M2 Ingénieurie du jeu Vidéo
Gamagora - ICOM 2024/2025

Module Bases des moteurs de jeux (Unreal Engine)

Travaux Pratiques 3

Révisions *Blueprint*, intégration dans la scène et fractures d'objets

(illustrations réalisées sous UE5 version 5.4.4)

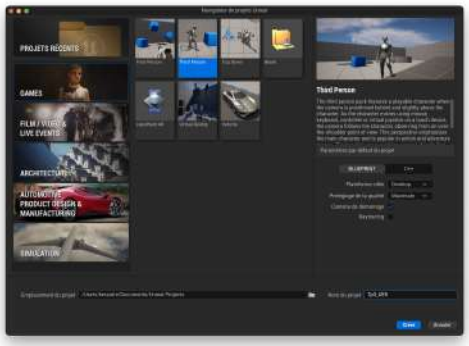
Objectif: Après avoir découvert l'interface d'*Unreal Engine* et fait des premiers programmes *Blueprint* en mode console, nous allons réviser les différents éléments du cours en lien avec la programmation *Blueprint*. Ensuite, nous verrons comment intégrer du code dans le ***Level Blueprint*** en rajoutant des éléments dynamiques dans la scène. Dans une troisième partie, nous verrons comment utiliser les mode *Fracture* introduit depuis *EU5.0*

Partie I: Révision programmation et suite *Blueprint*

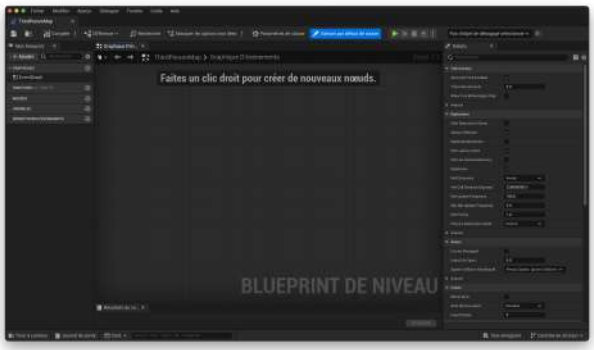
Afin de réviser les principales notions de *Blueprint* vues en cours, nous allons les utiliser à travers la création et l'accès à des éléments d'un tableau. Dans cette partie, les notions suivantes seront traitées :

- Révisions sur les manipulations de tableaux et parcours itératifs.
- Déclenchement d'évenement au lancement du jeu.
- Révision de la définition et l'utilisation d'une `fonction`.
- Utilisation d'une *Blueprint* Map.

1.1 Créez un nouveau projet associé à cette séance en utilisant le modèle *Third Person*(image (a)) et les réglages par défaut.



(a)



(b)

1.2 Ouvrez l'éditeur du ***Level Blueprint*** à partir de l'icone *Blueprint*. Comme vous pourrez le constater (image (b) ci-dessus), l'éditeur est vierge.

1.3 A partir d'un clique droit sur la zone d'édition de code, créez un nouveau noeud associé à l'évènement de lancement du jeu (appelé `EventBeginPlay`) de façon à créer un point d'entré qui sera exécuté à chaque lancement du jeu.

1.4 Comme vu en cours, utilisez la fonction `PrintString` pour afficher le message suivant lors du lancement du jeu :

1.5 Toujours dans le ***Level Blueprint***, rajoutez une variable nommée `tabPrenom` de type tableau de `Chaîne` contenant un tableau de prénoms (voir image (a) ci-dessous).

(a)

(b)

1.6 Ajoutez 6 prénoms manuellement dans le tableau dans le champ *default value* de la variable.

1.7 En général, il est plus fréquent de rajouter des éléments dans un tableau que d'utiliser l'initialisation par défaut. Dans le code du ***Level Blueprint***, rajoutez votre prénom dans le tableau en deuxième position.

1.8 En utilisant une boucle `foreach`, générez automatiquement l'affichage de tous les prénoms comme illustré sur l'image (b) ci-dessus.

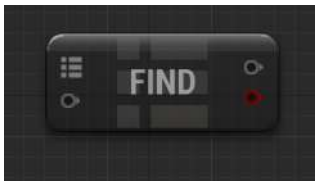
1.9 On souhaite désormais aussi afficher la position dans laquelle se trouve le prénom dans le tableau. Utilisez le champ *Array index* de façon à obtenir un affichage ressemblant au suivant :

1.10 Regrouper le code lié à l'affichage de tous les étudiants dans une fonction unique. Il s'agira d'afficher le tableau passé en paramètre comme sur l'image suivante :

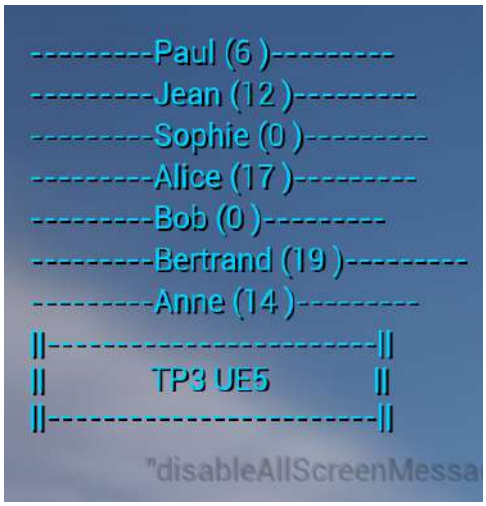
1.11 Dans le ***Level Blueprint***, on souhaite stocker, pour certains étudiants, une note en utilisant une *carte* qui associera chaque prénom à un entier. De la même façon que pour la question 1.5, rajoutez une nouvelle variable que vous nommerez `notesEtu` et qui associera une `Chaîne` à un `Integer`.

1.12 Dans les valeurs par défaut de la variable, associer des notes (sans 0) pour différents prénoms et rajouter une fonction qui cette fois sera destinée à afficher les notes (question suivante).

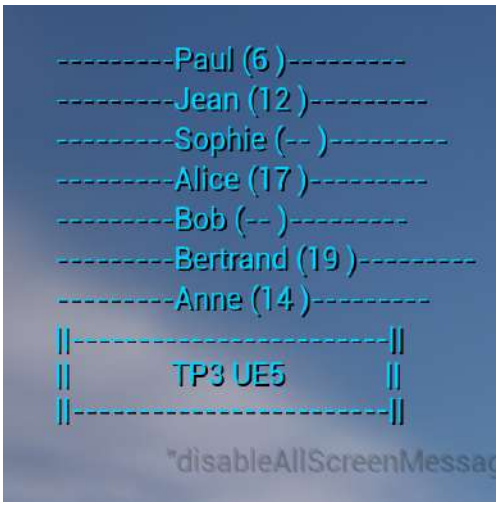
1.13 En utilisant la fonction `find` afficher les notes des étudiants.



1.14 Comme vous pouvez le voir sur l'image (a) ci-dessous, les étudiants qui n'ont pas de note ont 0 affiché à la place. Améliorez votre code de façon à mettre des symboles "-" si l'étudiant n'a pas de note (comme sur l'image (b)).



(a)



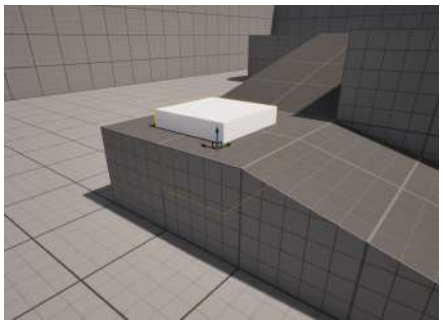
(b)

Partie II: Rajout d'éléments dynamiques dans la scène

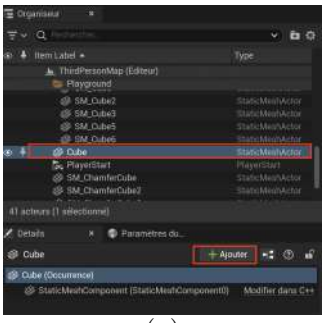
Dans cette partie, nous allons principalement créer du code dans la partie ***Level Blueprint***. Dans un premier temps, nous allons rajouter des lanceurs permettant à l'acteur de se déplacer très haut dans les airs ou de se téléporter. Les objectifs en *Blueprint* sont les suivants :

- Savoir récupérer l'instance d'un joueur.
- Appliquer une transformation géométrique.

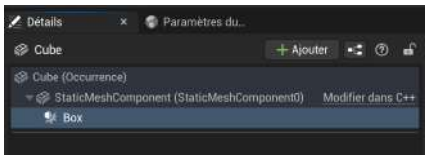
2.1 Dans votre scène, rajoutez un cube en haut des escaliers. Enfoncez le cube de façon à ce qu'il apparaisse comme le lanceur illustré sur la page suivante.



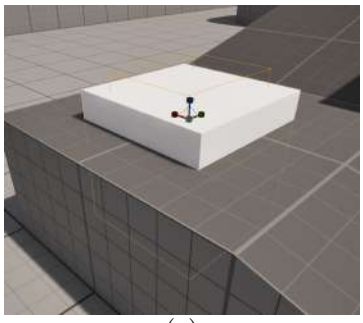
2.2 On souhaite maintenant pouvoir rattacher à cet un objet un *box collision* de façon à détecter le passage du joueur. Comme il est plus pratique que de déplacer les deux objets en même temps, il s'agit de rattacher la *box collision* sur l'objet initial. Pour cela, il faut sélectionner l'objet, puis lui rajouter l'objet la box à partir du bouton vert *+Add Component* (voir images (a)).



(a)



(b)



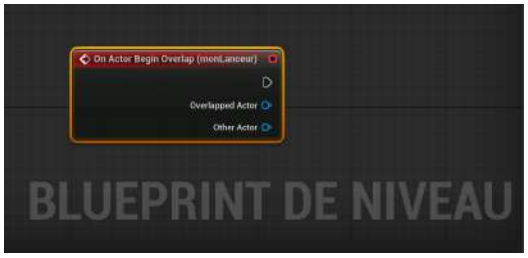
(c)

2.3 Une fois la *box collision* rajoutée, vous devez la voir rattachée à votre cube comme sur l'image (b). Sélectionnez la *box collision* dans la hiérarchie de l'image (b) et la déplacer de façon à ce qu'elle soit au dessus du cube (image (c)). Maintenant, quand vous déplacerez votre cube, la *box collision* sera elle aussi déplacée avec le même écart relatif au cube.

Maintenant que nous avons construit le détecteur de passage, il ne reste plus qu'à l'associer à un évènement et à programmer l'action associée (saut en hauteur) dans le ***Level Blueprint***.

2.4 Pour créer un nouvel événement dans le ***Level Blueprint***, il faut sélectionner le cube puis allez dans l'éditeur de ***Level Blueprint*** et à partir d'un clic droit l'ajout d'un événement lié au cube sera proposé (image (a) cu dessous).

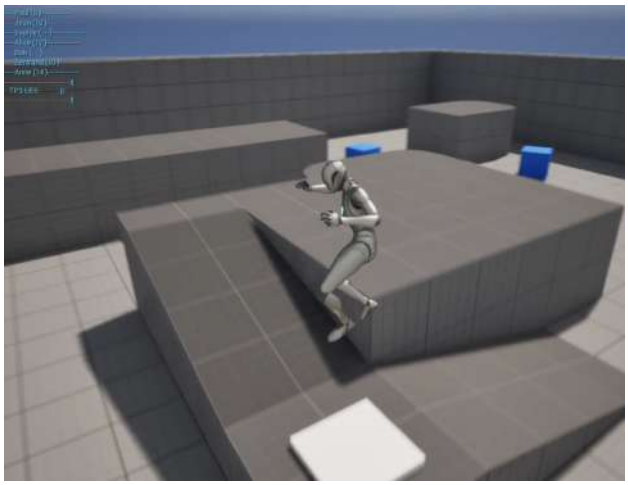
(a)



(b)

2.5 Rajoutez des instructions de façon à pouvoir utiliser la fonction `LaunchCharacter` qui permet de lancer un joueur à une certaine vitesse. Vous aurez besoin d'utiliser la fonction `CastToCharacter` qui permet de convertir l'acteur détecté.

2.6 Testez le bon fonctionnement du lanceur en déplaçant le personnage sur la zone détectée. Si tout fonctionne comme prévu vous devriez pouvoir sauter plus haut que les murs comme illustré sur l'image suivante :

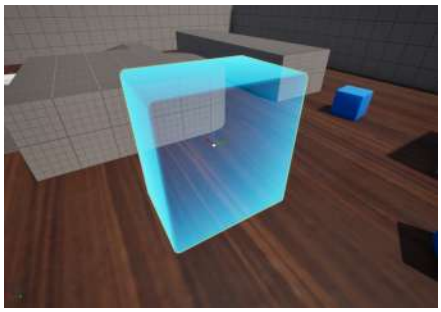


2.7 On souhaite faire en sorte que le lanceur disparaisse ensuite après le passage de l'acteur (utiliser la fonction `destroy Actor`).

2.8 Dupliquez votre objet et testez si les lanceurs fonctionnent toujours ?

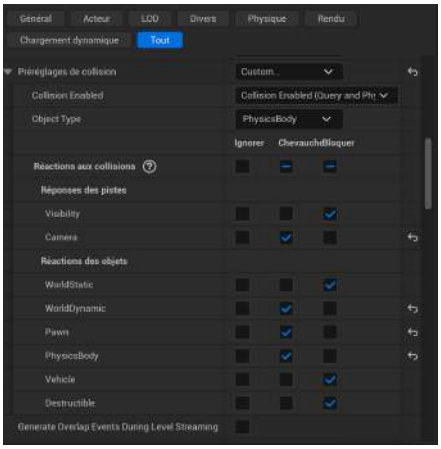
Maintenant que nous avons pu mettre en place une zone permettant à l'acteur de sauter plus haut, nous allons procéder de même pour créer une zone de téléportation.

2.9 De la même façon que précédemment, rajoutez une zone de téléportation qui sera symbolisée par un cube déformé que le joueur pourra traverser (image (a) ci-dessous). Pour cela vous pourrez utiliser la même texture que l'image (b) et changer les réglage de *chevauchement* (comme sur l'image (c)).



(a)

(b)



(c)

2.10 De façon à rendre la téléportation plus spectaculaire, dupliquez les éléments de la scène et déplacez l'un à coté comme sur l'image suivante :

2.11 Dans le nouveau monde dupliqué, rajoutez un objet de votre choix symbolisant la zone d'arrivée de la téléportation.

2.12 En utilisant un glissé/déposer, ajoutez l'objet précédant dans l'éditeur de niveau ***Level Blueprint*** (comme sur l'image ci-dessous).

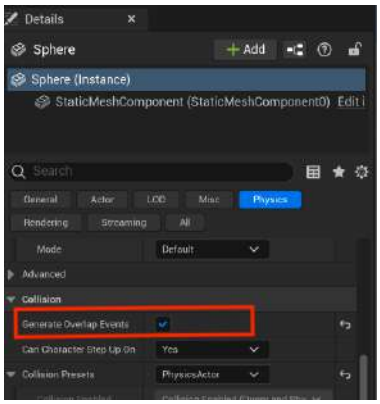
2.13 En ajoutant les noeuds suivants : `SetActorLocation` et `GetActorLocation`, finalisez la téléportation.

2.14 Testez le bon fonctionnement de la téléportation.

On souhaite désormais tester si la téléportation fonctionne aussi avec un autre objet porté sur la zone cible.

2.15 Rajoutez un objet facile à déplacer dans la scène (balle par exemple) et activez la simulation physique ainsi que la gestion des collisions pour pouvoir déplacer l'objet.

2.16 Tester de faire téléporter la balle en même temps que le joueur.



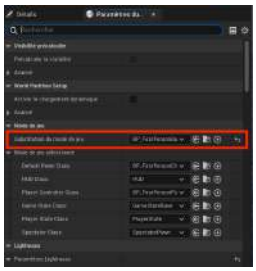
Partie III: Mode Fracture

Depuis la version 5 d'*Unreal Engine*, il est désormais possible de fracturer des éléments de la scène sans avoir à activer de plug-in particuliers. Dans cette partie, nous allons explorer ces possibilités avec des exemples de fracturation d'objets.

Les étapes à suivre pour fracturer un objet sont les suivantes :

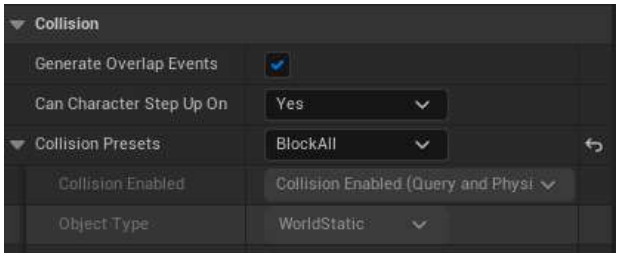
— Sélectionner un objet que vous souhaitez fracturer (élément de la scène : mur, cube, bloc)
— Entrez dans le mode *Fracture*. puis générez une nouvelle géométrie.
— Sélectionnez le type de fracture souhaitée.
— Désactivez la coloration des morceaux de l'objet.

3.1 Pour la suite, il sera plus simple de tester la destruction d'objets en étant en mode de jeu à la première personne. Pour cela, ajoutez dans votre projet le contenu associé au mode de jeu en *First Person*. Rajoutez un nouveau joueur *First Person* et définir le mode de jeu associé (voir image ci-dessous).



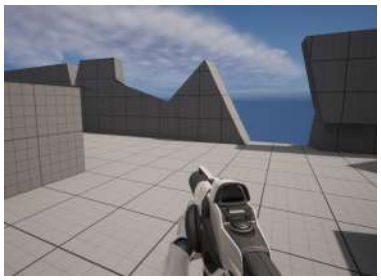
3.2 Il sera nécessaire aussi de rajouter une arme. Pour que les projectiles provoquent des dommages aux objets destructibles, éditez les propriétés de collision des deux composants suivants :

(a) collision box component



(b) collision de la sphere

3.3 Appliquez ces étapes pour que le joueur puisse détruire potentiellement les murs du plateau comme sur l'image ci-dessous.



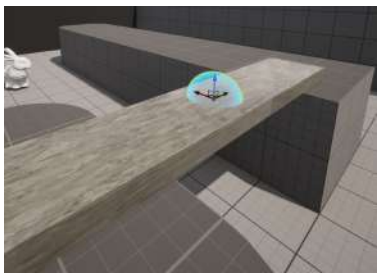
3.4 Sur un objet que vous importerez (comme le maillage du lapin (a) disponible sur moodle en format `.obj`), appliquez une destruction qui produira trois niveaux de cubes comme sur les images suivantes.



Dans certaines situations, il peut être aussi intéressant d'avoir un outil capable d'appliquer des dégradations sans actions spécifiques de l'utilisateur. Un outil possible est l'objet `FS_MasterField` (visible dans les assets affichant bien les éléments du moteur de jeu (option de visualisation d'assets *Show Engine Content*)). Ce dernier (illustré ci-dessous) permet de déclencher une dégradation sur les objets destructibles autour de lui (soit après un temps d'attente ou après un trigger).

3.5 Afin de tester cette possibilité, rajoutez un objet pour avoir une planche en marbre qui reliera les deux blocs comme sur l'image suivante :

3.6 Tester de rajouter l'objet ajouter l'objet `FS_MasterField` sur la planche et configurez la de façon à ce qu'elle se fracture toute seule au bout de 10 secondes.



3.7 Modifiez votre code pour faire en sorte que la plance se fracture lorsque le joueur passe dessus.

Partie IV: Pour les plus rapides...

4.1 Dans la continuité des éléments dynamiques, on souhaite réaliser un piège où le joueur se retrouverai enfermé (une grille qui lui tomberai dessus). Pour cela, soit vous pouvez récupérer la grille disponible sur moodle, ou la créer sur à partir d'un logiciel de modélisation tel que *Blender*.

4.2 Faire en sorte que le joueur puisse finir par briser la grille en tirant ou en bougeant rapidement.
