Cours sur Unreal Engine 4

1.0 Introduction :

Date de création : 09/08/18  
Dernière modification : 11/08/18   
Version compatible avec ce cours : Toute les versions supérieur à la 4.0  
Nombre de chapitre du cours : 4  
Nombre de Pages : 4

Unreal Engine est un moteur de jeu, créé par Epic Games, permettant de développer des jeux sur différent support et avec plusieurs possibilités de création. Il est orienté pour les jeux à la 1ère personne même s’il tant à se diversifier avec la 3ème personne et la 1ère personne sur une voiture (même s’il est possible de créer un jeu à la 1ère personne avec une voiture, non-dépendant du contrôleur). Il est très connu pour avoir créé des jeux comme Fornite ou Unreal Tournament. Il peut s’associer à tous les styles de jeux vidéo : combat ou action comme Fornite ou horreur comme Outlast 2. Egalement, il est adapté pour la création de jeu 3D mais un peu moins pour la 2D.

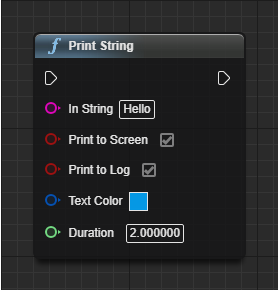
1.0.1 Support d’utilisation :

Le moteur de jeu est disponible sur Linux (Ubuntu), Windows (version supérieur à Windows 7) et Apple (supérieur à OSX 10).

1.0.2 Histoire du moteur

Unreal Engine est à la base fondée sur une startup, elle-même crée en 1991 sous le nom d’Epic Games par Tim Swenney. L’éditeur est publié le 22 mai 1998 par GT-Interactive.

2.0 Principe de programmation

Le langage n’est ni plus ni moins un script, L’UnrealScript crée afin de permettre a ce langage la possibilité d’être plus simple et plus robuste. Le support de création des composants est appelé Blueprint.   
Le code se présente comme ceci :   
  


Cette exemple permet de voir comment s’organise un blueprint ;  
Le signe premier > indique qu’il doit être relié à un brun. Si l’on veut déclencher l’événement « Print String » (écrire sur l’écran), ce brun doit être relié à un événement quel qu’il soit.   
Le deuxième > permet de relier cette événement à un deuxième événement. L’analyse de « Print String sera plus détaillé dans sa partie qui lui sera consacré »

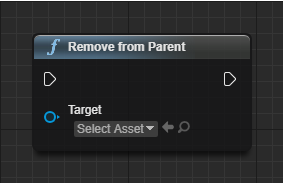
2.1 Programmation Orienté Objet :

La POO est une méthode de programmation qui permet de rassembler des objets avec leurs parents, de faire communiquer des classes avec des objets et vice-versa, de faire communiquer des tableaux avec des objets et de faire communiquer des objets entre eux. Justement dans Unreal Engine, le travail est mâché : grâce à une communication par chevron « < ; > » les objets sont directement reliés entre eux. Mais il est impossible dans certains cas de faire communiquer des parents leurs enfants ou héritage de classes. Du coup une seconde méthode est créée et il s’agit de la méthode « Parents To Child » qui grâce a un blueprint rassemble Parents et Enfant et simplifie l’héritage de classes.

2.2 Programmation Parents To Child :

La PPC est une méthode secondaire dans Unreal Engine qui permet de faire communiquer Parents et Enfants entre eux ainsi que l’aide à l’héritage de classe.   
L’exemple le plus flagrant est la fonction « Remove From Parents » qui permet de supprimer un enfant depuis son parent dans la classe Widget.

Ici la fonction Target permet de faire communiquer la classe Widget (Celui de notre choix) avec l’événement « Remove from parents » et donc de supprimer le Widget



La programmation Parents To Child permet donc de donner une action à un, voire des enfants depuis le parent.   
/!\ A noter qu’un enfant ne peut avoir une action sur son parent, exceptée la demande d’information ainsi qu’un héritage possible de classe que s’il s’apparente à cette dernière. Il ne peut ni modifier ni supprimer son parent. /!\

3.3 Programmation numéraire :

La programmation numéraire (qui s’utilise dans Unreal Engine 4) est une méthode de programmation qui revient à mélanger les classes numéraire et non linéaire. Dans ces classes, on retrouve les « Floats », nombre à virgule, les « Integer », nombre entier et les « byte », 0 ou 1. Il reste également les coordonnées qui sont numéraires mais ce sera pour plus tard. Donc cette méthode de programmation revient à la possibilité de les convertir entre eux toujours avec un Blueprint qui leur correspond.

La première chose à savoir est qu’Unreal à un code couleur pour les classes de données. Les float sont colorés en vert très clair, les integer sont colorés en vert foncé et les bytes en vert très foncé. Le code est compliqué mais on si fait très facilement.

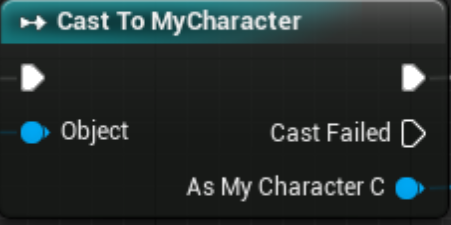
Dans Unreal Engine on utilise énormément les nombres de quelque classe qu’ils soient. Ce sont des donnés qui font partie de la gestion du jeu crée (inventaire, point de vie, point de santé).

4.0 Blueprint et Programmation :

Dans Unreal Engine, un Bleuprint permet d’ajouter du contenu à un objet quel qu’il soit. Par exemple pour faire tourner une porte par la droite à 90°, on réglera cela dans le Blueprint de l’objet et non celui du contrôleur. Par contre c’est celui-ci qui devra cliquer pour effectuer l’action.

Le Blueprint du contrôleur est différent de celui d’un objet. On définit les actions d’un objet et celui-ci n’a pas le choix et doit les respecter. Par contre le contrôleur est libre comme une intelligence artificielle et encore elle n’effectuera que les possibilités qu’on lui a données.   
Le contrôleur est à chaque fois le parent. S’il est appelé il peut provoquer le partage ainsi que l’héritage de classes. Du coup tous les enfants pourront communiquer avec lui en allant chercher l’information auprès du parent. Cette fonction est appelé « cast to » c'est-à-dire « chercher auprès de… ».

Donc ici, nous avons la fonction Cast To MyCharacter. /!\ Le nom est « MyCharacter » seulement parce que le créateur lui à donner ce nom.



La variable Set appartient à MyCharacter. Pour la récuperer, on appliquer la programmation Parents to Child, et on récupère la variable. Et l’option Target doit être reliée à l’option « As My Character » afin de la déclarer. Sinon on appelle cela une « blank variable », c'est-à-dire une variable qui ne sert pas car elle n’est pas déclarer comme appartenant à « MyCharacter ». Alors, nous récupérons les enfants depuis les parents. Comme le parent est un Blueprint et que l’enfant est une variable, il se produit un phénomène nommé « Héritage de Classe ».

5.0 Les Booléens :

Peut-être que cela ne mérite pas un chapitre entier, mais il ne peut pas aller dans un sous-chapitre.

Donc les Booléens constituent une classe très importantes dans l’informatique car c’est à partir d’eux que le code sera constitué en choix. Les booléens sont extrêmement très proche des Bytes mais au lieu de contenir une donnée numéraire (0 ou 1), les booléens contiennent une donnés particulière : True ou False (Vrai ou Faux).

Du coup, ils peuvent définir si un événement va se produire ou non. C’est une catégorie appart parce qu’au lieu de contenir une information binaire (0 ou 1), une information numéraire (nombre) ou une donnée linéaire (string ou char) elles contiennent une information True-False qui définissent donc l’action cruciale de l’utilisateur ou d’une IA. Dans un jeu c’est important de définir s’ils peuvent faire telle ou telle actions. Par contre ils peuvent être modifiables en cours de partie. Par exemple si un joueur à terminer une action et que la porte à besoin d’être déverrouiller et bien la variable booléenne peut s’activer et donc la porte se déverrouille.



La case n’est pas cochée donc la variable est sur False

Ici nous avons ce qu’on appelle « un Set Booléen ». C’est une fonction qui permet de régler la variable « rotate Camera » sur soit True ou False. Si c’est True alors la case sera cochée. Sinon (False), la case n’est pas cochée.

Interface d’Unreal Engine 4 :