Mini-projet WebGL/Three.js.

Objectif : vous devez créer un mini-jeu « sérieux » pour faire découvrir une anecdote, un objet particulier, un détail de décoration, etc. à un jeune public.

Pour cela, vous devrez faire déplacer un personnage en vue subjective à l'intérieur d'un environnement clos - une partie de la Cathédrale d'Amiens - et lui faire chercher des indices lui permettant de résoudre une énigme et de parvenir au but final.

Construction de la scène et des Objets 3D

Construction du décor

On part sur la base de ce qui a été vu en première séance de TD :

- chargement de l'objet, redressement en position verticale

- chargement de la ou des textures : au moins la texture principale représentant le transept sud, et éventuellement une texture utilisée pour générer un effet de Bump-Mapping (http://fr.wikipedia.org/wiki/Placage_de_relief) ou encore une autre pour gérer l'effet de réflexion (http://en.wikipedia.org/wiki/Specularity)

- utilisation d'un lumière ambiante, ou de plusieurs lumières ponctuelles

Groupes d'objets 3d

Les objets utilisés dans une scène peuvent être ajoutés un par un (méthode Scene.*add*) et donc manipulés individuellement. On peut également ajouter des objets à d'autres objets pour constituer un groupe. L'avantage est ici de pouvoir modifier les paramètres de position/rotation/visibilité de tous les objets du groupe en une seule opération. On peut en outre faire modifier des paramètres individuellement aux objets à l'intérieur du groupe (dont position et rotation)

Exemple :

1 Personne = 1 torche + 1 camera

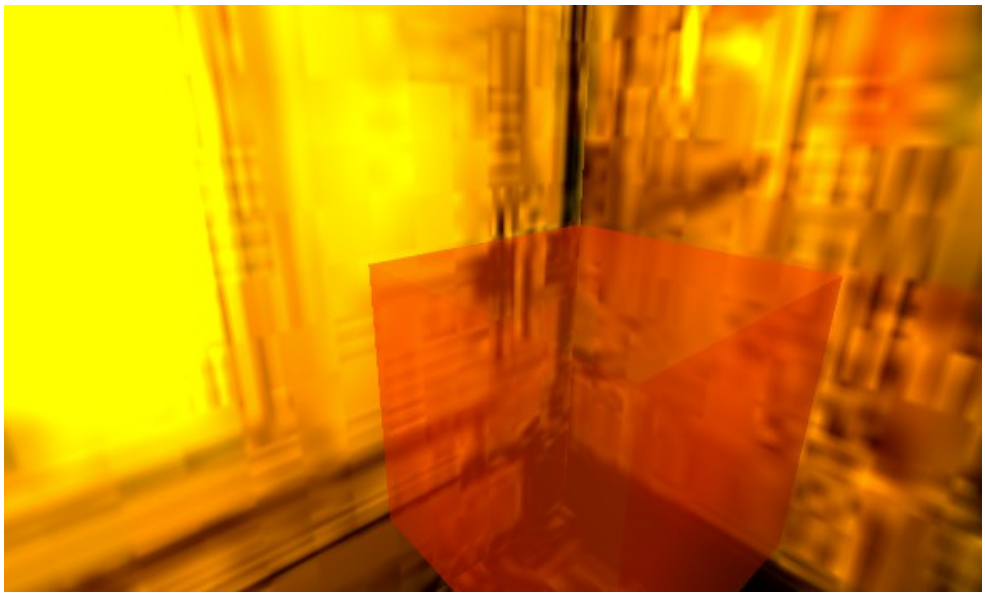
1 Indice = un cube + un message

etc.

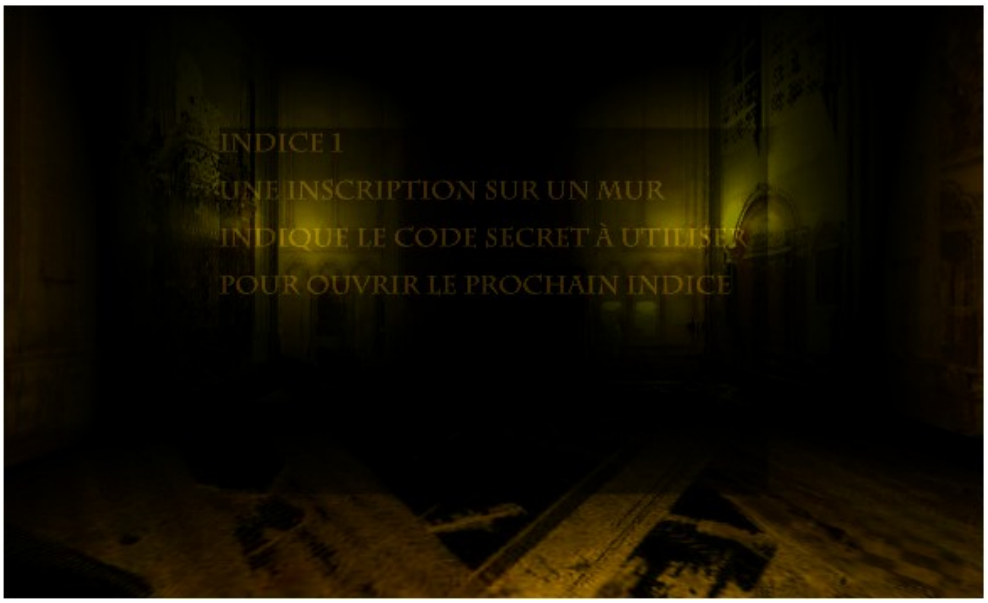
Disposition et rôle des indices

Les indices peuvent être symbolisés par des détails sur la texture (du texte ajouté par exemple), des objets virtuels apparaissant lors du passage du joueur à proximité (un cube, un point d'interrogation pivotant, etc.).

Les indices peuvent indiquer à l'utilisateur l'emplacement d'un autre indice ; un code ou une partie de code à taper pour découvrir l'indice final ; la combinaison de touches à utiliser (par exemple SHIFT/ALT) pour l'aider à trouver d'autres modes de déplacement (orientation tête/déplacement latéral) ; une touche permettant de voir du point de vue d'une caméra disposée en hauteur pointant sur un autre indice, etc. Soyez inventifs.



Un cube apparaissant au passage du joueur à proximité



Un indice en surbrillance sur un panneau transparent

Déplacement du personnage (i.e. de la caméra+torche)

Principe

Pour donner une impression de déplacement « à la première personne », on fait varier la caméra dans la scène, ainsi qu'une lampe qui serait tenue à la main (cf plus haut *Groupes d'objets*)
La plupart des fonctions de la classe Object3D suffiront à assurer cette tâche.
http://threejs.org/docs/#Reference/Core/Object3D

Gestion des touches du clavier

Les 4 touches directionnelles sont utilisées pour avancer/reculer (touches haut, bas), effectuer une rotation à gauche/droite (touches gauche/droite).

Le principe que l'on utilise généralement pour gérer le déplacement est de capturer les événements de touches enfoncées ou relâchées, et de mettre à jour des *flags*

indiquant quelles sont les touches actuellement enfoncées ou non

Pour cela, on utilise les événements Javascript standards, avec les lesquels on récupère le code de la touche à l'aide de l'attribut `keyCode`.

Dans la boucle d'animation, on se sert des flags pour modifier les coordonnées du personnage (donc de la caméra).

Les codes des touches directionnelles sont 37 (←), 39 (→), 38 (↑) et 40 (↓).

L'attribut altKey permet de savoir si la touche ALT a été enfoncée en même temps : on peut ainsi imaginer imaginer un déplacement latéral (straff) en combinant les touches ALT et ←/ → .

De même la combinaison SHIFT et ↑/↓ (attribut shiftKey) peut être utilisée pour orienter la caméra de haut en bas. Pour ce cas particulier, vous veillerez à faire une rotation uniquement sur la caméra et non sur le personnage.

Affichage de texte ou de vidéo (opt.)

Three.js propose un objet *PlaneGeometry* permettant d'afficher un simple rectangle, que l'on peut utiliser comme support pour afficher une texture particulière, notamment du texte ou de la vidéo.

Pour l'affichage de texte, on peut créer dynamiquement (i.e en js) un <canvas> sur lequel on va écrire le texte. On se servira alors de ce canvas comme texure d'un matériau de type Phong ou Lambert, appliqué au rectangle.

Pour la vidéo, il faut d'abord créer un <canvas> en html, puis utiliser le canvas en tant que texture d'un matériau, comme pour le texte. En revanche dans la fonctiuon d'animation, il faudra redessiner systématiquement l'image de la vidéo sur le canvas, et demander à three.js de mettre à jour la texture.

Travail à rendre

Au minimum, afficher la structure texturée, déplacer la caméra à l'aide du clavier

Option 1 : limiter le déplacement à la structure (pas de possibilité de sortie)
Option 2 : associer une lumière à la caméra, donnant l'effet d'une torche
Option 3 : ajouter des zones d'indices et panneaux d'informations
Option 4 : ne faire apparaître les indices que lors du passage à proximité
Option 5 : utiliser plusieurs caméras

E.M. Mouaddib & D.Durand