Rapport du projet CRV

Cindy Even (c9even@enib.fr)

Guillaume Biannic (g9bianni@enib.fr)

Décembre 2014

Sommaire

Introduction :

Dans le cadre du module de CRV (Comportements Anthropomorphes en Réalité Virtuelle et Augmentée) il nous a été demandé de développer une simulation afin de mettre en pratique les notions étudiées en cours.

Voici l’énoncé du projet :

*On considère un musé contenant une quarantaine de tableaux (ou posters, vieilles cartes postales, …). Il s’agit de guider l’utilisateur dans ce musée en tenant compte de ses préférences. Plutôt que d’utiliser un système de fléchage (dynamique ou non) ou d’incarner un système de guidage par un acteur virtuel, on préfère ici utiliser les autres visiteurs (représentés par des acteurs virtuels) pour inciter l’utilisateur à se diriger vers un tableau plutôt qu’un autre.*

L’hypothèse qui est faite ici est la suivante :

*Un attroupement de visiteurs devant un tableau doit inciter l’utilisateur à aller vers ce tableau.*

Nous avons choisi de travailler sur ce projet à l’aide du logiciel Unity3D. Dans ce rapport nous vous présentons le travail que nous avons effectué ce semestre sur ce projet. Dans un premier temps nous vous présentons le logiciel Unity3D et les différents composants du logiciel que nous avons utilisé dans notre application. Ensuite nous expliquons comment utiliser l’interface de la simulation. Enfin nous consacrons une partie pour chacune des questions du projet qui nous a été posé en détaillant les différentes méthodes et algorithmes que nous avons utilisés pour répondre à ces questions.

1. Présentation de Unity3D

Unity3D est un système multiplateforme de création de jeu vidéo développé par Unity Technologies. Ce système comprend un moteur de jeu et environnement de développement intégré (IDE) et utilise entre autre le moteur physique PhysX de Nvidia. Afin de faciliter la compréhension du travail que nous avons réalisé, voici un petit glossaire comprenant différents éléments importants utilisés dans Unity3D :

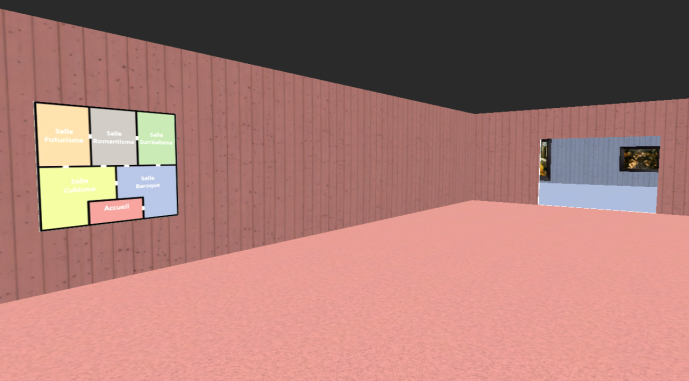
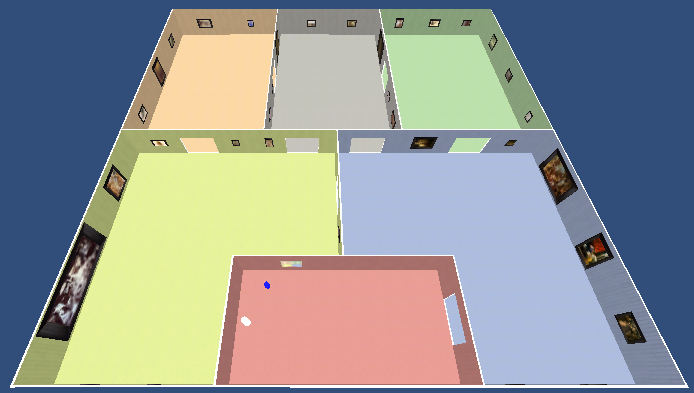
- GameObjet (GO) : ce sont les entités les plus importantes dans Unity3D. Tout objet dans l’application est un GO, cependant, ils ont besoin de composants (qui héritent de la classe Component) pour se différencier les uns des autres. On peut voir un GO comme une boîte vide à laquelle il faut ajouter des composants pour en faire un personnage, une lumière, ou des effets spéciaux par exemple.

- Component : les composants (ou Components) sont les pièces fonctionnelles d’un GO. Certains existent déjà dans Unity3D, mais il est possible de créer nos propres composants à l’aide des Scripts.

Voici la liste des composants utilisés dans notre application :

- Transform : c’est le seul composant que tout GO possède. Il permet de définir la position, la rotation et l’échelle du GO dans la scène. Ce composant permet aussi le Parenting, ce concept permet d’attacher un GO à un autre GO, le premier étant alors enfant du second. Un enfant héritera alors du mouvement et de la rotation de son parent.

- Camera : il permet de capturer et d’afficher le monde pour l’utilisateur. Il peut y avoir plusieurs caméras dans la scène mais une seule peut être active à la fois. Pour la visite du musée nous avons deux points de vue (et donc deux caméras). L’un est le point de vue du visiteur (First Person View) et le second permet de voir le musé en entier. Pour le FPV il suffit d’attacher la caméra au visiteur et de la placer au niveau des yeux.

- Scripts : comme dit précédemment, Unity nous permet de créer nos propres composants à l'aide de scripts. Pour cela il suffit de créer une nouvelle classe héritant de MonoBehaviour. Nous pourrons alors utiliser les méthodes Start(), Update() et autres pour implémenter les fonctionnalités voulues.

- Start() est appelée à la toute première frame.

- Update() est appelée toutes les frames.

- OnGUI() est appelée pour le rendu et la gestion des événements de l'interface graphique.

Nous pouvons aussi créer des Scripts contenant des classes n’héritant pas de MonoBehaviour et ne seront alors pas considérés comme des composants mais comme des classes normales.

- Rigidbody : c’est le composant principal qui permet à un objet d’être soumis aux lois de la Physique. En ajoutant ce composant à un objet, ce dernier sera immédiatement répondre à la gravité. Il nous permet aussi d’appliquer des forces sur l’objet.

- Collider : ce composant permet de définir la forme utilisée pour les collisions. Plusieurs formes sont possibles comme un cube, une sphère, un cylindre, ou même la forme même de l’objet : le mesh. Plus la forme est simple et plus les calculs de collisions seront rapides.

Il est aussi possible d’utiliser le moteur physique pour détecter quand un Collider pénètre dans un autre Collider sans créer de collision. Pour cela il suffit d’utiliser la propriété Trigger du Collider. L’objet ne se comporte alors pas comme un objet solide et permettra simplement d'autres Colliders de passer à travers. Quand un Collider entre son espace, ceci va appeler la fonction OnTriggerEnter().

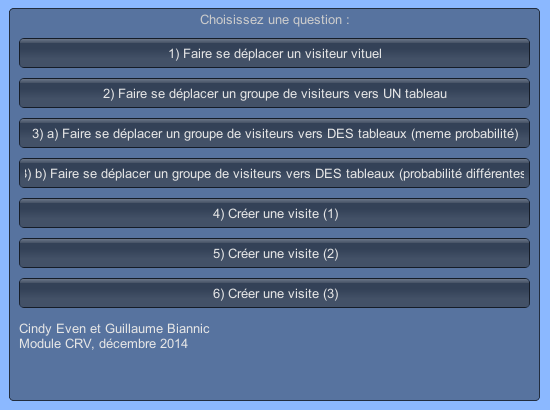
- Tag : Un tag est un mot qui permet d’accéder à un ou plusieurs GO. Nous pouvons les utiliser dans les Scripts pour trouver un GO qui contient le Tag souhaitée. Ce résultat est obtenu en utilisant la fonction GameObject.FindWithTag().

Ces explications devraient être suffisantes pour comprendre la suite mais pour des informations complémentaires il est possible de consulter la documentation d’Unity3D : <http://docs.unity3d.com/ScriptReference> et <http://docs.unity3d.com/Manual>.

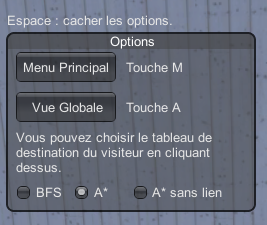
1. Présentation de l’interface

Pour lancer l’application, double cliquez sur le fichier exécutable. Une fenêtre s’ouvre et vous invite à choisir la configuration qui vous convient. Une fois la configuration faite, cliquez sur le bouton .

Le menu principal s’affiche alors. A partir de ce menu vous pouvez accéder à différentes parties de la simulation et quitter l’application. Afin que vous puissiez voir notre travail étape par étape, nous avons séparé les questions. A chaque question correspond un bouton dans le menu (voir image ci-dessous). Pour voir notre travail sur la première question : Faire se déplacer un visiteur virtuel, il suffit de cliquer sur le bouton associé.



Après avoir choisi une question, vous allez vous trouver dans le hall du musée dans la peau d’un visiteur. A tout moment vous pouvez afficher ou cacher le panneau des Options en cliquant sur la touche Espace (voir figure ci-dessous).



A partir de ce tableau vous pouvez :

\* Revenir au menu principal

\* Changer la vue de la simulation

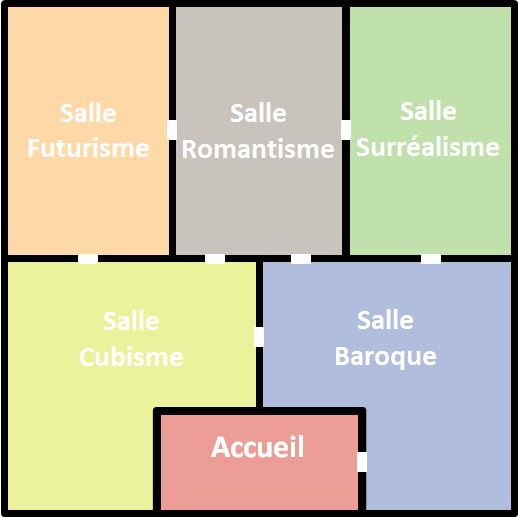
\* Modifier d’autres paramètres qui sont spécifiques à la question courante.

Notez que à tout moment de la simulation vous pouvez revenir au menu principal grâce à la touche M et passer de la vue globale à la vue visiteur et vis-versa avec la touche A.

Les options spécifiques seront détaillées dans les parties du rapport suivantes.

1. Création du musée :
   1. Objets 3D constituant la scène :

Les modèles 3D du musée et d’un tableau ont été construits à l’aide d’Autodesk 3ds Max qui est un logiciel d'animation, de rendu et de modélisation 3D. Le musée est constitué de 6 sales comme indiqué sur le plan ci-dessous :

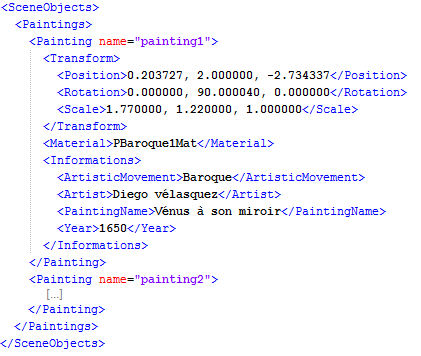


La salle Accueil est la salle où commence la simulation. C’est à partir de cette salle que tous les agents virtuels et le visiteur commencent la visite du musée. Les cinq autres salles contiennent dix tableaux chacune, répartis de façon uniforme. Chaque salle représente l’un de ces mouvements artistiques : baroque, cubisme, futurisme, romantisme, surréalisme.

Un tableau possède une face avant et une face arrière différente l’une de l’autre.

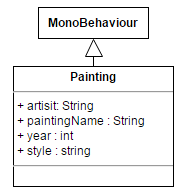


Le modèle 3D du tableau possède deux matériaux (Materials dans Unity3D) : un pour le cadre et un autre pour la peinture. De ce fait, il nous suffit de remplacer le Material de la peinture et de modifier les dimensions du tableau pour créer de nouveaux tableaux dans le musée. Afin d’économiser du temps nous avons choisi d’automatiser ce processus de création et positionnement des tableaux dans le musée. Pour ce faire, nous avons créé un fichier XML contenant toutes les informations nécessaires pour tous les tableaux. Voici la structure du fichier XML :



Pour lire et écrire dans ce fichier XML nous avons implémenté une classe (PaintingsAutoInitInspector) qui utilise différentes classes de l'espace de noms System.Xml fournissant une prise en charge standard du traitement XML.

Nous avons modifié l’interface d’Unity3D en ajoutant deux boutons (ces modifications sont possibles grâce à la classe Editor). Le premier bouton «Instanciate Paintings» permet d’instancier tous les tableaux de la scène, instances de la classe Painting (voir représentation UML ci-dessous), avec la position, rotation, taille et Material définit dans le fichier XML. S’il y a un problème avec un tableau (se trouve au mauvais endroit, a les mauvaises dimensions, …) nous pouvons corriger le problème directement dans Unity3D et cliquer sur le second bouton «Save modifs» pour rectifier l’erreur dans le fichier XML.



* 1. Déplacements de l’utilisateur

L’utilisateur peut se déplacer dans le musée grâce aux touches du clavier :

↑ ou Z pour avancer

↓ ou S pour reculer

← ou Q pour se déplacer latéralement sur la gauche

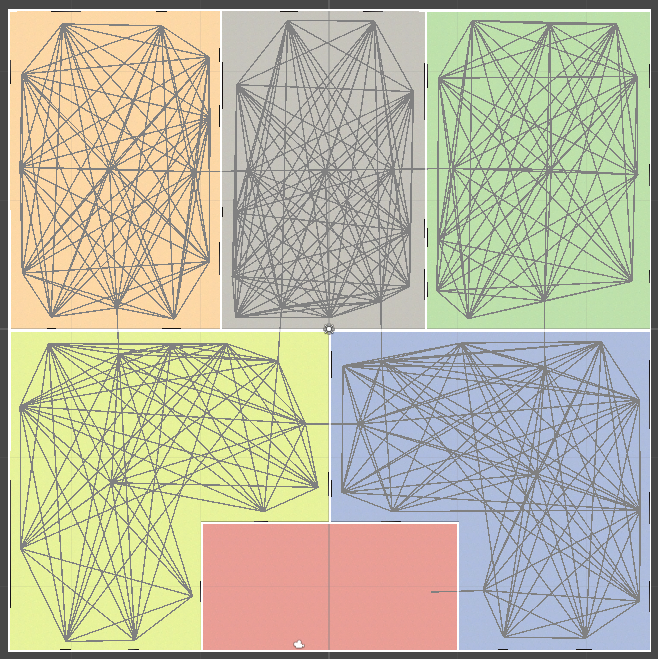
→ ou D pour se déplacer latéralement sur la droite.

Le visiteur se tourne de façon à suivre les mouvements de la sourie sur l’axe horizontale. Et la caméra suit les mouvements de la sourie sur l’axe vertical.

Ces comportements sont implémentés dans les classes VisitorMouvement et MouseLook.

Lorsque l’utilisateur est suffisamment proche d’un tableau et que le curseur de la sourie est au-dessus de ce tableau, des informations sur ce dernier s’affichent (nom de l’artiste, titre du tableau et année de sa création).

* 1. Graphe de navigation



Pseudo code ou diagramme UML de la classe Point et dire que on a fait la même chose que pour les tableaux (XML + classes)

1. Faire se déplacer un visiteur virtuel :

Parcours de graphe

BFS

A\*

A\* sans lien

1. Faire se déplacer un groupe de visiteurs virtuels vers UN tableau :

Steering (Boids)

1. Faire se déplacer un groupe de visiteurs virtuels vers DES tableaux :

Algo de sélection roulette

Pour a) tous les tableaux ont la même fitness

Pour b) la fitness varie en fonction des tableaux.

1. Créer une visite (1)
2. Créer une visite (2)
3. Créer une visite (3)

Conclusion

Bibliographie