées aux livres numériques, diaporamas, publicités interactives, images dynamiques... Les bibliothèques 3D, répondent quant à elles à d'autres problématiques comme la cartographie, la visualisation de données scientifiques ou les jeux vidéo...

## 5.2 Différences entre une bibliothèque et un framework

Afin de différencier les librairies et frameworks présentés par la suite, il est important de faire la différence entre une librairie et un framework. Ces deux éléments ont le même but, simplifier le développement en mettant à disposition des fonctionnalités pour résoudre des problèmes récurrents. Le code des librairies et frameworks est développé par une personne tierce qui met à disposition des fonctionnalités qui sont généralisées pour pouvoir être appliquées à un maximum de cas. La différence entre une librairie et un framework se situe dans son utilisation. Une librairie vient s'ajouter au code et il est possible de faire appel à ses fonctionnalités à n'importe quel moment dans le code. Un framework est comme un squelette où le code doit être inséré à des endroits précis. Il permet d'avoir des fonctionnalités plus poussées mais impose plus de contraintes lors du développement. Contrairement à une librairie, ce n'est plus notre propre code qui est au centre de l'application mais le code du framework.

Figure 12 : Différences entre une librairie et un framework



<http://tomasp.net/blog/2015/library-frameworks/>

Chacune de ces solutions a ses avantages et ses inconvénients. Les librairies permettent une plus grande liberté en gardant le code au centre du développement tout en apportant des fonctionnalités utiles que le développeur n'a pas besoin de coder. Les désavantages des librairies sont que celles-ci offrent des solutions à des problèmes récurrents plutôt que des fonctionnalités poussées. Il est également facile de se retrouver rapidement avec beaucoup d'appels à différentes librairies. Cela peut nuire à la transparence du code qui fait appel à plusieurs fonctionnalités. Ces fonctionnalités agissent comme des boîtes noires pour produire leur résultat.

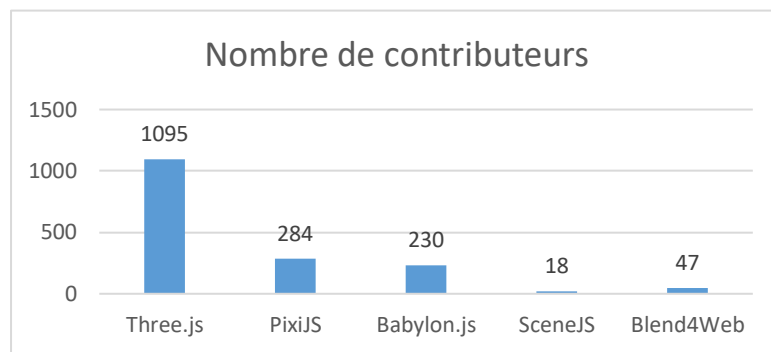
Les frameworks permettent un développement de fonctionnalités plus poussées mais imposent leurs propres règles de codage. Cela pose problème au niveau de la liberté offerte au développeur. Comme indiqué plus haut, notre code n'est plus au centre du développement, il laisse sa place au framework. Il reste possible d'apporter ses propres fonctionnalités au framework mais cela reste une tâche difficile, car le framework

n'indique pas forcément clairement ses valeurs d'entrées et de sorties. Cela peut provoquer des erreurs difficiles à résoudre.

### 5.3 Les bibliothèques/frameworks les plus utilisées

Les bibliothèques et frameworks existants basés sur WebGL sont très nombreux, en effet il existe beaucoup de différents domaines où la 3D peut être intéressante. Parmi ces solutions, certaines sont populaires grâce à leur stabilité dans le temps et d'autres grâce à leur apport en innovation. Il n'existe pas de statistiques sur la popularité de chaque bibliothèque ou framework, le graphique suivant a donc été basé sur le nombre de contributeurs de ces solutions sur GitHub. GitHub est un site très populaire pour l'hébergement et la gestion de développement de logiciels, c'est pourquoi j'ai décidé de me fier à ses résultats. Cette donnée n'est donc pas exacte mais reflète en partie la popularité des solutions.

Figure 13 : Popularité des bibliothèques et frameworks



Grâce à ces chiffres, on constate que les bibliothèques et frameworks les plus populaires sont ceux qui existent depuis plusieurs années et qui ont donc prouvé leur stabilité et leur évolutivité. La quantité de solutions est telle que le renouvellement et l'assimilation des nouvelles fonctionnalités offertes par WebGL est un point très important. Un autre facteur qui définit la popularité d'une solution est son orientation. Selon les chiffres exposés au-dessus, Three.js est la bibliothèque la plus populaire car elle convient à la majeure partie des applications 3D. Les autres bibliothèques et frameworks sont nettement moins utilisés que Three.js. Cela peut s'expliquer par leur orientation qui est plus spécifique ou par leur fonctionnement qui diffère de la plupart des solutions. PixiJS est principalement destiné aux applications 2D, BabyLon.js est conçu pour les jeux vidéo. SceneJS est conçu pour des domaines techniques et médicaux et Blend4Web nécessite l'utilisation d'un logiciel de modélisation 3D spécifique. Ces solutions sont utilisées quand la solution la plus populaire ne convient pas, cela explique donc le nombre nettement moins important de contributeurs pour les autres solutions. D'autres facteurs comme une

pauvre documentation en ligne ou une communauté très restreinte peuvent très bien définir le succès ou non d'une solution.

## 5.4 Les bibliothèques 2D

Les bibliothèques 2D sont moins présentes que les bibliothèques 3D car il y a moins d'utilité à faire des rendus 2D. Elles peuvent néanmoins s'appliquer à un champ d'utilisation très vaste lorsque l'on fait preuve d'un peu de créativité. Les bibliothèques 2D peuvent être utilisées pour des applications plus conventionnelles comme les jeux vidéo 2D ou encore la cartographie 2D. Cependant, des développeurs utilisent ces bibliothèques pour enrichir l'expérience des utilisateurs en proposant une approche différente pour naviguer sur un site Web. Les interactions avancées avec tous les éléments à l'écran ainsi que des effets visuels permettent de laisser sa créativité s'exprimer. Le résultat ne répond pas toujours à des besoins métiers, mais il peut réellement mettre en avant une idée, un service ou encore un produit. Parmi ces expériences innovantes sur le Web, la chaîne télévisée BBC propose sur son site une nouvelle interactive, disponible via le lien suivant : <http://www.bbc.co.uk/guides/z3b77hv>

Cet exemple n'est qu'une des diverses manières d'utiliser les bibliothèques 2D de manière créative. L'utilisation d'une bibliothèque 2D a permis de rendre cette bande dessinée interactive, ainsi, les animations défilent en fonction de interactions de l'utilisateur. Des éléments sonores sont également déclenchés en fonction des interactions et un menu est disponible pour régler certains paramètres. Ce genre de réalisation pourrait être réalisé avec un simple navigateur de photo, cependant l'expérience en serait réduite avec des éléments sonores absents et des transitions inexistantes.

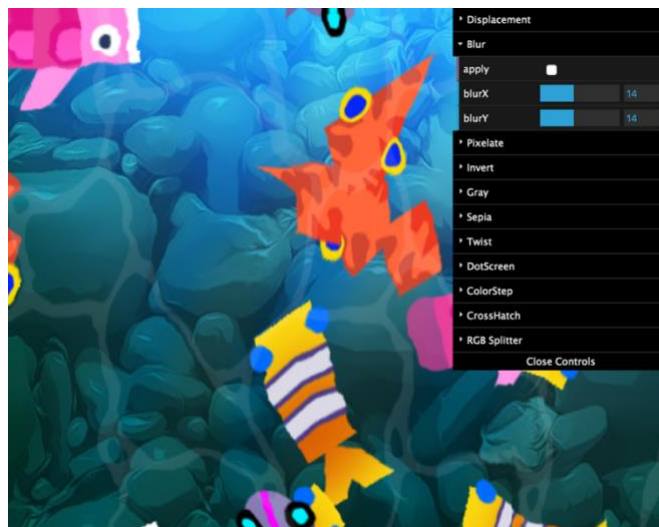
### 5.4.1 PixiJS

PixiJS est une bibliothèque WebGL open source qui a vu le jour le 21 janvier 2013. Elle est la bibliothèque de référence pour faire des rendus 2D dans un navigateur Web via l'utilisation de JavaScript. C'est une bibliothèque de milieu de niveau, c'est-à-dire qu'elle se situe entre le bas-niveau, où quasiment tout est possible mais où il faut aussi gérer plusieurs aspects techniques. Et entre le haut-niveau, où tout est déjà géré automatiquement mais où le développeur est aussi restreint à utiliser les fonctionnalités qui lui sont proposées. En tirant le meilleur de ces deux mondes, PixiJS permet aux développeurs d'avoir plus de liberté. Sans pour autant avoir à gérer des aspects techniques fastidieux comme la gestion de la mémoire.

Cette bibliothèque a comme autres avantages d'être très rapide et de consommer peu de ressources, c'est pourquoi elle est souvent utilisée pour des jeux vidéo 2D.

Cependant, elle n'est pas prévue à cet effet car elle ne dispose ni d'un moteur physique, ni d'intelligence artificielle, ni de système de son. Tous ces éléments doivent être ajoutés via des extensions qui sont ensuite parfaitement intégrées. PixiJS embarque néanmoins des fonctionnalités très utiles comme le système de filtre. Les rendus faits à base de simples éléments graphiques, comme des images, peuvent être modifiés à volonté. Le système de filtre permet de manipuler différents éléments du rendu comme le flou, la pixellisation ou encore la déformation.

Figure 14 : Différents filtres de PixiJS



<https://www.goodboydigital.com/pixijs/examples/15/indexAll.html>

Ce rendu d'un aquarium permet d'essayer les différents filtres tout en regardant le résultat en direct. Peu importe le nombre d'effets appliqués, les performances restent stables. En apparence, ce système de filtres peut paraître très basique, mais il ne faut pas oublier qu'à la base tous les éléments à l'écran sont en 2D. Les animations et les effets sont uniquement gérés par PixiJS !

PixiJS est donc une des bibliothèques les plus complètes et plus utilisées pour les rendus 2D. Ses performances, sa simplicité et son extensibilité en font une bibliothèque adéquate pour différents usages variés et son succès reste grandissant.

## 5.5 Les bibliothèques/frameworks 3D

Les solutions 3D sont nettement plus nombreuses que les solutions 2D et ce n'est pas un hasard. Les domaines applicatifs sont bien plus nombreux et la demande de ce genre de solution ne fait qu'évoluer.

Les domaines applicatifs sont beaucoup plus variés, tellement variés que certaines bibliothèques/frameworks se destinent à une utilisation bien particulière afin de répondre



à une problématique précise. Certaines bibliothèques comme BabylonJS se destinent à développer des jeux vidéo. D'autres comme SceneJS sont conçues pour être utilisées dans des domaines scientifiques ou bien d'ingénierie où la visualisation de modèles 3D complexes est requise. Il existe également d'autres bibliothèques comme Three.js qui sont très flexibles et qui permettent de répondre à la plupart des problématiques. Le choix d'une solution ou d'une autre est plutôt compliqué. Malgré l'orientation de certaines solutions, cela n'implique pas qu'elles ne puissent pas répondre à d'autres problématiques. Il faut donc prendre en compte les fonctionnalités offertes par les solutions lors du choix de celle-ci. D'autres arguments plus décisifs peuvent également être la documentation en ligne mise à disposition ou l'importance de la communauté. Ces atouts peuvent permettre de faciliter la résolution d'un problème durant le développement.

### **5.5.1 Three.js**

Three.js est une bibliothèque WebGL qui permet de faire des rendus 3D. La première version sortie le 24 avril 2010 continue à être actualisée depuis près de 10 ans. Sa longévité ne fait qu'appuyer le succès de cette bibliothèque, qui est une des plus connue et utilisée. Ce succès est aussi dû à une documentation en ligne très complète, avec des tutoriels qui expliquent comment utiliser les fonctionnalités mises à disposition. Des exemples visuels sont également disponibles afin de démontrer la versatilité de Three.js, qui se prête en effet à énormément d'usages différents. Une importante communauté extrêmement active est également au cœur du succès de cette bibliothèque. Chaque problème rencontré peut être résolu sur des forums où des personnes collaborent entre elles pour trouver des solutions ou encore pour indiquer des bugs.

Three.js fait partie des bibliothèques de haut-niveau, il n'est donc pas nécessaire de gérer les aspects complexes des rendus 3D. Plusieurs fonctionnalités sont mises à disposition du développeur afin de rapidement produire des applications 3D. Sa facilité d'apprentissage est un de ses atouts et permet de rendre facile l'utilisation de WebGL qui lui demande bien plus d'efforts lors de sa prise en main. Cette bibliothèque n'a pas d'orientation particulière, grâce à ses différentes fonctionnalités elle permet de réaliser beaucoup d'applications différentes. Cependant, d'autres bibliothèques restent plus adaptées à certaines utilisations comme Babylon.js pour la création de jeux vidéo. Three.js est principalement optimisé pour faire des rendus 3D mais permet également de faire des rendus 2D. Cependant, comme abordé avec la bibliothèque précédente, certaines bibliothèques comme PixiJS sont mieux adaptées aux rendus 2D. Voici quelques-unes des différentes fonctionnalités proposées par Three.js :

- Gestion des caméras, lumières, matériaux, ombres.
- Modules de chargement des modèles 3D.
- Système de gestion des particules.
- Système d'animation des modèles.
- Outils mathématiques pour simplifier la gestion de l'environnement 3D.

Le but principal de Three.js est de rendre le développement des applications 3D facile et rapide et cet objectif a bien été rempli. Alors que WebGL nécessite plus de 80 lignes de code pour afficher un triangle de couleur unie sans aucune animation. Three.js ne nécessite pas plus de 20 lignes de code pour afficher un cube de plusieurs couleurs qui est animé avec un rotation sur tous ses axes.

Figure 15 : Exemple de code basique de Three.js

```

<script src="js/three.js"></script>
<script>
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera( 75, window.innerWidth/window.innerHeight, 0.1, 1000 );

var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );

var geometry = new THREE.BoxGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );

camera.position.z = 5;


var animate = function () {
  requestAnimationFrame( animate );

  cube.rotation.x += 0.01;
  cube.rotation.y += 0.01;

  renderer.render( scene, camera );
};

animate();
</script>

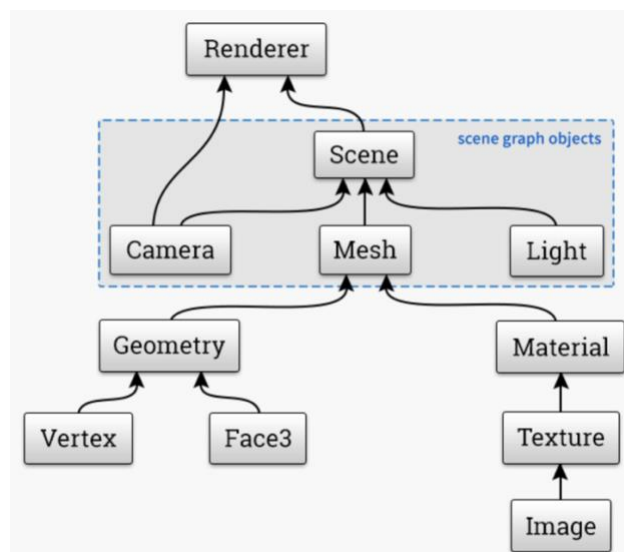
```



<https://threejs.org/docs/#manual/en/introduction/Creating-a-scene>

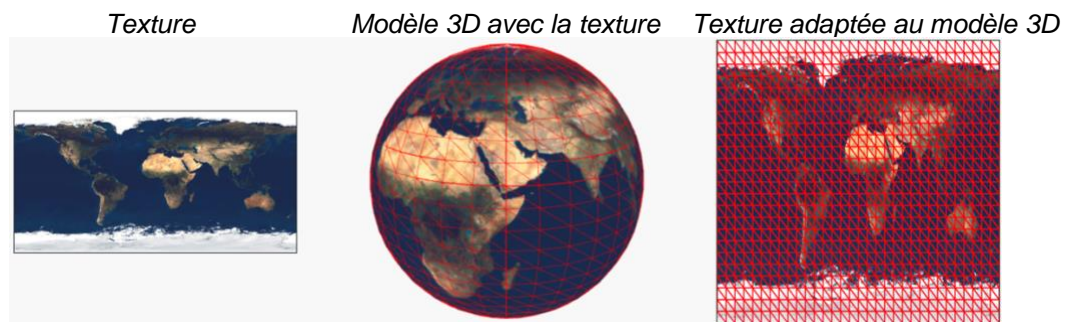
Cette simplicité de développement est en partie due à la structure d'un rendu 3D Three.js. En effet, la structure du rendu 3D fonctionne sous forme d'arborescence.

Figure 16 : Structure d'une rendu 3D Three.js



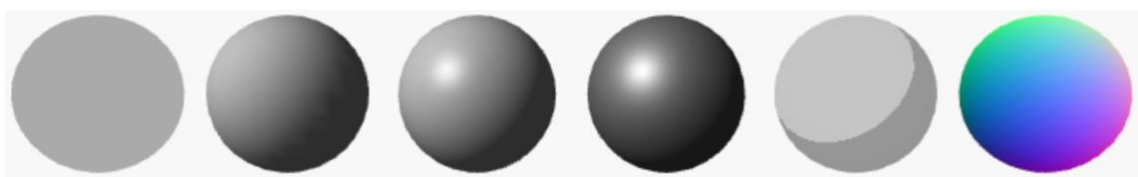
Le rendu (renderer) est l'application 3D qui est développée, elle possède 2 nœuds enfants qui sont la scène (scene) et la caméra (camera). La scène est le conteneur de tous les objets 3D qui composent le rendu 3D. La caméra quant à elle, fait partie du rendu et de la scène afin qu'elle puisse éventuellement être fixée sur un objet. La lumière (light) permet quant à elle de définir une ou plusieurs sources de lumières qui sont réfléchies par les objets. Pour fonctionner, le minimum que doit contenir un rendu 3D est une scène et une caméra. Afin de produire un rendu 3D avec du contenu visuel il est possible d'ajouter des formes géométriques (meshes) composées de vertices et de faces. Afin de donner un aspect aux formes géométriques, il est possible de leur ajouter une texture basée sur une image. Cette dernière sera déformée ou répétée pour ensuite être appliquée à l'objet 3D.

Figure 17 : Application d'une texture à base d'une image



Il est également possible d'utiliser un matériel qui est en fait une texture basique et paramétrable prédéfinie dans Three.js. Voici tous les matériaux proposés par Three.js :

Figure 18 : Matériaux de base disponibles



Ces matériaux possèdent plusieurs paramètres modulables qui permettent de personnaliser le rendu des objets :

- Brillance : Définit le degré de réflexion de la lumière.
- Métal : Définit le degré de l'aspect métallique.
- Aspérité : Définit le degré d'aspérité ou de régularité.
- Ombrage : Définit l'intensité des ombres.

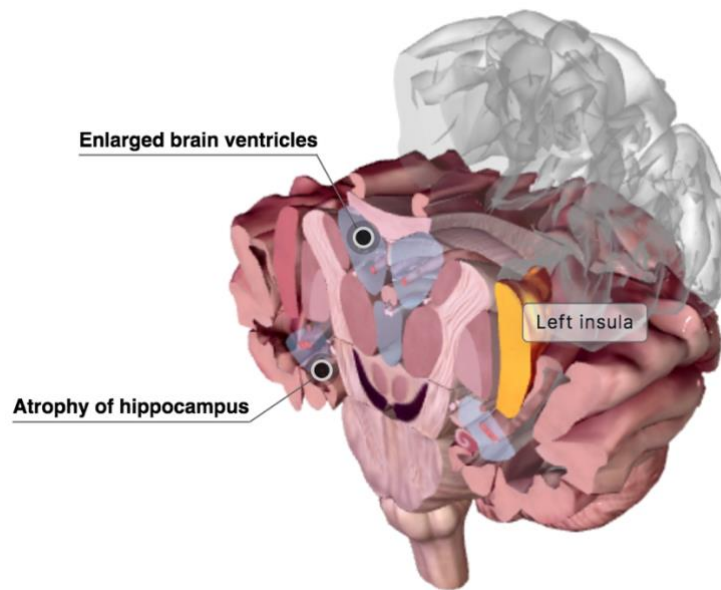
Three.js est une bibliothèque qui propose beaucoup de fonctionnalités tout en simplifiant le développement d'applications 3D par rapport à WebGL. Three.js est un peu le couteau Suisse des bibliothèques WebGL, elle peut tout faire sans être spécialisée dans un domaine spécifique. C'est donc un choix très prisé pour les développeurs Web3D et son avenir ne devrait pas être mis en danger tant que WebGL reste dans la course du Web3D.

### **5.5.2 SceneJS**

SceneJS est une bibliothèque open source basée sur WebGL, elle a été créée en 2009 et est aujourd'hui à sa version 4.2. SceneJS a une communauté plus réduite mais dispose de nombreux exemples qui mettent en œuvre les fonctionnalités proposées. Comme la plupart des bibliothèques de haut-niveau, SceneJS permet de simplifier l'utilisation de WebGL. Mais ce qui démarque cette bibliothèque des autres est le fait que SceneJS se destine surtout à être un visionneur de modèles 3D complexes. C'est une bibliothèque qui répond aux besoins de domaines spécifiques comme la médecine, l'architecture, l'ingénierie mécanique et tout autre domaine où des modèles 3D détaillés existent. Cette bibliothèque est donc optimisée pour produire des rendus 3D très rapidement et gérer un grand nombre d'objets affichés à l'écran. Elle dispose aussi d'une fonctionnalité qui permet de pouvoir sélectionner un objet spécifique du modèle 3D afin d'analyser celui-ci en détail. Dû à cette orientation, cette bibliothèque ne dispose pas de certaines fonctionnalités comme les réflexions de lumière dynamique, les effets d'ombres dynamique ou autres. SceneJS est donc une bibliothèque moins flexible que d'autres, mais elle n'en reste pas moins une bonne solution pour beaucoup de domaines professionnels.

Les fonctionnalités de SceneJS sont très bien illustrées par Biodigital, qui sont spécialisés dans le visionnage de modèles 3D dans le domaine de la médecine. Plusieurs modèles de l'anatomie humaine sont mis à disposition et montrent les fonctionnalités de SceneJS. Les modèles possèdent beaucoup d'éléments et chaque élément du modèle 3D est interactif. L'utilisation de ses fonctionnalités permet de sélectionner un élément pour l'isoler, le mettre en transparence, afficher ses informations ou encore de ne pas l'afficher.

Figure 19 : Modèle 3D interactif d'un cerveau



[https://human.biodigital.com/view?id=production%2FmaleAdult%2Falzheimers\\_disease\\_brain\\_cross\\_section&type=module](https://human.biodigital.com/view?id=production%2FmaleAdult%2Falzheimers_disease_brain_cross_section&type=module)

Comme les autres bibliothèques, SceneJS dispose d'un « scene graph » qui s'occupe de l'arrangement spatial et du rendu du modèle 3D. C'est en fait une structure de données qui contient tous les nœuds du modèle 3D et qui place ses points dans l'espace. La spécificité du « scene graph » dans SceneJS est qu'il est basé sur une structure qui s'inspire du format JSON. Il y a donc une notion d'arborescence, de nœuds parents et enfants mais aussi d'héritage entre les nœuds.

Figure 20 : Exemple de code SceneJS

```
SceneJS.createNode({
  type: "scene",
  id: "my-scene",
  canvasId: "the-canvas",

  nodes: [
    { type: "lookAt", id: "the-lookat",
      eye: { x: -1.0, y: 0.0, z: 15 } ...

      nodes: [
        { type: "camera", optics: { fovy: 55.0 ... },

          nodes: [
            { type: "light", color: {r:1.0, g:1.0, b:1.0}},

            nodes: [ { type: "teapot" } ...
```



<https://www.slideshare.net/lindsayk/scenejs-6952843>

Comme expliqué plus haut, SceneJS est une API qui s'inspire du format JSON, elle est donc bien différente de Three.js. L'API de SceneJS offre une approche différente et est plus basé sur les données 3D et leur structure, tandis que Three.js se base plus sur une programmation orienté objet. On retrouve donc les principes fondamentaux de JSON, les éléments 3D sont des balises qui possèdent leurs attributs. On retrouve également un système d'arborescence qui définit les relations de parents et d'enfants entre les éléments.

SceneJS est donc une bibliothèque qui excelle dans un domaine très précis qui est la représentation de modèles 3D complexes. Son API offre une approche différente qui peut permettre une meilleure compréhension du modèle 3D. Sa longévité n'est plus à prouver malgré une communauté plus petite que d'autre bibliothèques.

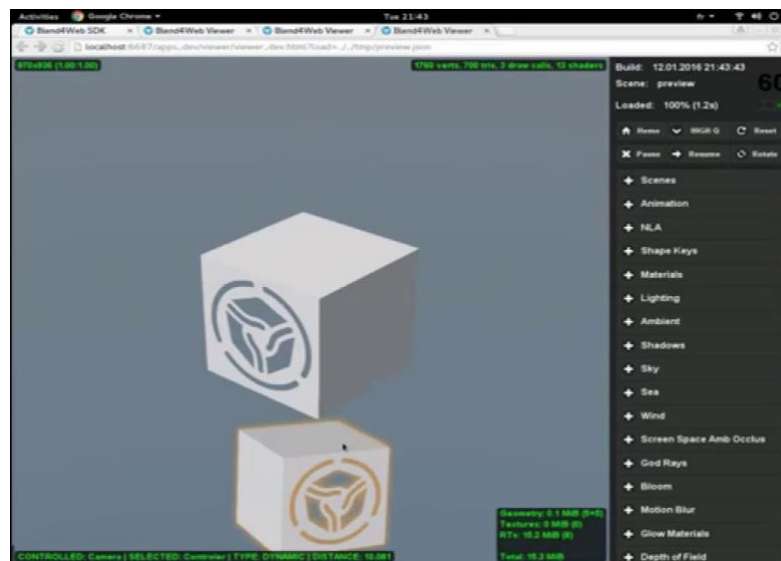
### **5.5.3 Blend4Web**

Blend4Web est un framework à double licence développé par Triumph LLC et qui est sorti le 28 mars 2014. Sa dernière version date du 31 août 2017 mais le projet n'en reste pas moins actif. Le framework est disponible en licence open source et via une licence commerciale. Le choix de la licence varie si le développeur utilise la version standard du « web player » mis à disposition qui permet de visionner ses créations sur un navigateur Web ou s'il utilise un visionneur sur mesure. Ainsi, si le projet utilise la version standard du visionneur, la licence est gratuite et les fichiers sources peuvent être gardés privés. Si un projet utilise un visionneur sur mesure, les fichiers sources doivent être partagés afin d'utiliser la licence gratuite. En revanche, si un visionneur sur mesure est utilisé mais que le développeur ne désire pas partager les fichiers sources il faudra utiliser une licence payante.

Ce framework se base sur WebGL et permet de créer des modèles 3D interactifs dans un navigateur Web en s'appuyant sur l'éditeur 3D Blender. Il est possible de créer des modèles 3D facilement grâce à l'éditeur Blender puis d'exporter le modèle pour l'utiliser dans une application Web. Plusieurs outils sont mis à disposition des développeurs pour simplifier l'utilisation de Blender et Blend4Web. Par exemple, un plugin est disponible sur Blender pour définir un profil de projet, qui limite les fonctionnalités de Blender à des fonctionnalités qui sont pertinentes sur Blend4Web. Blend4Web possède également une interface graphique qui permet de gérer les ombres, les particules et autres points qui, sans cet outil, devraient être fait sur Blender. Un système d'animation de base est aussi présent dans Blend4Web ce qui permet de se passer en partie de scripts pour des réalisations basiques. Blender étant un outil de conception 3D très poussé, il propose beaucoup de fonctionnalités afin de supporter les projets les plus ambitieux. Par

exemple, un moteur de jeu est proposé et embarque un moteur physique, un système d'effets audio 3D et un système d'animations. Des outils de création ont également été mis à disposition, ainsi, il est possible de créer rapidement un environnement extérieur avec de la végétation, de l'eau, des effets de lumière et d'atmosphère. Les performances des rendus sur les navigateurs Web ont également été un point important. Pour des questions d'optimisation des solutions ont été mises en place. Parmi ces solutions, la détection de surfaces cachées a été mise en place afin de ne pas calculer le rendu du modèle 3D qui est caché par un objet ou tout simplement non visible à l'écran.

Figure 21 : Interface graphique de Blend4Web



<https://i.ytimg.com/vi/N-ByVjv9Edo/maxresdefault.jpg>

Pour passer de Blender à Blend4Web il faut simplement exporter son modèle 3D au format JSON ou HTML. En exportant le modèle 3D au format JSON, seul le modèle sera présent dans le fichier. Cela permet d'avoir un fichier léger et qui contient uniquement l'essentiel des informations du modèle 3D et est donc le format le plus optimisé pour des questions de performances. En revanche, l'utilisation du format HTML possède des avantages, comme le fait de pouvoir visionner le résultat en ligne et hors ligne. Le format HTML intègre le modèle, le visionneur standard et d'autres éléments qui sont compressés dans le fichier et qui permettent le fonctionnement du visionneur hors ligne.

Contrairement aux bibliothèques vues précédemment, le développement de scènes 3D se fait en grande partie sur Blender. En effet, Blender possède une interface graphique très complète qui simplifie grandement la création de modèle 3D. Des formes basiques ou complexes peuvent y être créées, les autres éléments comme la caméra, la lumière, la texture sont également gérés directement sur Blender. Au final, Blend4Web est principalement un visionneur de modèles 3D qui permet tout de même d'apporter des

fonctionnalités supplémentaires. La gestion des interactions, des contrôles et des animations se fera par exemple directement par le biais de JavaScript. Dans la plupart des cas, les possibilités offertes par Blender suffisent pour produire une scène 3D de base et ne pas avoir besoin de coder. À l'exception que la solution développée soit assez spécifique.

On constate donc que Blend4Web propose une approche légèrement différente pour créer une scène 3D. Une grande partie du développement se passe sur Blender, leur logiciel de modélisation 3D, puis des outils sont proposés pour faire communiquer les deux solutions et simplifier le développement. L'approche avec les bibliothèques vues précédemment est quasiment la même. Les modèles sont créés sur des logiciels de modélisations 3D, puis ils sont exportés pour être utilisés et manipulés via les fonctionnalités de la bibliothèque. Avec Blend4Web, le fait d'utiliser des logiciels de la même famille permet de simplifier la communication entre le logiciel de modélisation 3D et celui qui va s'occuper de l'affichage sur le navigateur Web. Blend4Web dispose d'outils de modification du modèle directement sur le navigateur. Cela permet d'éviter de faire ses changements directement sur le modèle utilisé dans le navigateur, sans devoir retourner sur le logiciel de modélisation ou faire ça de manière programmatique. Blender offre donc de nombreuses possibilités, la limitation de cette solution se trouve au niveau de Blend4Web qui propose un visionneur Web qui est certes complet, mais qui peut ne pas correspondre à toutes les applications. Dans certains cas, il sera nécessaire d'opter pour une licence payante pour pouvoir développer sa propre solution.

#### **5.5.4 BabylonJS**

BabylonJS est une bibliothèque JavaScript créée en 2013 par deux employés de Microsoft et est sous licence Apache 2.0. Basé sur WebGL, cette bibliothèque est écrite en TypeScript puis compilée en JavaScript afin d'utiliser ce langage pour interagir avec l'API. Elle est principalement conçue pour créer des jeux vidéo sur des navigateurs Web, mais peut être utilisée à d'autres fins. Elle embarque donc beaucoup de technologies comme WebVR, WebAudio et bien d'autres qui permettent d'enrichir l'expérience vidéoludique. BabylonJS a été développé pour être principalement un moteur 3D de jeux vidéo, c'est tout naturellement que cette bibliothèque est une bibliothèque de bas-niveau. Cela permet d'offrir un maximum de liberté aux développeurs et d'optimiser les performances. Les jeux vidéo sont de grands consommateurs de ressources graphiques, il est donc nécessaire de permettre un accès direct aux ressources de la machine. Malgré son statut de bibliothèque de bas-niveau, BabylonJS reste une bibliothèque qui simplifie grandement la création de jeux vidéo. L'utilisation de WebGL et du JavaScript permet donc de simplifier le développement grâce à des classes et

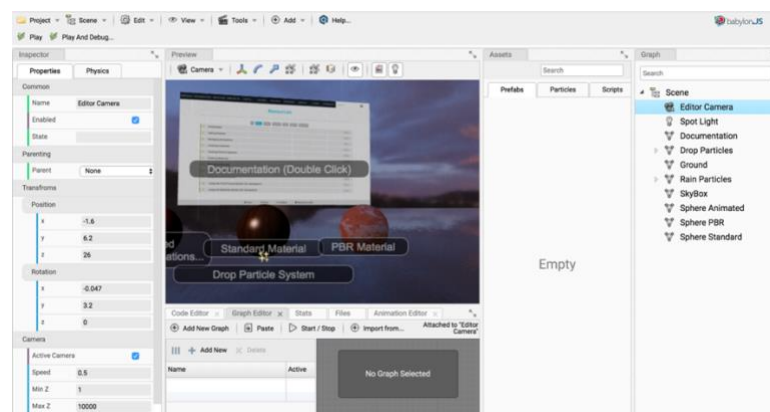


méthodes mises à disposition afin de contrer les désavantages d'une bibliothèque bas-niveau. Afin de proposer tous les outils nécessaires à la création d'un jeu vidéo, BabylonJS intègre toutes les fonctionnalités de base. Les caméras, l'éclairage dynamique, l'importation de modèles 3D, le multi-texturage et l'animation de modèles 3D font partie de ces fonctionnalités. Les autres fonctionnalités que cette bibliothèque apporte par rapport à d'autres concurrents sont :

- La gestion des collisions.
- Un moteur physique.
- La gestion de la gravité.
- Un système de particules.

Le moteur physique n'est pas directement intégré au moteur 3D de cette bibliothèque car cela permet de garder un moteur 3D plus léger. La physique des objets 3D n'est pas forcément utilisée dans toutes les applications, elle est donc rendue possible par une autre bibliothèque, Canon.js. Des interfaces sont prévues pour facilement intégrer Canon.js ou un autre moteur physique. Avec autant de fonctionnalités disponibles, la complexité de BabylonJS est plus élevée que celles des autres bibliothèques. Cet inconvénient est vite rattrapé par une documentation technique qui détaille le fonctionnement du moteur 3D et qui donne des exemples pour présenter chaque fonctionnalité. La communauté est également très active. Ainsi, beaucoup de différents tutoriels en ligne détaillent le processus de création d'un jeu vidéo du début à la fin. Des outils sont également mis à disposition afin de simplifier le processus de création, comme un éditeur en ligne très complet. Parmi les bibliothèques et frameworks présentés, c'est l'éditeur en ligne le plus poussé. Il permet de créer une scène très complète à l'aide d'outils graphiques puis d'exporter la scène produite.

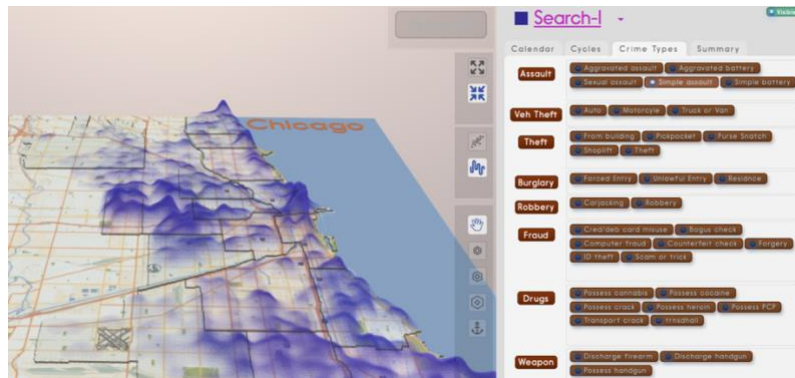
Figure 22 : Editeur en ligne de BabylonJS



<https://editor.babylonjs.com/>

Comme exposé plus haut, BabylonJS se destine principalement à la création de jeux vidéo. Cependant, d'autres applications sont possibles grâce à cette bibliothèque. Par exemple, l'application ci-dessous permet de visualiser sur une carte la fréquence de différents types de crimes qui peuvent être sélectionnés sur la partie droite. En fonction des critères choisis, la carte sera actualisée pour permettre de visualiser les zones qui ont été les plus touchées par la criminalité.

Figure 23 : Autre utilisation de BabylonJS



<https://chicago.il.illuminated.city/#location.geo-search/>

BabylonJS est donc un des moteurs 3D les plus poussés et qui utilise au mieux les possibilités offertes par WebGL. Sa communauté très active et sa documentation complète sont de réels atouts qui permettent de réduire la complexité de l'apprentissage de cette bibliothèque. Référence majeure dans le domaine du développement de jeux vidéo, cette bibliothèque cherche toujours à améliorer son offre. C'est pourquoi, depuis mai 2019, BabylonJS s'intéresse à WebGPU afin de se renouveler autour d'une nouvelle API pour pousser encore plus loin ses performances. La longévité de cette bibliothèque est donc très prometteuse et BabylonJS devrait rester une référence dans son domaine dans les années qui suivent.

## 6. Cas concrets d'utilisations de la 3D

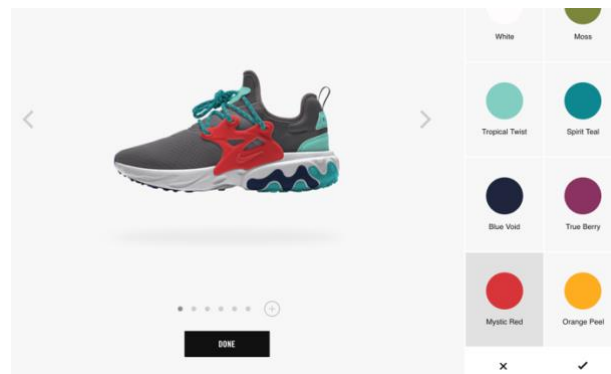
Il est vrai qu'une grande partie des applications qui utilisent de la 3D ne poussent pas forcément au maximum les possibilités offertes par les différentes solutions. Cela n'en reste pas moins un point positif. Cela permet de démontrer que l'intégration de la 3D au Web est possible, qu'elle apporte une plus-value et cela permet de la démocratiser de plus en plus. Il existe des grandes entreprises comme Google qui misent sur la 3D et qui sont des exemples dans l'utilisation poussée de WebGL. Ce n'est pas pour autant la seule entreprise ni le seul domaine à avoir saisi la plus-value que la 3D peut apporter.

D'autres domaines sont également demandeurs et le Web 3D a permis certaines avancées. La médecine, la visualisation de données scientifiques ou l'ingénierie sont des domaines où la 3D est utilisée de manière poussée pour répondre à des besoins précis. La 3D sur le Web semble aussi s'ouvrir au grand public et peut être un bon argument marketing. En effet, proposer au client une expérience où il peut personnaliser son véhicule, sa maison ou ses vêtements lui permet de mieux se projeter, ce qui mène souvent à la vente du bien ou service.

### 6.1 Personnalisation de produits

La personnalisation est devenue un réel argument marketing afin de vendre un service ou un produit. Proposer des systèmes de personnalisation en ligne est donc devenu un enjeu pour des entreprises de plusieurs domaines différents comme le prêt à porter ou encore l'automobile. C'est ainsi que ces domaines ont fait appel à la 3D pour proposer aux clients des systèmes qui leur permettent d'adapter les produits à leurs envies. Dans le domaine du prêt à porter, Nike a par exemple mis en place un système de personnalisation d'une grande partie de ses modèles de chaussures. Le client peut sélectionner un modèle et modifier les couleurs de chaque élément de la chaussure. Les changements sont ensuite appliqués sur un modèle 3D qui est affiché sous un nombre limité de différents angles. Les changements peuvent aussi être effectués en cliquant directement sur un élément de la chaussure qui sera mis en surbrillance et dont les options de personnalisation seront affichées.

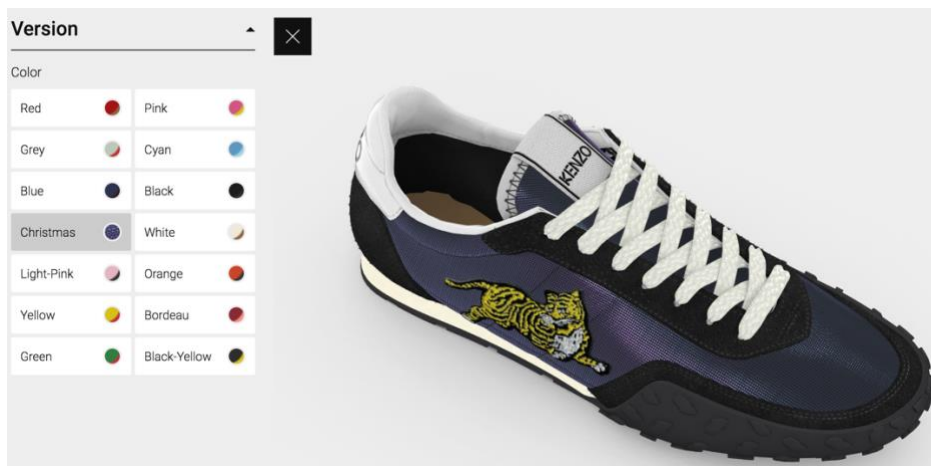
Figure 24 : Système de personnalisation de chaussures Nike



[https://store.nike.com/us/en\\_us/product/custom-nike-react-presto-by-you/?piid=10000515&pbid=779721444](https://store.nike.com/us/en_us/product/custom-nike-react-presto-by-you/?piid=10000515&pbid=779721444)

Dans le cas du système de personnalisation de Nike, il est difficile de parler d'un réel système de personnalisation 3D. Cependant, le fait d'utiliser un modèle basé sur des photos, qui reste interactif, est un bon moyen d'enrichir l'expérience des consommateurs tout en gardant des coûts relativement bas. C'est sûrement cette baisse des coûts qui a permis à Nike d'appliquer ce système sur une grande partie de leurs chaussures. Kenzo, une autre marque de prêt à porter a décidé de miser sur un réel système de personnalisation en 3D. Il n'est pas encore disponible sur le site de Kenzo mais est en démonstration, sur le site des développeurs de cette solution. Pour l'instant uniquement disponible pour un modèle de chaussure spécifique, le fonctionnement se rapproche de celui de Nike. Il est possible de changer la couleur de la chaussure mais la réelle différence se situe au niveau de la visualisation. Kenzo propose un système où il est possible de visualiser la chaussure sous tous ses angles avec un système de rotation sur plusieurs axes. L'utilisation d'un réel modèle 3D a donc été requis, mais cela a également permis de mettre en place plusieurs fonctionnalités. Comme des transitions lors du passage d'une couleur à une autre et des effets de lumière par rapport à la matière de la chaussure et son inclinaison.

Figure 25 : Système de personnalisation de chaussures Kenzo



<https://3d.apviz.io/kenzo/shoes/v1.3.0/validation/index-new.html>

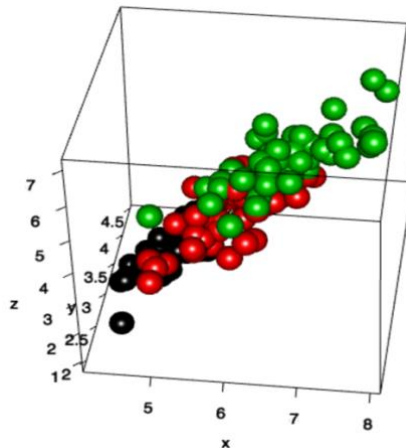
Kenzo a peut-être mieux exploité les possibilités offertes par le WebGL, mais leur système de visionnage a uniquement été appliqué à un seul produit pour l'instant. Ce qui, d'une manière, peut démontrer que le fait d'avoir des systèmes de visualisation aussi poussés soit trop couteux ou demande trop de temps de développement. D'autres entreprises optent pour des solutions plus simples comme Nike, c'est aussi le cas de Yamaha, un constructeur de motos. Le système de personnalisation de Yamaha consiste également à avoir un modèle 3D qui est visible sous quelques angles prédéfinis et qui change en fonction des options sélectionnés sur la moto.

Certaines entreprises font donc le choix d'exploiter au maximum les fonctionnalités offertes par la 3D tandis que d'autres préfèrent un système moins complexe et moins interactif. Cette tendance traduit une transition qui n'a pas été faite intégralement et qui est sûrement freinée par les coûts financiers ou le temps de développement. Ces freins ne permettent donc pas encore de miser complètement sur la 3D.

## 6.2 Visualisation de données statistiques

La science est un domaine très vaste dont certaines branches peuvent tirer profit de la 3D sur le Web. Cet exemple illustre bien la plus-value que la 3D peut apporter à la visualisation de données. Dans ce cas, les données visualisées sont des données statistiques sur le logiciel R, un logiciel que nous avons utilisé lors de notre formation à la HEG. Les résultats, parfois complexes, peuvent facilement devenir illisibles à cause de la grande quantité de données affichées. La visualisation des résultats graphiques se fait normalement en 2D mais grâce à l'intégration de la 3D dans R, les graphiques peuvent désormais être affichés en 3D. Bien entendu, la 3D n'a d'intérêt que lorsque les données doivent être représentées dans des plans orthonormés à 3 axes.

Figure 26 : Représentation graphique interactive en 3D



<https://cran.r-project.org/web/packages/rgl/vignettes/legacyWebGL.html>

Dans l'exemple ci-dessus, WebGL a été utilisé afin de pouvoir voir le graphique sous tous ses angles grâce à une caméra qui gravite autour de celui-ci. Dans ce nuage de points, il est désormais possible de visualiser les points qui sont cachés par d'autres points au premier plan, en changeant l'angle de la caméra.

Il serait possible d'utiliser d'autres fonctionnalités de WebGL afin d'enrichir encore plus les fonctionnalités de cet exemple. WebGL pourrait permettre de sélectionner un point parmi ce nuage afin d'avoir plus de détails sur la nature du point et ses valeurs exactes. Il serait également possible de pouvoir filtrer les différents points en fonction de leur nature et ainsi pouvoir isoler et simplifier la compréhension des données précises. Le WebGL permet donc de réellement pousser les possibilités de la 3D. Même dans des domaines scientifiques où les données sont souvent complexes, tout en apportant une amélioration de la représentation et la compréhension de celles-ci.

### 6.3 Cartographie 3D sur Google maps

Google est une entreprise qu'il n'est plus nécessaire de présenter, ses projets tous plus ambitieux les uns que les autres font avancer le Web et bien d'autres domaines à pas de géant. Parmi ces projets, il y a Google maps, un service de cartographie en ligne lancé en 2004. Il permet de partir de la planète terre et de zoomer jusqu'au point de voir une ville dans ses moindres détails. C'est donc un service très complet proposé par Google qui ne cesse d'évoluer. Les premières avancées ont été les différentes vues disponibles, elles permettent de passer d'une vue de carte classique à une vue satellite plus détaillée avec des photos prises du ciel. Le service proposé par Google est tellement bon qu'il est même désormais utilisé comme système d'identification de lieux d'intérêts et de GPS. Puis est venu Google Street View qui permet de naviguer dans la

majeure partie des rues des grandes villes en vue subjective, grâce à des photos prises depuis des véhicules terrestres. Une fois ces photos assemblées, elles permettent de reconstituer une vue très réaliste des rues d'une ville.

En 2011, Google annonça MapsGL qui est en fait une nouvelle version de Google Maps qui intègre la 3D. WebGL a été utilisé à différents niveaux du service de cartographie de Google, dans Google Street View il a été utilisé afin de réduire les distorsions des photos. L'effet « *fish eye* » qui arrondissait les lignes verticales aux extrémités de l'écran et qui rapetissait le centre de la photo a ainsi été supprimé. WebGL a donc permis un rendu des photos plus lisible et réaliste mais a également contribué à la fluidité des transitions lors du déplacement d'une image à une autre. WebGL a également été utilisé pour apporter plus de détails et de relief aux bâtiments. Lorsque l'on s'approche suffisamment des éléments architecturaux qui ressortent, ils sont mis en valeur grâce à du relief, ce qui donne une réelle impression de 3D très réaliste.

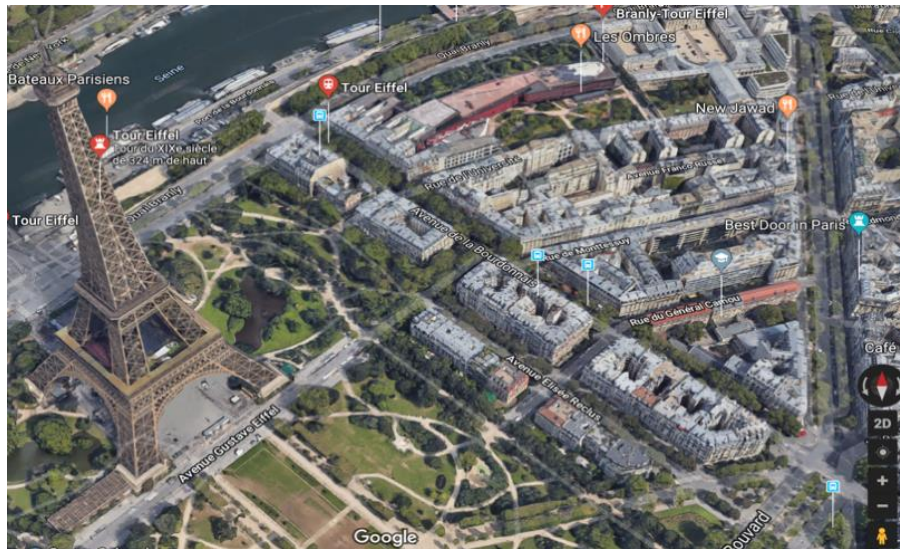
Figure 27 : Google Street View sans et avec WebGL



<https://www.nextinpact.com/archive/66437-google-maps-opengl-webgl-cartographie.htm>

WebGL a également été utilisé dans les modes de vue satellite et carte classique. En vue satellite il est désormais possible de passer d'une vue de haut à 90° en 2D à une vue à 45° en 3D qui permet de faire ressortir les bâtiments et monuments. En vue carte classique, les bâtiments ont été représentés en 3D avec une légère transparence afin d'ajouter du relief sans pour autant nuire à la lisibilité de la carte.

Figure 28 : Vue satellite à 45° en 3D



[https://www.google.com/maps/search/tour+eiffel/@48.8540146,2.2964841,668a,35y,39.27t/data=!3m1!1e](https://www.google.com/maps/search/tour+eiffel/@48.8540146,2.2964841,668a,35y,39.27t/data=!3m1!1e3)

3

Il est difficile de trouver plus d'informations sur la manière dont Google a utilisé WebGL sur son service de cartographie. Ce qui est sûr, c'est que Google a préféré délaissé les solutions nécessitant des plugins externes pour se tourner vers WebGL. Dans un avenir très proche, il est fort à parier que Google puisse renforcer la présence de la 3D et même se tourner vers la réalité virtuelle. Encore un bon point pour WebGL qui propose déjà la réalité virtuelle et qui devrait rester l'API de prédilection pour les avancées de Google.

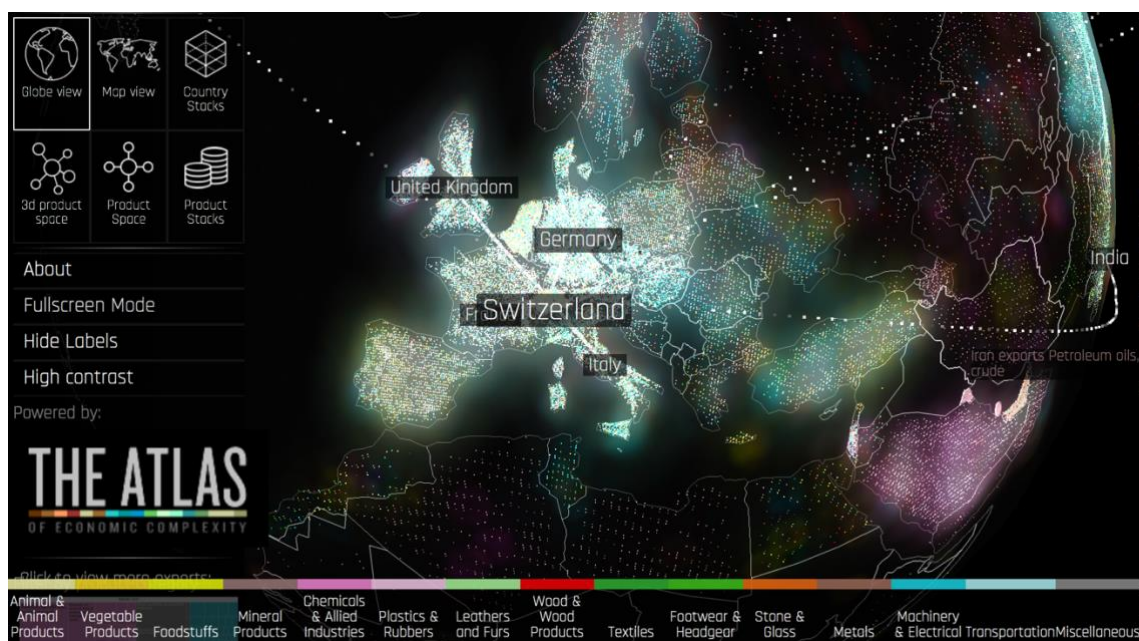
## 6.4 Visualisation de données économiques

La visualisation de données scientifiques est un des domaines qui exploite au mieux les possibilités de la 3D sur le Web. L'exemple suivant, nommé « The globe of economic complexity », a été créé par Owen Cornec et Romain Vuillemot en septembre 2015. Ce projet a vu le jour dans le cadre du centre de développement international à l'université d'Harvard. Le but de ce projet est d'afficher sur un globe interactif l'économie de l'exportation dans le monde. Chaque pixel de couleur représente une valeur de 100 millions de dollars sur une valeur d'exportation totale de 15 trillions de dollars dans le monde en 2012. Des pixels sont placés dynamiquement sur le globe en fonction du pays et leur couleur représente un secteur économique bien précis.

Les possibilités d'interactions sont très nombreuses, il est possible de changer de vue et d'ainsi afficher le globe, une carte 2D, une représentation de points liés entre eux selon leurs exportations et bien d'autres. Il est également possible d'appliquer des critères pour filtrer les données par pays ou par catégorie de biens exportés.



Figure 29 : Liens d'exportations de la Suisse



<http://globe.cid.harvard.edu/?mode=gridSphere&id=CH#>

Très peu d'informations sont présentes par rapport à la conception de ce projet. Cependant, lors de l'affichage du code source de la page, il est possible de voir le chargement de Three.js. On peut donc conclure que le projet a en tout cas en partie été créé à l'aide de la librairie Three.js, ce qui démontre encore une fois les grandes possibilités offertes par les librairies WebGL.

## 7. Réalisation pratique

La partie pratique de ce travail consiste à explorer les possibilités offertes par une bibliothèque en mettant en place plusieurs fonctionnalités. Le projet réalisé n'a donc pas pour but d'être une application concrète ou utile mais plutôt de mettre en œuvre différentes fonctionnalités. Le projet consiste principalement à intégrer le visionnage de modèles 3D à un site Web. Cette fonctionnalité sera mise en œuvre à travers un site Web qui présente des maquettes de maisons en 3D.

### 7.1 Three.js

J'ai choisi de commencer mon projet par cette bibliothèque car c'est la plus populaire, j'ai voulu voir quels sont les points qui ont fait son succès. Three.js est une bibliothèque très verbeuse qui se base sur la programmation orientée objet. Ces fonctionnalités très variées permettaient de mettre en œuvre mon projet. Un système d'importation de modèle 3D ainsi qu'une caméra orbitale avec une source de lumière ont été mis en place. Cependant, après avoir mis en place les bases du projet, j'ai trouvé que la bibliothèque se rapprochait trop des autres langages que j'ai appris pendant ma formation. J'ai également trouvé la bibliothèque légèrement plus complexe que d'autres.

Au final, cette bibliothèque est très complète et permettait de réaliser mon projet, je n'ai donc pas de réel point négatif à lui accorder. Cependant, j'ai préféré essayer une autre bibliothèque qui pourrait être plus adaptée aux besoins de mon projet. Je voulais également essayer les différentes approches que les autres bibliothèques peuvent offrir.

### 7.2 SceneJS

J'ai décidé d'essayer cette bibliothèque car elle offre une approche différente pour développer une solution 3D. Elle n'est certes pas très populaire, mais les divers exemples de solutions développées avec SceneJS m'ont motivé à choisir cette bibliothèque. J'ai également trouvé plus enrichissant d'utiliser une bibliothèque qui a une syntaxe différente des langages que j'ai appris pendant ma formation. J'ai également trouvé cette bibliothèque plus adaptée à mon projet car elle offre les fonctionnalités que je cherche.

Voici la liste des fonctionnalités mises en place :

- Caméra orbitale déplaçable : caméra en rotation autour d'un point qui peut être défini par l'utilisateur avec la possibilité de zoomer.
- Caméra orbitale : caméra en rotation autour d'un point fixe avec la possibilité de zoomer.
- Importation d'un modèle 3D.

- Création de formes géométriques.
- Arrangement d'objets dans l'espace 3D.
- Système de clipping (coupure).
- Application de texture.
- Rotation d'objets 3D.
- Affichage d'informations de l'objet 3D sélectionné.

Après avoir développé mon projet avec cette bibliothèque, je suis satisfait de l'avoir choisie. Elle m'a permis de développer rapidement mon projet dans un laps de temps assez court.

Malgré cette facilité, j'ai eu du mal à pousser plus loin les fonctionnalités de mon projet. Les exemples sur le site officiel sont très complets, mais par forcément très nombreux. J'ai donc eu du mal à adapter certaines fonctionnalités à mon projet et les solutions sur le Web étaient peu nombreuses. J'ai également du mal à savoir si la structure et la syntaxe de mon projet est correcte. Le résultat visuel est correct et il n'y a pas d'erreurs, mais la hiérarchie des éléments de la structure est assez compliquée à comprendre.

## 8. Conclusion

Ce travail s'est surtout focalisé sur une présentation globale de WebGL où plusieurs points ont été abordés. Que ça soit son historique, son fonctionnement, une présentation de quelques cas où WebGL est utilisé de manière poussée ou encore l'exposition de failles de sécurité.

J'ai décidé de faire un travail de recherche car je n'ai pas eu l'opportunité d'étudier le WebGL pendant mon cursus à la HEG. Cette vision plus globale m'a permis de m'initier à cette technologie, de mieux comprendre quelle est la situation actuelle du Web3D. La partie pratique m'a permis de mieux me plonger dans la partie technique et de pouvoir exploiter par moi-même des fonctionnalités de WebGL via l'utilisation d'une librairie.

Grâce à ce travail, j'ai pu étudier le WebGL sous différents angles. La partie consacrée aux technologies précédentes, aux cas d'utilisations et aux futures technologies m'a permis de pouvoir faire quelques hypothèses par rapport au futur de WebGL et plus généralement de la place de la 3D sur le Web.

Dans un prochain travail, il serait intéressant de se focaliser sur l'utilisation approfondie d'une librairie WebGL. Beaucoup de fonctionnalités sont disponibles, mais le temps nécessaire pour les assimiler et les exploiter est important. Créer un projet innovant et poussé permettrait de démontrer l'innovation que la 3D peut apporter au Web. Cependant, il est compliqué de trouver une idée et de choisir quel outil de développement serait le plus adapté à un tel projet.

Au cours de ce travail, j'ai rencontré des difficultés à trier la grande quantité de données que j'avais accumulée durant mes recherches. Beaucoup de sources exposaient les mêmes points, cependant, il était compliqué de trouver plus d'informations techniques quant au fonctionnement interne des solutions. Ce fut le cas pour la présentation de cas qui exploitaient pleinement le WebGL ; peu d'informations étaient disponibles sur la façon dont Google a réellement utilisé WebGL dans Google Maps, pour ne citer que cet exemple.

Enfin, ce fut une expérience enrichissante et positive qui m'a permis d'approfondir un domaine qui m'intéresse.

## Bibliographie

Web 2.0. *Wikipédia* [en ligne]. Dernière modification de la page le 17 mars 2009 à 15:46. [Consulté le 12 mars 2019] Disponible à l'adresse : [https://fr.wikipedia.org/wiki/Web\\_2.0](https://fr.wikipedia.org/wiki/Web_2.0)

LE FALHER, Gérald, 2011. À la découverte de WebGL ! *Daureg.free.fr* [en ligne]. [Consulté le 12 mars 2019]. Disponible à l'adresse : [http://daureg.free.fr/ta\\_webit/index.html](http://daureg.free.fr/ta_webit/index.html)

BONNAFFE, Hugo, 2016. La réalité virtuelle est le Web de demain. *Ovh.com* [en ligne]. 4 mars 2016. [Consulté le 12 mars 2019]. Disponible à l'adresse : <https://www.ovh.com/fr/blog/idees-la-realite-virtuelle-est-le-web-de-demain/>

BELL, Gavin, PARISI, Anthony et PESCE, Mark, 1995. The virtual reality modeling language. *Paulbourke.net* [en ligne]. 9 novembre 1995. Mis à jour le 25 janvier 1996. [Consulté le 12 mars 2019]. Disponible à l'adresse : <http://paulbourke.net/dataformats/vrml1/#Mission%20Statement>

Web3D. *Wikipédia* [en ligne]. Dernière modification de la page le 9 mars 2018 à 10:11. [Consulté le 13 mars 2019] Disponible à l'adresse : <https://fr.wikipedia.org/wiki/Web3D>

LE FALHER, Géraud, 2011. À la découverte de WebGL ! L'histoire de la 3D sur Internet. *Daureg.free.fr* [en ligne]. [Consulté le 13 mars 2019]. Disponible à l'adresse : [http://daureg.free.fr/ta\\_webit/histoire.html](http://daureg.free.fr/ta_webit/histoire.html)

Virtual reality markup language, *Wikipédia* [en ligne]. Dernière modification de la page le 13 février 2019 à 09:43. [Consulté le 13 mars 2019]. Disponible à l'adresse : [https://fr.wikipedia.org/wiki/Virtual\\_Reality\\_Markup\\_Language](https://fr.wikipedia.org/wiki/Virtual_Reality_Markup_Language)

STRASSER, Didier. VRML - Modélisation d'objets en 3D cours d'initiation. *Ecolemoser.ch* [en ligne]. [Consulté le 13 mars 2019]. Disponible à l'adresse : [http://www.ecolemoser.ch/atelier\\_vrml/VRMLLecon.html](http://www.ecolemoser.ch/atelier_vrml/VRMLLecon.html)

VERNA, Didier, 1995. Une introduction à VRML. *Lrde.epita.fr* [en ligne]. Décembre 1995. [Consulté le 17 mars 2019]. Disponible à l'adresse : <https://www.lrde.epita.fr/~didier/lectures/vrml.php>

VERNA, Didier, 1998. Une introduction à VRML 97. *Lrde.epita.fr* [en ligne]. Avril 1998. [Consulté le 17 mars 2019]. Disponible à l'adresse : <https://www.lrde.epita.fr/~didier/lectures/vrml97.php>

Extensible 3D Graphics (X3D). *Techopedia* [en ligne]. [Consulté le 21 mars 2019]. Disponible à l'adresse : <https://www.techopedia.com/definition/10223/extensible-3d-graphics-x3d>

X3D. *EduTechWiki* [en ligne]. Dernière modification de la page le 3 mai 2016 à 00:36. [Consulté le 21 mars 2019]. Disponible à l'adresse : <http://edutechwiki.unige.ch/fr/X3D>

Extensible 3D. *Wikipédia* [en ligne]. Dernière modification de la page le 18 mai 2017 à 10:16. [Consulté le 22 mars 2019]. Disponible à l'adresse : [https://fr.wikipedia.org/wiki/Extensible\\_3D](https://fr.wikipedia.org/wiki/Extensible_3D)

Web3D Consortium. *Wikipédia* [en ligne]. Dernière modification de la page le 27 mai 2018 à 11:25. [Consulté le 22 mars 2019]. Disponible à l'adresse : [https://en.wikipedia.org/wiki/Web3D\\_Consortium](https://en.wikipedia.org/wiki/Web3D_Consortium)

BRUTZMAN, Don, MARCHETTI, Vince et POLYS, Nicholas. X3D. *Web3D Consortium* [en ligne]. [Consulté le 22 mars 2019]. Disponible à l'adresse : <http://www.web3d.org/working-groups/x3d>

BRUTZMAN, 2007. Chapter 1 - Technical overview. *X3DGraphics.com* [en ligne]. [Consulté le 22 mars 2019]. Disponible à l'adresse :

[https://x3dgraphics.com/examples/X3dForWebAuthors/Chapter01-TechnicalOverview/Chapter01Technical\\_Overview.pdf](https://x3dgraphics.com/examples/X3dForWebAuthors/Chapter01-TechnicalOverview/Chapter01Technical_Overview.pdf)

What is X3D? *Web 3D Consortium* [en ligne]. [Consulté le 22 mars 2019]. Disponible à l'adresse : <http://www.web3d.org/x3d/what-x3d>

GARAY, Jérôme, 2009. O3D : API Google pour du Web en 3D. *Generation-nt.com* [en ligne]. 22 avril 2009. [Consulté le 25 mars 2019]. Disponible à l'adresse : <https://www.generation-nt.com/o3d-google-api-3d-web-mozilla-opengl-actualite-580371.html>

O3D. *Wikipédia* [en ligne]. Dernière modification de la page le 3 décembre 2018 à 04:02. [Consulté le 25 mars 2019]. Disponible à l'adresse : <https://en.wikipedia.org/wiki/O3D>

PASSENAUD, Mathieu, 2009. Google O3D - Une API Javascript pour la 3D. *Synergeek.fr* [en ligne]. 14 septembre 2009. [Consulté le 25 mars 2019]. Disponible à l'adresse : <https://www.synergeek.fr/google-o3d/>

JEGX, 2009. (Tested) Google O3D API : First Test and Impressions. *Geeks3d.com* [en ligne]. 24 avril 2009. [Consulté le 25 mars 2019]. Disponible à l'adresse : <http://www.geeks3d.com/20090424/test-google-o3d-api-first-test-and-impressions/>

CAVAZZA, Fred, 2009. Google lance O3D, un plug-in pour faire de la 3D dans le navigateur. *Fredcavazza.net* [en ligne]. 22 avril 2009. [Consulté le 25 mars 2019]. Disponible à l'adresse : <https://fredcavazza.net/2009/04/22/google-lance-o3d-un-plug-in-pour-faire-de-la-3d-dans-le-navigateur/>

Canvas : 3D. *Wiki.mozilla.org* [en ligne]. Dernière modification de la page le 25 janvier 2010 à 20:35. [Consulté le 25 mars 2019]. Disponible à l'adresse : <https://wiki.mozilla.org/Canvas:3D>

An introduction to WebGL. *Oreilly.com* [en ligne]. [Consulté le 2 avril 2019]. Disponible à l'adresse : <https://www.oreilly.com/library/view/webgl-up-and/9781449326487/ch01.html>

BARAT, Marc, 2012. Applications populaires en WebGL. *Igm.univ-mlv.fr* [en ligne]. 2012. [Consulté le 2 avril 2019]. Disponible à l'adresse : <http://igm.univ-mlv.fr/~dr/XPOSE2012/Introduction%20au%20WebGL/utilisations.html>

WebGL – Introduction. *Tutorialspoint.com* [en ligne]. [Consulté le 5 avril 2019]. Disponible à l'adresse : [https://www.tutorialspoint.com/webgl/webgl\\_introduction.htm](https://www.tutorialspoint.com/webgl/webgl_introduction.htm)

Rodolphe, 2009. WebGL la future 3D pour le web. *Alsacreations.com* [en ligne]. 17 décembre 2009. [Consulté le 8 avril 2019]. Disponible à l'adresse : <https://www.alsacreations.com/actu/lire/918-webgl-3d-api-opengl-web-javascript.html>

WebGL Comment ça marche. *Webglfundamentals.org* [en ligne]. [Consulté le 8 avril 2019]. Disponible à l'adresse : <https://webglfundamentals.org/webgl/lessons/fr/webgl-how-it-works.html>

Explication des bases théoriques de la 3D. *developer.mozilla.org* [en ligne]. 23 mars 2019. [Consulté le 11 avril 2019]. Disponible à l'adresse : [https://developer.mozilla.org/fr/docs/Games/Techniques/3D\\_on\\_the\\_web/Basic\\_theory](https://developer.mozilla.org/fr/docs/Games/Techniques/3D_on_the_web/Basic_theory)

Getting started with WebGL. *Csuregina.ca* [en ligne]. [Consulté le 13 avril 2019]. Disponible à l'adresse : <http://www.cs.uregina.ca/Links/class-info/315/WebGL/Lab1/wip.html>

OpenGL extension wrangler library. *Wikipédia* [en ligne]. Dernière modification de la page le 9 mars 2019 à 21:51. [Consulté le 14 avril 2019]. Disponible à l'adresse : [https://fr.wikipedia.org/wiki/OpenGL\\_extension\\_wrangler\\_library](https://fr.wikipedia.org/wiki/OpenGL_extension_wrangler_library)

MAAZOUN, Mouna, 2017. WebGL. *Fr.slideshare.net* [en ligne]. 7 aout 2017. [Consulté le 16 avril 2019]. Disponible à l'adresse : <https://fr.slideshare.net/MounaMaazoun/webgl-78630413>

BARAT, Marc, 2012. Limites de WebGL. *Igm.univ-mlv.fr* [en ligne]. 2012 [Consulté le 20 avril 2019]. Disponible à l'adresse : <http://igm.univ-mlv.fr/~dr/XPOSE2012/Introduction%20au%20WebGL/inconvenients.html>

FRAUSTO-ROBLEDO, Anthony, 2019. WebGL 2.0 – Why it's the path to stable open standards-based 3D web graphics. *Architosh.com* [en ligne]. 12 février 2019. [Consulté le 20 avril 2019]. Disponible à l'adresse : <https://architosh.com/2019/02/webgl-2-0-why-its-the-path-to-stable-open-standards-based-3d-web-graphics/>

Vulkan (API). *Wikipédia* [en ligne]. Dernière modification de la page le 21 avril 2019 à 18:43. [Consulté le 21 avril 2019]. Disponible à l'adresse : [https://fr.wikipedia.org/wiki/Vulkan\\_\(API\)](https://fr.wikipedia.org/wiki/Vulkan_(API))

LADEVIE, Quentin, 2018. Vulkan, le nouvel OpenGL en mieux ? *Wanadev.fr* [en ligne]. 2018. [Consulté le 22 avril 2019]. Disponible à l'adresse : <https://www.wanadev.fr/143-vulkan-le-nouvel-opengl-en-mieux/>

WebGPU : Apple propose de créer un nouveau standard pour les graphismes 3D sur le web. *Developpez.com* [en ligne]. 8 février 2017. [Consulté le 25 avril 2019]. Disponible à l'adresse : <https://www.developpez.com/actu/116852/WebGPU-Apple-propose-de-creeer-un-nouveau-standard-pour-les-graphismes-3D-sur-le-web-vers-une-alternative-bas-niveau-a-WebGL/>

Why 3D is the future of the internet ? *Medium.com* [en ligne]. 4 décembre 2018. [Consulté le 25 avril 2019]. Disponible à l'adresse : <https://medium.com/naker/why-3d-is-the-future-of-the-internet-44d616633aba>

MURDOCH, Duncan, 2019. User interaction in WebGL. *Cran.r-project.org* [en ligne]. 12 mars 2019. [Consulté le 25 avril 2019]. Disponible à l'adresse : <https://cran.r-project.org/web/packages/rgl/vignettes/WebGL.html>

Is WebGL a security concern ? *Security.stackexchange.com* [en ligne]. [Consulté le 26 avril 2019]. Disponible à l'adresse : <https://security.stackexchange.com/questions/13799/is-webgl-a-security-concern>

WebGL – More WebGL security flaws. *Contextis.com* [en ligne]. [Consulté le 30 avril 2019]. Disponible à l'adresse : <https://www.contextis.com/en/blog/webgl-more-webgl-security-flaws>

WebGL – 3D canvas graphics. *Caniuse.com* [en ligne]. [Consulté le 3 mai 2019]. Disponible à l'adresse : <https://caniuse.com/#search=webgl>

NINOMIYA, Kai, MO, Zhenyao, RUSSELL, Ken, 2017. WebGL 2.0 is here : what you need to know. *Khronos.org* [en ligne]. 11 avril 2017. [Consulté le 3 mai 2019]. Disponible à l'adresse : [https://www.khronos.org/assets/uploads/developers/library/2017-webgl-webinar/Khronos-Webinar-WebGL-20-is-here\\_What-you-need-to-know\\_Apr17.pdf](https://www.khronos.org/assets/uploads/developers/library/2017-webgl-webinar/Khronos-Webinar-WebGL-20-is-here_What-you-need-to-know_Apr17.pdf)

Benjamin, 2017. De WebGL à WebGL 2. *Wanadev.fr* [en ligne]. 2017 [Consulté le 8 mai 2019]. Disponible à l'adresse : <https://www.wanadev.fr/111-de-webgl-a-webgl-2/>

Le WebGL a-t-il un avenir ? *Jeux.developpez.com* [en ligne]. 2014 [Consulté le 8 mai 2019]. Disponible à l'adresse : <https://jeux.developpez.com/actu/74941/Le-WebGL-a-t-il-un-avenir-Faisons-le-point-sur-le-standard-Web-permettant-d-integrer-un-rendu-3D-dans-une-page-Web/>

PETRICEK, Tomas, 2015. Library patterns : Why frameworks are evil. *Tomasp.net* [en ligne]. 3 mars 2019 [Consulté le 9 mai 2019]. Disponible à l'adresse : <http://tomasp.net/blog/2015/library-frameworks/>

The difference between a framework and a library. *Freecodecamp.org* [en ligne]. 1 février 2019 [Consulté le 15 mai 2019]. Disponible à l'adresse : <https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/>

WALKER, James, 2016. GPU multi-texture sprite batching ! *Medium.com* [en ligne]. 25 janvier 2016 [Consulté le 15 mai 2019]. Disponible à l'adresse : <https://medium.com/goodboy-digital/gpu-multi-texture-sprite-batching-21c90ae8f89b>

DIDIER, Jean-Yves. Présentation de three.js. *Isc.univ-evry.fr* [en ligne]. [Consulté le 16 mai 2019]. Disponible à l'adresse : <http://isc.univ-evry.fr/~didier/home/lib/exe/fetch.php?media=cours:ig:threejs.pdf>

Three.js. *Wikipédia* [en ligne]. Dernière modification de la page le 10 juin 2019 à 17:09. [Consulté le 19 mai 2019]. Disponible à l'adresse : <https://en.wikipedia.org/wiki/Three.js>

Three.js materials. *Threejsfundamentals.org* [en ligne]. [Consulté le 19 mai 2019]. Disponible à l'adresse : <https://threejsfundamentals.org/threejs/lessons/threejs-materials.html>

KAY, Lindsay, 2013. SceneJS overview. *Github.com* [en ligne]. 20 septembre 2013 [Consulté le 20 mai 2019]. Disponible à l'adresse : <https://github.com/xeolabs/scenejs/wiki/SceneJS-Overview>

LI, Sing, 2019. Code less, do more with WebGL libraries. *Ibmcode-staging.us-east.containers.mybluemix.net* [en ligne]. 26 avril 2019 [Consulté le 22 mai 2019]. Disponible à l'adresse : <https://ibmcode-staging.us-east.containers.mybluemix.net/tutorials/wa-webgl2/>

Blend4Web. *Wikipédia* [en ligne]. Dernière modification de la page le 22 aout 2018 à 12:59. [Consulté le 23 mai 2019]. Disponible à l'adresse : <https://fr.wikipedia.org/wiki/Blend4Web>

FROMY, Anthony, 2018. Le site dédié aux technologies du web. *Idboard.net* [en ligne]. 12 janvier 2018. [Consulté le 27 mai 2019]. Disponible à l'adresse : <http://idboard.net:10000/web/2018/01/12/faire-un-jeu-avec-blend4web/>

Babylon.js. *Wikipédia* [en ligne]. Dernière modification de la page le 3 mai 2019 à 09:37. [Consulté le 29 mai 2019]. Disponible à l'adresse : <https://en.wikipedia.org/wiki/Babylon.js>

Rodolphe, 2014. Babylon.js. *Alsacreations.com* [en ligne]. 29 mai 2014. [Consulté le 2 juin 2019]. Disponible à l'adresse : <https://www.alsacreations.com/outils/lire/1631-BabylonJS.html>

WebGPU is coming to Babylon.js. *Medium.com* [en ligne]. 10 mai 2019 [Consulté le 7 juin 2019]. Disponible à l'adresse : <https://medium.com/@babylonjs/webgpu-is-coming-to-babylon-js-c44f8065ac05>

GAVOIS, Sébastien, 2011. MapsGL : Google Maps passe à WebGL et devient plus fluide. *Nextinpact.com* [en ligne]. 17 octobre 2011. [Consulté le 7 juin 2019]. Disponible à l'adresse : <https://www.nextinpact.com/archive/66437-google-maps-opengl-webgl-cartographie.htm>

BELFIORE, Guillaume, 2011. MapsGL : une version de Google Maps basée sur WebGL. *Cubic.com* [en ligne]. 13 octobre 2011. [Consulté le 11 juin 2019]. Disponible à l'adresse : <https://www.cubic.com/internet/univers-google/google-maps/actualite-452502-google-maps-basee-webgl.html>

CHARLIER, Jean-Noël, 2018. WebGL : Usages et applications de la 3D sur le web. *Wanadev.fr* [en ligne]. 2018. [Consulté le 11 juin 2019]. Disponible à l'adresse : <https://www.wanadev.fr/134-webgl-usages-et-applications-de-la-3d-sur-le-web-1/>



Adobe Flash Player. *Wikipédia* [en ligne]. Dernière modification de la page le 16 avril 2019 à 11:05. [Consulté le 11 juin 2019]. Disponible à l'adresse : [https://fr.wikipedia.org/wiki/Adobe\\_Flash\\_Player](https://fr.wikipedia.org/wiki/Adobe_Flash_Player)

## Annexe 1 : Compatibilité des premières versions des navigateurs d'ordinateur de bureau

Navigateur	WebGL 1.0		WebGL 2.0	
	Version	Date	Version	Date
Internet Explorer	11	17.10.2013	Non compatible	
Edge	12	29.07.2015	Non compatible	
Firefox	4	06.07.2010	51	24.01.2017
Chrome	9	03.02.2011	56	25.01.2017
Safari	5.1	20.07.2011	Non compatible	
Opera	12.1	05.11.2012	43	07.02.2017

## Annexe 2 : Compatibilité des premières versions des navigateurs pour mobiles

Navigateur	WebGL 1.0		WebGL 2.0	
	Version	Date	Version	Date
iOS Safari	8.1	17.09.2014	Non compatible	
Blackberry Browser	10	30.01.2013	Non compatible	
Opera mobile	12	01.03.2012	43	23.09.2016
Chrome pour Android	25	10.01.2013	58	03.05.2017
Firefox pour Android	4	12.11.2011	51	18.01.2017
Internet Explorer mobile	Non compatible		Non compatible	
Edge	10	10.06.2016	Non compatible	

## Annexe 3 : Partie pratique – fichier « index.html »

```
1. <!--
2.   Auteur   : Ivo de Carvalho Matosinhos
3.   Date    : 28.06.2019
4.   Version : 1.0
5.   Projet  : Travail de Bachelor - HouseViewer
6. -->
7.
8. <!doctype html>
9. <html lang="fr">
10.   <head>
11.     <!-- Required meta tags -->
12.     <meta charset="utf-8">
13.     <meta name="viewport" content="width=device-width, initial-
14.       scale=1, shrink-to-fit=no">
15.     <!-- Bootstrap CSS -->
16.     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap
17.       /4.3.1/css/bootstrap.min.css" integrity="sha384-
18.       ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="an
19.       onymous">
20.     <link rel="stylesheet" href="style.css">
21.   </head>
22.   <body>
23.     <!-- Navigation -->
24.     <nav class="navbar navbar-dark bg-dark fixed-top">
25.       <div class="container">
26.         <a class="navbar-brand" href="index.html">House viewer</a>
27.         <button class="navbar-toggler" type="button" data-
28.           toggle="collapse" data-target="#navbarResponsive" aria-
29.           controls="navbarResponsive" aria-expanded="false" aria-
30.           label="Toggle navigation">
31.           <span class="navbar-toggler-icon"></span>
32.         </button>
33.         <div class="collapse navbar-collapse" id="navbarResponsive">
34.           <ul class="navbar-nav ml-auto">
35.             <li class="nav-item">
36.               <a class="nav-
37.                 link" href="houseViewerCologne_3D.html">Maison à Cologne</a>
38.             </li>
39.             <li class="nav-item">
40.               <a class="nav-
41.                 link" href="houseViewerSatigny_3D.html">Appartement à Satigny</a>
42.             </li>
43.             <li class="nav-item">
44.               <a class="nav-
45.                 link" href="houseViewerMaquette.html">Maquette</a>
46.             </li>
47.           </ul>
48.         </div>
49.       </div>
50.     </nav>
51.     <header>
52.       <div id="carouselExampleIndicators" class="carousel slide" data-
53.         ride="carousel">
54.         <ol class="carousel-indicators">
55.           <li data-target="#carouselExampleIndicators" data-slide-
56.             to="0" class="active"></li>
57.           <li data-target="#carouselExampleIndicators" data-slide-
58.             to="1"></li>
59.         </ol>
```

```

51.         <div class="carousel-inner" role="listbox">
52.             <!--
53.             - Slide One - Set the background image for this slide in the line below -->
54.             <div class="carousel-item active" style="background-
55.             image: url('http://localhost:8888/models/img/Cognony.png')">
56.                 <a href="houseViewerCognony_3D.html">
57.                     <div class="carousel-caption d-none d-md-block">
58.                         <h3 class="display-4">Maison à Cognony</h3>
59.                         <p class="lead">Maison à étage avec un grand jard
60.                     in</p>
61.                     </div>
62.                 </a>
63.             </div>
64.             <!--
65.             - Slide Two - Set the background image for this slide in the line below -->
66.             <div class="carousel-item" style="background-
67.             image: url('http://localhost:8888/models/img/Satigny.png')">
68.                 <a href="houseViewerSatigny_3D.html">
69.                     <div class="carousel-caption d-none d-md-block">
70.                         <h3 class="display-
71.                         4">Appartement à Satigny</h3>
72.                         <p class="lead">Appartement moderne avec mezzanin
73.                         e au coeur de la campagne</p>
74.                     </div>
75.                 </a>
76.             </div>
77.             <div>
78.                 <a class="carousel-control-
79.                 prev" href="#carouselExampleIndicators" role="button" data-slide="prev">
80.                     <span class="carousel-control-prev-icon" aria-
81.                     hidden="true"></span>
82.                     <span class="sr-only">Previous</span>
83.                 </a>
84.                 <a class="carousel-control-
85.                 next" href="#carouselExampleIndicators" role="button" data-slide="next">
86.                     <span class="carousel-control-next-icon" aria-
87.                     hidden="true"></span>
88.                     <span class="sr-only">Next</span>
89.                 </a>
90.             </div>
91.         </header>
92.
93.         <!-- Page Content -->
94.         <section class="py-5">
95.             <div class="container">
96.                 <h1 class="font-weight-light">Bienvenue sur House Viewer</h1>
97.                 <p class="lead">Vous trouverez le bien de vos rêves et cela depui
98.                 s chez vous grâce à nos maquettes 3D de nos biens en vente</p>
99.             </div>
100.        </section>
101.
102.        <!-- Optional JavaScript -->
103.        <!-- jQuery first, then Popper.js, then Bootstrap JS -->
104.        <script src="https://code.jquery.com/jquery-
105.        3.3.1.slim.min.js" integrity="sha384-
106.        q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="an
107.        onymous"></script>
108.        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/
109.        popper.min.js" integrity="sha384-
110.        U02eT0CpHqDz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="an
111.        onymous"></script>
112.        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootst
113.        rap.min.js" integrity="sha384-
114.        JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEeFf/njGzIxFDs40xIM+B07jRM" crossorigin="an
115.        onymous"></script>
116.    </body>

```

96. `</html>`

## Annexe 4 : Partie pratique – fichier « houseViewerSatigny\_3D.html »

```
1. <!--
2.   Auteur   : Ivo de Carvalho Matosinhos
3.   Date    : 28.06.2019
4.   Version : 1.0
5.   Projet  : Travail de Bachelor - HouseViewer
6. -->
7.
8. <!doctype html>
9. <html lang="fr">
10.  <head>
11.    <!-- Required meta tags -->
12.    <meta charset="utf-8">
13.    <meta name="viewport" content="width=device-width, initial-
14.  scale=1, shrink-to-fit=no">
15.
16.    <!-- Bootstrap CSS -->
17.    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap
18.  /4.3.1/css/bootstrap.min.css" integrity="sha384-
19.  ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="an
20.  onymous">
21.    <link rel="stylesheet" href="style.css">
22.
23.    <script src="https://code.jquery.com/jquery-
24.  3.3.1.slim.min.js" integrity="sha384-
25.  q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="an
26.  onymous"></script>
27.    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/umd/
28.  popper.min.js" integrity="sha384-
29.  U02eT0CpHqSJQ6hJty5KVphtPhzWj9W01c1HTMGA3JDZwrnQq4sF86dIHNDz0W1" crossorigin="an
30.  onymous"></script>
31.    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootst
32.  rap.min.js" integrity="sha384-
33.  JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEeFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="an
34.  onymous"></script>
35.
36.    <script src="http://scenejs.org/api/latest/scenejs.js"></script>
37.
38.  </head>
39.  <body>
40.    <!-- Navigation -->
41.    <nav class="navbar navbar-dark bg-dark fixed-top">
42.      <div class="container">
43.        <a class="navbar-brand" href="index.html">House viewer</a>
44.        <button class="navbar-toggler" type="button" data-
45.  toggle="collapse" data-target="#navbarResponsive" aria-
46.  controls="navbarResponsive" aria-expanded="false" aria-
47.  label="Toggle navigation">
48.          <span class="navbar-toggler-icon"></span>
49.        </button>
50.        <div class="collapse navbar-collapse" id="navbarResponsive">
51.          <ul class="navbar-nav ml-auto">
52.            <li class="nav-item">
53.              <a class="nav-
54.  link" href="houseViewerCologne_3D.html">Maison à Cologne</a>
55.            </li>
56.            <li class="nav-item active">
57.              <a class="nav-
58.  link" href="houseViewerSatigny_3D.html">Appartement à Satigny
59.              <span class="sr-only">(current)</span>
60.            </li>
61.          </ul>
62.        </div>
63.      </div>
64.    </nav>
65.  </body>
66. </html>
```

```

44.         </li>
45.         <li class="nav-item">
46.             <a class="nav-
link" href="houseViewerMaquette.html">Maquette</a>
47.         </li>
48.     </ul>
49. </div>
50. </div>
51. </nav>
52.
53. <script>
54.     SceneJS.setConfigs({
55.         pluginPath: "http://scenejs.org/api/latest/plugins"
56.     });
57.
58.     var scene = SceneJS.createScene({
59.         // Camera
60.         nodes: [
61.             {
62.                 type: "cameras/pickFlyOrbit",
63.                 id: "maCamera",
64.                 yaw: 0,
65.                 pitch: -20, // -90=vueDuDessus / -20=vue3D
66.                 maxPitch: -10,
67.                 minPitch: -80,
68.                 zoom: 800,
69.                 eye: {x: 0, y: 150, z: -1000},
70.                 look: {x: 0, y: -210, z: -20},
71.                 zoomSensitivity: 10,
72.                 showCursor: false,
73.                 cursorSize: 0.1,
74.
75.                 // Clipping
76.                 nodes: [{
77.                     type: "translate",
78.                     id: "monClipping",
79.                     x: 0.0,
80.                     y: -200.0, // -78=RDC / -200=1er
81.                     z: 0.0,
82.
83.                     // Clipping
84.                     nodes: [{
85.                         type: "clips",
86.                         clips: [{
87.                             x: 0,
88.                             y: 0.1,
89.                             z: 0,
90.                             mode: "inside"
91.                         }],
92.
93.                         nodes: [{
94.                             type: "flags",
95.                             flags: {
96.                                 solid: true
97.                             },
98.
99.                             // Rotation
100.                            nodes: [{
101.                                type: "rotate",
102.                                y: 1,
103.                                angle: 130,
104.                                nodes: [{
105.                                    type: "materia
106.                                },
                                baseColor: {r:
0.5, g: 0.5, b: 0.5}],

```



```

107.                                     nodes: [{
108.                                     type:
"import/obj",
109.                                     src: "
http://localhost:8888/models/house5.obj"
110.                                     }]
111.                                     }]
112.                                     }]
113.                                     }]
114.                                     },
115.                                     // Skybox
116.                                     {
117.                                     type: "skybox/cloudySea",
118.                                     size: 5000
119.                                     }
120.                                     ]
121.                                     }]
122.                                     });
123.                                     </script>
124.                                     </body>
125.                                     </html>
126.

```

## Annexe 5 : Partie pratique – fichier « houseViewerMaquette.html »

```
1. <!--
2.   Auteur   : Ivo de Carvalho Matosinhos
3.   Date    : 28.06.2019
4.   Version : 1.0
5.   Projet  : Travail de Bachelor - HouseViewer
6. -->
7.
8. <!doctype html>
9. <html lang="fr">
10.  <head>
11.    <!-- Required meta tags -->
12.    <meta charset="utf-8">
13.    <meta name="viewport" content="width=device-width, initial-
14.  scale=1, shrink-to-fit=no">
15.
16.    <!-- Bootstrap CSS -->
17.    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap
18.  /4.3.1/css/bootstrap.min.css" integrity="sha384-
19.  ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="an
20.  onymous">
21.    <link rel="stylesheet" href="style.css">
22.
23.    <script src="https://code.jquery.com/jquery-
24.  3.3.1.slim.min.js" integrity="sha384-
25.  q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="an
26.  onymous"></script>
27.    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/umd/
28.  popper.min.js" integrity="sha384-
29.  U02eT0CpHqSJQ6hJty5KVphtPhzWj9W01c1HTMGA3JDZwrnQq4sF86dIHNDz0W1" crossorigin="an
30.  onymous"></script>
31.    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootst
32.  rap.min.js" integrity="sha384-
33.  JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEeAf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="an
34.  onymous"></script>
35.
36.    <script src="http://scenejs.org/api/latest/scenejs.js"></script>
37.
38.  </head>
39.  <body>
40.    <!-- Navigation -->
41.    <nav class="navbar navbar-dark bg-dark fixed-top">
42.      <div class="container">
43.        <a class="navbar-brand" href="index.html">House viewer</a>
44.        <a class="navbar" id="infoPiece" style="color:white;"></a>
45.        <button class="navbar-toggler" type="button" data-
46.  toggle="collapse" data-target="#navbarResponsive" aria-
47.  controls="navbarResponsive" aria-expanded="false" aria-
48.  label="Toggle navigation">
49.          <span class="navbar-toggler-icon"></span>
50.        </button>
51.        <div class="collapse navbar-collapse" id="navbarResponsive">
52.          <ul class="navbar-nav ml-auto">
53.            <li class="nav-item">
54.              <a class="nav-
55.  link" href="houseViewerCologne_3D.html">Maison à Cologne</a>
56.            </li>
57.            <li class="nav-item">
58.              <a class="nav-
59.  link" href="houseViewerSatigny_3D.html">Appartement à Satigny</a>
60.            </li>

```

```

44.         <li class="nav-item">
45.             <a class="nav-
link active" href="houseViewerMaquette.html">Maquette
46.                 <span class="sr-only">(current)</span>
47.             </a>
48.         </li>
49.     </ul>
50. </div>
51. </div>
52. </nav>
53.
54. <script>
55.     SceneJS.setConfigs({
56.         pluginPath: "http://scenejs.org/api/latest/plugins"
57.     });
58.
59.     var myScene = SceneJS.createScene({
60.
61.         // Camera
62.         nodes: [{
63.             type: "cameras/orbit",
64.             zoom: 20,
65.             zoomSensitivity: 0.7,
66.
67.             // Identification du noeud
68.             nodes: [{
69.                 type: "name",
70.                 name: "salleDeBain",
71.
72.                 // Positionnement
73.                 nodes: [{
74.                     type: "translate",
75.                     x: -6,
76.                     y: -4.5,
77.                     z: 0,
78.
79.                     // Texture
80.                     nodes: [{
81.                         type: "texture",
82.                         src: "http://localhost:8888/model
s/textures/bois.jpg",
83.
84.                         // Creation de la forme geometriq
ue
85.                         nodes: [{
86.                             type: "geometry/box",
87.                             xSize: 3,
88.                             ySize: 2,
89.                             zSize: 0.1
90.                         }]
91.                     }]
92.                 }]
93.             },
94.             {
95.                 // Identification du noeud
96.                 type: "name",
97.                 name: "salon",
98.
99.                 // Positionnement
100.                nodes: [{
101.                    type: "translate",
102.                    x: 3,
103.                    y: -0.5,
104.                    z: 0,
105.
106.                    // Texture

```

```

107.                 nodes: [{
108.                     type: "texture",
109.                     src: "http://localhost:8888/mo
dels/textures/brique.jpg",
110.
111.                     // Creation de la forme geomet
rique
112.                 nodes: [{
113.                     type: "geometry/box",
114.
115.                     xSize: 6,
116.                     ySize: 6,
117.                     zSize: 0.1
118.                 }]
119.             }],
120.         },
121.         {
122.             // Identification du noeud
123.             type: "name",
124.             name: "cuisine",
125.
126.             // Positionnement
127.             nodes: [{
128.                 type: "translate",
129.                 x: -6,
130.                 y: 1.5,
131.                 z: 0,
132.
133.                 // Texture
134.                 nodes: [{
135.                     type: "texture",
136.                     src: "http://localhost:8888/mo
dels/textures/beton.jpg",
137.
138.                     // Creation de la forme geomet
rique
139.                 nodes: [{
140.                     type: "geometry/box",
141.
142.                     xSize: 3,
143.                     ySize: 4,
144.                     zSize: 0.1
145.                 }]
146.             }],
147.         }],
148.     }],
149. });
150. // Pick hit handler
151. // Shows pick info in an HTML element
152. var pickInfo = document.getElementById("infoPiece");
153. myScene.on("pick",
154.     function (hit) {
155.
156.         pickInfo.innerHTML = "Pièce sélectionnée : " + JSON.st
ringify(hit.name);
157.         // To illustrate, these are the params to expect on th
e pick hit:
158.         var name = hit.name;
159.         var path = hit.path; // Eg. "foo.object1"
160.         var nodeId = hit.nodeId;
161.         var canvasX = hit.canvasPos[0];
162.         var canvasY = hit.canvasPos[1];
163.     });
164. // Called when nothing picked

```

```

165.         myScene.on("nopick",
166.             function (hit) {
167.                 pickInfo.innerHTML = "Aucune pièce sélectionnée";
168.             });
169.         // Mouse event handling to do a pick on each mouse click
170.
171.         var canvas = myScene.getCanvas();
172.         var lastX;
173.         var lastY;
174.         var dragging;
175.         canvas.addEventListener('mousedown',
176.             function (event) {
177.                 lastX = event.clientX;
178.                 lastY = event.clientY;
179.                 dragging = true;
180.             }, true);
181.         canvas.addEventListener('mouseup',
182.             function (event) {
183.                 if (dragging && Math.abs(event.clientX - lastX) < 3 &&
Math.abs(event.clientY - lastY) < 3) {
184.
185.                     // Do pick
186.                     myScene.pick({
187.                         canvasPos: [event.clientX, event.clientY]
188.                     });
189.                 }
190.                 dragging = false;
191.             }, true);
192.         canvas.addEventListener('touchstart',
193.             function touchStart(event) {
194.                 lastX = event.targetTouches[0].clientX;
195.                 lastY = event.targetTouches[0].clientY;
196.                 dragging = true;
197.             }, true);
198.         canvas.addEventListener('touchend',
199.             function touchEnd(event) {
200.                 if (dragging && event.targetTouches[0].clientX == last
X && event.targetTouches[0].clientY == lastY) {
201.
202.                     // Do pick
203.                     myScene.pick({
204.                         canvasPos: [event.targetTouches[0].clientX, ev
ent.targetTouches[0].clientY]
205.                     });
206.                 }
207.                 dragging = false;
208.             }, true);
209.
210.         </script>
211.     </body>
212. </html>

```

## Annexe 6 : Partie pratique – fichier « houseViewerCologne\_3D.html »

```
1. <!--
2.   Auteur   : Ivo de Carvalho Matosinhos
3.   Date    : 28.06.2019
4.   Version : 1.0
5.   Projet  : Travail de Bachelor - HouseViewer
6. -->
7.
8. <!doctype html>
9. <html lang="fr">
10.  <head>
11.    <!-- Required meta tags -->
12.    <meta charset="utf-8">
13.    <meta name="viewport" content="width=device-width, initial-
14.      scale=1, shrink-to-fit=no">
15.
16.    <!-- Bootstrap CSS -->
17.    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap
18.      /4.3.1/css/bootstrap.min.css" integrity="sha384-
19.      ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="an
20.      onymous">
21.    <link rel="stylesheet" href="style.css">
22.
23.    <script src="https://code.jquery.com/jquery-
24.      3.3.1.slim.min.js" integrity="sha384-
25.      q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="an
26.      onymous"></script>
27.    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/
28.      popper.min.js" integrity="sha384-
29.      U02eT0CpHqSJQ6hJty5KVphtPhzWj9W01c1HTMGA3JDZwrnQq4sF86dIHNDz0W1" crossorigin="an
30.      onymous"></script>
31.    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootst
32.      rap.min.js" integrity="sha384-
33.      JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEeAf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="an
34.      onymous"></script>
35.
36.    <script src="http://scenejs.org/api/latest/scenejs.js"></script>
37.
38.  </head>
39.  <body>
40.    <!-- Navigation -->
41.    <nav class="navbar navbar-dark bg-dark fixed-top">
42.      <div class="container">
43.        <a class="navbar-brand" href="index.html">House viewer</a>
44.        <button class="navbar-toggler" type="button" data-
45.          toggle="collapse" data-target="#navbarResponsive" aria-
46.          controls="navbarResponsive" aria-expanded="false" aria-
47.          label="Toggle navigation">
48.          <span class="navbar-toggler-icon"></span>
49.        </button>
50.        <div class="collapse navbar-collapse" id="navbarResponsive">
51.          <ul class="navbar-nav ml-auto">
52.            <li class="nav-item">
53.              <a class="nav-
54.                link" href="houseViewerCologne_3D.html">Maison à Cologne</a>
55.            </li>
56.            <li class="nav-
57.              link active" href="houseViewerCologne_3D.html">Visualisation 3D
58.            </li>
59.          </ul>
60.        </div>
61.      </div>
62.    </nav>
63.  </body>
64. </html>
```

```

43.         <a class="nav-
link" href="houseViewerCologne_2D_RDC.html">Visualisation 2D : Rez-de-
chaussée</a>
44.         <a class="nav-
link" href="houseViewerCologne_2D_1er.html">Visualisation 2D : 1er étage</a>
45.     </ul>
46. </li>
47. <li class="nav-item">
48.     <a class="nav-
link" href="houseViewerSatigny_3D.html">Appartement à Satigny</a>
49. </li>
50. <li class="nav-item">
51.     <a class="nav-
link" href="houseViewerMaquette.html">Maquette</a>
52. </li>
53. </ul>
54. </div>
55. </div>
56. </nav>
57.
58. <script>
59.     SceneJS.setConfigs({
60.         pluginPath: "http://scenejs.org/api/latest/plugins"
61.     });
62.
63.     var scene = SceneJS.createScene({
64.         // Camera
65.         nodes: [{
66.             type: "cameras/pickFlyOrbit",
67.             id: "maCamera",
68.             yaw: 0,
69.             pitch: -20, // -90=vueDuDessus / -20=vue3D
70.             maxPitch: -10,
71.             minPitch: -80,
72.             zoom: 1000,
73.             eye: {x: 0, y: 150, z: -1000},
74.             look: {x: 0, y: -300, z: -50},
75.             zoomSensitivity: 10,
76.             showCursor: false,
77.             cursorSize: 0.1,
78.
79.             // Clipping
80.             nodes: [{
81.                 type: "translate",
82.                 id: "monClipping",
83.                 x: 0.0,
84.                 y: -350.0, // -78=RDC / -200=1er
85.                 z: 0.0,
86.
87.                 nodes: [{
88.                     type: "clips",
89.                     clips: [{
90.                         x: 0,
91.                         y: 0.1,
92.                         z: 0,
93.                         mode: "inside"
94.                     }],
95.
96.                     nodes: [{
97.                         type: "flags",
98.                         flags: {
99.                             solid: true
100.                        },
101.
102.                        // Rotation
103.                        nodes: [{

```

```

104.                                     type: "rotate",
105.                                     y: 1,
106.                                     angle: 130,
107.                                     nodes: [{
108.                                         type: "material",
109.                                         baseColor: {r:
110.                                             0.5, g: 0.5, b: 0.5},
111.                                         nodes: [{
112.                                             type:
113.                                                 "import/obj",
114.                                             src: "
115.                                                 http://localhost:8888/models/house2.obj"
116.                                             }]}]
117.                                     },
118.                                     // Skybox
119.                                     {
120.                                         type: "skybox/cloudySea",
121.                                         size: 5000
122.                                     }
123.                                 ]
124.                            }]}]
125.                    }];
126.            </script>
127.        </body>
128.    </html>

```



## Annexe 7 : Partie pratique – fichier « houseViewerCologne\_2D\_RDC.html »

```
1. <!--
2.   Auteur   : Ivo de Carvalho Matosinhos
3.   Date    : 28.06.2019
4.   Version : 1.0
5.   Projet  : Travail de Bachelor - HouseViewer
6. -->
7.
8. <!doctype html>
9. <html lang="fr">
10.   <head>
11.     <!-- Required meta tags -->
12.     <meta charset="utf-8">
13.     <meta name="viewport" content="width=device-width, initial-
14.       scale=1, shrink-to-fit=no">
15.     <!-- Bootstrap CSS -->
16.     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap
17.       /4.3.1/css/bootstrap.min.css" integrity="sha384-
18.       ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="an
19.       onymous">
20.     <link rel="stylesheet" href="style.css">
21.     <script src="https://code.jquery.com/jquery-
22.       3.3.1.slim.min.js" integrity="sha384-
23.       q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="an
24.       onymous"></script>
25.     <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/umd/
26.       popper.min.js" integrity="sha384-
27.       U02eT0CpHqSJQ6hJty5KVphtPhzWj9W01c1HTMGA3JDZwrnQq4sF86dIHNDz0W1" crossorigin="an
28.       onymous"></script>
29.     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootst
30.       rap.min.js" integrity="sha384-
31.       JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEeFf/njGzIxFDs4f4x0xIM+B07jRM" crossorigin="an
32.       onymous"></script>
33.     <script src="http://scenejs.org/api/latest/scenejs.js"></script>
34.   </head>
35.   <body>
36.     <!-- Navigation -->
37.     <nav class="navbar navbar-dark bg-dark fixed-top">
38.       <div class="container">
39.         <a class="navbar-brand" href="index.html">House viewer</a>
40.         <button class="navbar-toggler" type="button" data-
41.           toggle="collapse" data-target="#navbarResponsive" aria-
42.           controls="navbarResponsive" aria-expanded="false" aria-
43.           label="Toggle navigation">
44.           <span class="navbar-toggler-icon"></span>
45.         </button>
46.         <div class="collapse navbar-collapse" id="navbarResponsive">
47.           <ul class="navbar-nav ml-auto">
48.             <li class="nav-item">
49.               <a class="nav-
50.                 link" href="houseViewerCologne_3D.html">Maison à Cologne</a>
51.             </li>
52.             <li class="nav-
53.                 link" href="houseViewerCologne_3D.html">Visualisation 3D</a>
54.             </li>
55.             <li class="nav-
56.                 link active" href="houseViewerCologne_2D_RDC.html">Visualisation 2D : Rez-de-
57.                 chaussée
58.             </li>
59.           </ul>
60.         </div>
61.       </div>
62.     </nav>
63.   </body>
64. </html>
```

```

42.             <span class="sr-only">(current)</span>
43.             </a>
44.             <a class="nav-
link" href="houseViewerCognoy_2D_1er.html">Visualisation 2D : 1er étage</a>
45.         </ul>
46.     </li>
47.     <li class="nav-item">
48.         <a class="nav-
link" href="houseViewerSatigny_3D.html">Appartement à Satigny</a>
49.     </li>
50.     <li class="nav-item">
51.         <a class="nav-
link" href="houseViewerMaquette.html">Maquette</a>
52.     </li>
53. </ul>
54. </div>
55. </div>
56. </nav>
57.
58. <script>
59.     SceneJS.setConfigs({
60.         pluginPath: "http://scenejs.org/api/latest/plugins"
61.     });
62.
63.     var scene = SceneJS.createScene({
64.         // Camera
65.         nodes: [{
66.             type: "cameras/pickFlyOrbit",
67.             id: "maCamera",
68.             yaw: 0,
69.             pitch: -90, // -90=vueDuDessus / -20=vue3D
70.             maxPitch: -10,
71.             minPitch: -80,
72.             zoom: 1000,
73.             eye: {x: 0, y: 150, z: -1000},
74.             look: {x: 0, y: 0, z: 50},
75.             zoomSensitivity: 10,
76.             showCursor: false,
77.             cursorSize: 0.1,
78.
79.             // Clipping
80.             nodes: [{
81.                 type: "translate",
82.                 id: "monClipping",
83.                 x: 0.0,
84.                 y: -78.0, // -78=RDC / -200=1er
85.                 z: 0.0,
86.
87.                 nodes: [{
88.                     type: "clips",
89.                     clips: [{
90.                         x: 0,
91.                         y: 0.1,
92.                         z: 0,
93.                         mode: "inside"
94.                     }],
95.
96.                     nodes: [{
97.                         type: "flags",
98.                         flags: {
99.                             solid: true
100.                        },
101.
102.                        // Rotation
103.                        nodes: [{
104.                            type: "rotate",

```

```

105.                                     y: 1,
106.                                     angle: 90,
107.
108.                                     // Modele 3D
109.                                     nodes: [{
110.                                         type: "materia
111.                                         1",
112.                                         baseColor: {r:
113.                                         0.5, g: 0.5, b: 0.5},
114.                                         nodes: [{
115.                                             type:
116.                                             "import/obj",
117.                                             src: "
118.                                             http://localhost:8888/models/house2.obj"
119.                                             }]}]
120.                                     },
121.                                     // Skybox
122.                                     {
123.                                         type: "skybox/cloudySea",
124.                                         size: 5000
125.                                     }
126.                                 ]
127.                             }]}]
128.                         });
129.                     </script>
130.                 </body>
131.             </html>

```

## Annexe 8 : Partie pratique – fichier « houseViewerCologne\_2D\_1er.html »

```
1. <!--
2.   Auteur   : Ivo de Carvalho Matosinhos
3.   Date    : 28.06.2019
4.   Version : 1.0
5.   Projet  : Travail de Bachelor - HouseViewer
6. -->
7.
8. <!doctype html>
9. <html lang="fr">
10.   <head>
11.     <!-- Required meta tags -->
12.     <meta charset="utf-8">
13.     <meta name="viewport" content="width=device-width, initial-
14.       scale=1, shrink-to-fit=no">
15.     <!-- Bootstrap CSS -->
16.     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap
17.       /4.3.1/css/bootstrap.min.css" integrity="sha384-
18.       ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="an
19.       onymous">
20.     <link rel="stylesheet" href="style.css">
21.     <script src="https://code.jquery.com/jquery-
22.       3.3.1.slim.min.js" integrity="sha384-
23.       q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="an
24.       onymous"></script>
25.     <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/
26.       popper.min.js" integrity="sha384-
27.       U02eT0CpHqSJQ6hJty5KVphtPhzWj9W01c1HTMGA3JDZwrnQq4sF86dIHNDz0W1" crossorigin="an
28.       onymous"></script>
29.     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootst
30.       rap.min.js" integrity="sha384-
31.       JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEeAf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="an
32.       onymous"></script>
33.     <script src="http://scenejs.org/api/latest/scenejs.js"></script>
34.   </head>
35.   <body>
36.     <!-- Navigation -->
37.     <nav class="navbar navbar-dark bg-dark fixed-top">
38.       <div class="container">
39.         <a class="navbar-brand" href="index.html">House viewer</a>
40.         <button class="navbar-toggler" type="button" data-
41.           toggle="collapse" data-target="#navbarResponsive" aria-
42.           controls="navbarResponsive" aria-expanded="false" aria-
43.           label="Toggle navigation">
44.           <span class="navbar-toggler-icon"></span>
45.         </button>
46.         <div class="collapse navbar-collapse" id="navbarResponsive">
47.           <ul class="navbar-nav ml-auto">
48.             <li class="nav-item">
49.               <a class="nav-
50.                 link" href="houseViewerCologne_3D.html">Maison à Cologne</a>
51.             </li>
52.             <li class="nav-item">
53.               <a class="nav-
54.                 link" href="houseViewerCologne_3D.html">Visualisation 3D</a>
55.             </li>
56.             <li class="nav-item">
57.               <a class="nav-
58.                 link" href="houseViewerCologne_2D_RDC.html">Visualisation 2D : Rez-de-
59.                 chaussée</a>
60.             </li>
61.           </ul>
62.         </div>
63.       </div>
64.     </nav>
65.   </body>
66. </html>
```

```

42.         <a class="nav-
link active" href="houseViewerCologne_2D_1er.html">Visualisation 2D : 1er étage
43.             <span class="sr-only">(current)</span>
44.         </a>
45.     </ul>
46. </li>
47. <li class="nav-item">
48.     <a class="nav-
link" href="houseViewerSatigny_3D.html">Appartement à Satigny</a>
49. </li>
50. <li class="nav-item">
51.     <a class="nav-
link" href="houseViewerMaquette.html">Maquette</a>
52. </li>
53. </ul>
54. </div>
55. </div>
56. </nav>
57.
58. <script>
59.     SceneJS.setConfigs({
60.         pluginPath: "http://scenejs.org/api/latest/plugins"
61.     });
62.
63.     var scene = SceneJS.createScene({
64.         // Camera
65.         nodes: [{
66.             type: "cameras/pickFlyOrbit",
67.             id: "maCamera",
68.             yaw: 0,
69.             pitch: -90, // -90=vueDuDessus / -20=vue3D
70.             maxPitch: -10,
71.             minPitch: -80,
72.             zoom: 1000,
73.             eye: {x: 0, y: 150, z: -1000},
74.             look: {x: 0, y: 0, z: 50},
75.             zoomSensitivity: 10,
76.             showCursor: false,
77.             cursorSize: 0.1,
78.
79.             // Clipping
80.             nodes: [{
81.                 type: "translate",
82.                 id: "monClipping",
83.                 x: 0.0,
84.                 y: -200.0, // -78=RDC / -200=1er
85.                 z: 0.0,
86.
87.                 nodes: [{
88.                     type: "clips",
89.                     clips: [{
90.                         x: 0,
91.                         y: 0.1,
92.                         z: 0,
93.                         mode: "inside"
94.                     }],
95.
96.                     nodes: [{
97.                         type: "flags",
98.                         flags: {
99.                             solid: true
100.                        },
101.
102.                        // Rotation
103.                        nodes: [{
104.                            type: "rotate",

```

```

105.                                     y: 1,
106.                                     angle: 90,
107.
108.                                     // Modele 3D
109.                                     nodes: [{
110.                                         type: "materia
111.                                         1",
112.                                         baseColor: {r:
113.                                         0.5, g: 0.5, b: 0.5},
114.                                         nodes: [{
115.                                             type:
116.                                             "import/obj",
117.                                             src: "
118.                                             http://localhost:8888/models/house2.obj"
119.                                             }]}
120.                                     ]}
121.                                     },
122.                                     // Skybox
123.                                     {
124.                                         type: "skybox/cloudySea",
125.                                         size: 5000
126.                                     }
127.                                     ]
128.                                     });
129.                                     </script>
130.                                     </body>
131.                                     </html>

```

## Annexe 9 : Partie pratique – fichier « style.css »

```
1. /*
2.   Auteur   : Ivo de Carvalho Matosinhos
3.   Date    : 28.06.2019
4.   Version : 1.0
5.   Projet  : Travail de Bachelor - HouseViewer
6. */
7.
8. #houseViewer{
9.   width: 100%;
10.  float: left;
11.  border: 0px;
12. }
13.
14. .carousel-item {
15.  height: 65vh;
16.  min-height: 350px;
17.  background: no-repeat center center scroll;
18.  -webkit-background-size: cover;
19.  -moz-background-size: cover;
20.  -o-background-size: cover;
21.  background-size: cover;
22. }
```