

Les cahiers du développeur Unreal Engine

Grégory Gosselin De Bénicourt



Modélisation, Blueprints, Matériaux & Paysages



Versions 4.7+

Les cahiers du développeur Unreal Engine



MATERIAL

Editions
Graziel

Tome 1

LES CAHIERS D'UNREAL ENGINE



TOME 1

Modélisation, Blueprints, Matériaux et Paysages

«If You Love Something, Set It Free»
(Tim Sweeney)

Copyright Éditions Graziel – Dépôt Légal: Avril 2015

ISBN: 979-10-93846-02-6

Éditions Graziel

9, chemin des Barrouters
81300 Graulhet
Tel: 09 52 05 40 15
Fax: 09 57 05 40 15
RCS: Castres B 801 370 800

courriel: infos@graziel.com

site web: www.graziel.com



Auteur: Grégory GOSSELLIN DE BENICOURT

Collection «Les cahiers du développeur »

Le logo d'Unreal Engine est une marque déposée par EPIC Games, 620 Crossroads Blvd., Cary, NC 27518 - USA

L'utilisation du logo et de la marque sur la couverture de ce livre a été validée par Amber Sharif, Marketing Manager, Unreal Engine, Epic Games, www.unrealengine.com

Toutes les autres marques citées ont été déposées par leur éditeur respectif.

La loi du 11 mars 1957 n'autorisant aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les «copies ou reproductions strictement réservées à l'usage privé du copiste et non destinés à une utilisation collective», et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, «toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants-droit ou ayant cause, est illicite» (alinéa 1^{er} de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contre-façon sanctionnée par les articles 425 et suivants du Code Pénal.

Présentation de la collection

Unreal Engine est probablement l'un des moteurs de jeu les plus aboutis de notre époque. Il est accompagné d'un **éditeur à la fois simple et puissant**, qui le rend accessible à un public non-développeur: artistes en tous genres, architectes et ingénieurs, étudiants. A l'origine, il était orienté vers les **grosses productions de jeux vidéo**. Mais il s'est doté dernièrement d'un **nouveau système de licence** très attractif qui le rend accessible à tous, pour des projets qui dépassent de loin le secteur du jeu: présentations interactives temps-réel, cinématiques, ... là où il y a une histoire à raconter, un produit ou un concept à présenter, voir un concert «Next Gen» dans un univers virtuel en utilisant un casque de réalité virtuelle !

Si on se base sur le nombre croissant de ses «fans», il deviendra probablement, et assez rapidement, un des outils les plus utilisés par les professionnels du secteur du jeu vidéo, ce qui en fait un candidat de choix pour tous ceux qui souhaitent s'initier au domaine ou intégrer prochainement le secteur des professionnels du jeu vidéo.

Unreal Engine possède une **documentation abondante**, voir impressionnante. De très nombreux sujets y sont traités, mais cette pléthore d'informations rend aussi la découverte de l'outil très difficile, tant **la couverture fonctionnelle** du produit **semble interminable**. Et quand on se penche sur un thème précis, on se retrouve inévitablement en manque d'informations, ce qui peut paraître paradoxal. C'est pourquoi plusieurs tutoriels vidéo ont été réalisés et l'apprentissage de l'outil passe inévitablement par l'étude de ces derniers. Il est aussi nécessaire d'étudier les projets fournis en exemple. Toutefois, il faut déjà un certain niveau de connaissances pour pouvoir profiter de ces exemples.

«Autant ouvrir un livre écrit en latin en espérant pouvoir acquérir les bases du langage par la pratique !»

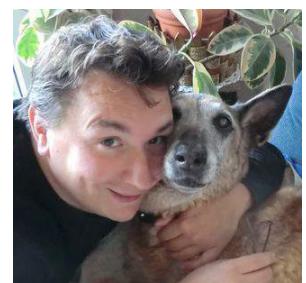
Ce que nous vous proposons est une **approche parallèle à cela**: un ensemble de plusieurs cahiers qui vous guideront dans la création de tel ou tel type de jeu, sans avoir la prétention de se substituer au manuel officiel. Au delà de la création des jeux, nous explorerons en détail tel ou tel aspect de l'outil, en faisant un point complet sur la question, comme «la création de cinématiques» (les «cute scenes») ou «la gestion des effets spéciaux». Chaque cahier peut être lu individuellement, et selon vos besoins. Mais si vous débutez totalement, nous vous conseillons fortement de démarrer votre apprentissage avec le premier tome de cette collection.

Vous nous excuserez ce ton parfois «dirigiste» qui s'apparente à celui d'une procédure, mais c'est l'approche qui nous semble la plus simple et la plus directe. Si vous avez besoin de précisions, il vous suffira de vous rapporter à la documentation officielle. Voyez plutôt ces cahiers comme une formation accompagnée de ses notes pour que vous puissiez rapidement retrouver l'information plus tard quand vous en aurez besoin.

L'auteur

Développeur de formation, l'auteur est tombé très jeune dans la création de jeux vidéo en réalisant ses premiers projets sur un Amstrad CPC. Puis, il a réalisé deux jeux qui ont été distribués en France, en Allemagne ainsi qu'aux USA.

Son premier livre «Créez vos propres jeux 3D comme les pros» est dédié à Blender et à son Game Engine: «J'ai écrit ce livre pour enseigner toutes **les techniques sous-jacentes à la conception d'un jeu 3D moderne**, ainsi que les étapes nécessaires à la création d'un jeu, seul ou en petite équipe. Les moteurs de jeu évolués masquent une bonne partie de l'implémentation de ces mécanismes. Je reste persuadé qu'on ne peut pas utiliser correctement ces outils sans avoir une connaissance profonde de la façon dont tout cela fonctionne au sein du moteur.»



Aujourd'hui, il est développeur freelance dans le secteur du jeu vidéo et travaille parallèlement sur un gros projet personnel avec une petite équipe: «The Lost Colonies» qui utilise principalement Unreal Engine.

«Lorsqu'on m'a demandé de travailler sur un manuel pour Unreal Engine 4, j'ai commencé par établir la couverture fonctionnelle de l'éditeur et du moteur – puis, toute la galaxie des plugins et outils tiers. J'ai alors réalisé que si je souhaitais écrire un livre complet sur le sujet, ce dernier prendrait probablement **plusieurs milliers de pages** et qu'il me faudrait faire de nombreux projets exemples. Tout cela nécessite également des recherches poussées dans le code source pour s'approprier plusieurs fonctionnalités non-documentées. Autant dire qu'il s'agit d'un **travail de titan**. Et puis, je me suis dit que ce serait certainement un livre imbuvable... il n'est pas nécessaire de maîtriser tous les aspects de l'outil pour l'utiliser convenablement dans ses propres projets. Ainsi, m'est venue l'idée de découper cette couverture par fonctions et par thèmes. Cela promet une grande collection de livres à terme, mais **cela permet aussi au lecteur de se focaliser sur ce dont il a besoin**: les jeux 2D, les jeux 3D, le level design, l'animation de personnages, le C++, la programmation visuelle, les courses de voitures ou les jeux de stratégie temps-réel, les cinématiques ou la programmation avancée des matériaux, le multijoueurs ... autant de domaines que j'aurais un grand plaisir à explorer avec vous.»

Nous espérons que ces cahiers répondront à vos attentes et sauront s'adapter à vos besoins au fil du temps. Nous comptons donc sur vos retours, vos suggestions, vos demandes pour améliorer cette collection.

Présentation du Tome 1 de la collection

Pour commencer cette série sur Unreal Engine, il nous fallait inévitablement commencer par les bases. Dans ce premier tome, nous allons vous aider à faire vos **premiers pas sous l'éditeur**.

Mais plutôt que de vous lister une à une chaque fonction, nous allons vous les faire découvrir par la pratique, en créant **votre premier jeu** sous UE4. Ce jeu sera complété par d'autres tomes au fil de vos lectures.

Tout d'abord, nous allons apprendre à utiliser les fonctions de **modélisation** de l'éditeur qui nous permettent de créer des objets à partir de formes de base (les «brushes») et leur appliquer des matériaux. C'est l'un des points forts de cet éditeur: il possède son propre outil de modélisation. Certes, il n'a pas la puissance d'outils comme Blender, Maya ou 3DSMax, mais sa simplicité et son intégration permettent de concevoir très rapidement la structure générale d'un niveau de jeu. Ce sera l'occasion de créer un premier bâtiment et son escalier, à aménager l'intérieur de ce bâtiment avec des objets externes. Ces objets peuvent être téléchargés, importés à partir d'autres projets ou modélisés par vos soins avec vos outils préférés.

Puis, nous allons créer nos premiers **Blueprints**: sortes de composants alliant objets 3D et programmation visuelle. Grâce aux Blueprints, on peut associer un «comportement», une fonctionnalité à un objet. Ainsi, nous allons créer plusieurs objets que nous placerons dans le bâtiment et qui auront un comportement particulier: boutons d'allumage de lumières, spots avec détecteurs de présence, système d'ouverture de portes, etc. Nous en profiterons pour vous montrer un petit aperçu de «**matinee**», l'éditeur de cinématiques qui permet, entre autres, de créer des séquences complexes d'animation.

Ensuite, nous explorerons un domaine assez important: la **création de matériaux**. En effet, Unreal Engine possède un système très abouti de programmation des matériaux. Si vous êtes graphiste et que vous modélez vous-même vos propres objets, il faudra en passer par là. Le Rendu Physique Réaliste ou PBR (Physically Based Rendering) est un modèle d'illumination pour créer des rendus 3D qui respectent les lois de la physique. Le rendu est époustouflant et se rapproche d'avantage de la synthèse d'images. C'est un domaine très vaste, mais au travers de la création de plusieurs matériaux, nous allons découvrir les principales fonctionnalités associées à ce domaine: les textures, les normal maps, la transparence et les masques, les matériaux émissifs, les matériaux animés et dynamiques, les déformations via le World Displacement, les Subsurface colors, la réfraction, etc. Nous créerons également nos propres fonctions de matériaux, ainsi que nos instances à partir de modèles.

Un chapitre très important de ce premier tome est la **modélisation de paysages** en utilisant les fonctions internes de l'éditeur. Pour notre jeu, nous allons donc modéliser un vaste paysage et y intégrer notre bâtiment. Nous explorerons toutes les fonctions de modelage du paysage (le sculpt): les différentes outils,

brosses, comment créer des chemins en utilisant les outils de plates-formes et rampes d'accès. Nous verrons également comment «copier-coller» des parties du paysage. Puis, nous aborderons la «peinture» du paysage en utilisant plusieurs calques (layers). Grâce aux connaissances que nous avons pu acquérir sur les matériaux, nous allons créer un matériau très particulier ayant la capacité de s'adapter à l'inclinaison du terrain. Ainsi, des rochers pourront surgir des dunes de sable, juste en élevant le niveau du terrain avec l'outil de sculpting. En créant ce genre de matériaux, la modélisation d'un paysage devient un délice et permet de décupler sa créativité.

Enfin, nous apprendrons à utiliser un autre outil fantastique qui fait la richesse de l'éditeur: l'**outil «foliage»**. On l'utilise pour ajouter au paysage des éléments comme la végétation, les roches, ou tout objet que l'on souhaite avoir en grande quantité mais avec lesquels on n'entre pas directement en interaction dans le jeu (autrement que par la collision par exemple). Cet outil extraordinaire permet de peindre un paysage avec des arbres comme on le ferait avec un pinceau et une toile. Il est hautement paramétrable et permet des optimisations spectaculaires, permettant ainsi de gérer des centaines milliers d'instances dans votre scène, tout en restant à un niveau acceptable de jouabilité. Il n'y qu'à voir cette démonstration «**Open Worlds**» présentée lors de la Game Developper Conference de 2015. Nous aurions pu assister à un film en images de synthèse où un cerf volant parcourt un vaste paysage... mais non, il s'agissait bien d'une animation temps-réel. Nous verrons comment créer de l'herbe ou importer des arbres et des rochers, et comment les associer au paysage.

Pour terminer, nous intégrerons un **personnage** au paysage – avatar que nous allons contrôler. Nous n'entrerons pas dans les détails de l'animation du personnage ou dans la création et l'importation de modèles existants, car ce sera l'objet du second tome, mais nous verrons comment gérer les entrées clavier et souris par exemple, et comment profiter des «Templates» disponibles en nous les appropriant.

Pour la lecture

Pour les opérations à la souris:

- «Cliquez» signifie tout simplement «appuyez avec le bouton gauche de la souris sur la zone désignée»
-  signifie «appuyez sur le bouton droit de la souris sur la zone désignée»

Nous allons utiliser **quelques abréviations** pour éviter les redondances:

- **CB**: fenêtre du «**Content Browser**»
- **WO**: fenêtre du «**World Outliner**»
- **DT**: fenêtre «**Details**»
- **MD**: fenêtre «**Modes**»

Lorsque nous utilisons le signe «+» entre deux touches (ex: ), cela signifie qu'il faut enfoncez les 2 touches **simultanément**.

Le plus souvent, dès que nous vous indiquerons une adresse pour un site web, nous vous fournirons le QR CODE correspondant. Ces derniers peuvent être lus par des applications disponibles sur les mobiles : il suffit de pointer son téléphone au dessus du QR Code pour que ce dernier le prenne en photo, l'interprète et lance le navigateur web en suivant le lien encodé dans le QR Code.



Certains Blueprints ne sont pas toujours faciles à lire. Nous vous proposons de **télé-charger les copies d'écran** que nous avons insérées dans ce livre à cette adresse: <http://bit.ly/1zsM54X>

Si vous avez des questions relatives au projet développé dans ce tome, nous vous invitons à vous rendre sur le forum dédié à l'adresse: <http://forum.benicourt.com>

Sommaire

PRISE EN MAIN.....	8
Qu'est-ce que l'Unreal Engine ?.....	9
Téléchargement et Installation.....	10
L'interface de lancement.....	11
L'éditeur de niveau.....	13
Découverte de l'interface.....	14
Effectuons quelques manipulations d'objets.....	18
La fenêtre de rendu.....	19
Différents types d'affichage.....	21
Configuration des fenêtres de l'éditeur.....	22
Le système d'aide intégré.....	23
Création d'un niveau.....	25
Modélisation d'un premier bâtiment.....	27
Aménageons un peu la scène.....	33
Création de composant programmable.....	34
PROGRAMMATION BLUEPRINT.....	38
Découverte.....	39
Débogage du jeu.....	41
Introduction au «mode debug».....	41
Les breakpoints.....	43
La fenêtre de débogage.....	44
A l'ancienne.....	44
Création d'un bouton d'allumage de lampes.....	45
Première animation avec Matinee.....	47
Programmation Blueprint de l'animation.....	50
Fermeture automatique de la porte.....	52
Les bases de la programmation Blueprint.....	53
Qu'est-ce qu'un Blueprint?.....	53
Types de données et variables.....	54
Tableaux et boucles.....	56
Instructions de branchement (flow control).....	57
L'ÉDITEUR DE MATERIAUX.....	59
Création d'un nouveau matériau.....	60
Découverte de l'interface.....	62
Un matériau Métallique.....	64
Les matériaux texturés.....	66
Un matériau de type «Miroir».....	68
Peindre un objet avec le vertex painting.....	70
L'utilisation des masques.....	71
Vertex painting et masques.....	74

Gestion de la transparence.....	75
Première approche: la technique des masques.....	75
Seconde approche: la transparence alpha.....	76
Matériaux Émissif.....	77
Matériaux animés.....	78
Matériaux dynamiques.....	78
Quelques mots sur les autres matériaux.....	79
Fonction de matériau.....	82
Instance de matériau.....	84
 MISE EN PLACE DU PAYSAGE.....	 86
Création d'un Landscape.....	87
Modeler le paysage.....	90
Les différents outils et brosses.....	90
Plates-formes et rampes d'accès.....	93
Tracer une route.....	94
Peindre le paysage.....	96
Gestion des layers.....	96
Un matériau qui s'adapte à l'inclinaison du terrain.....	98
Ajout d'un plan d'eau.....	101
Mise en place de la végétation et du décor.....	102
Création d'un matériau «herbe».....	102
Utilisation de l'outil foliage.....	105
Les assets de la démo «Open World».....	111
Le Foliage Starter Kit.....	112
Cycle jour/Nuit.....	113
Ajout d'un effet de brouillard.....	115
Brouillard atmosphérique.....	115
Brouillard type Exponential Height.....	116
Mise en place du joueur.....	117
Importation du template «Third Person».....	117
Contrôler le personnage.....	118
 ANNEXES.....	 120

PRISE EN MAIN



QU'EST-CE QUE L'UNREAL ENGINE ?

Unreal Engine est composé d'un éditeur offrant tous les outils nécessaires à la création d'un jeu vidéo moderne, ainsi qu'un moteur de jeu redistribuable permettant de faire tourner ce jeu sur de nombreuses plates-formes: ordinateurs de bureau (Windows, Linux, OSX), mobiles (Android, iOS), navigateurs web (WebGL/HTML5), ainsi que consoles (Microsoft XBOX, Sony Playstation) et même Oculus Rift.

L'Unreal Engine a été développé par Epic Games. A l'origine, il est le concurrent principal de moteurs tels que l'IdTech (Doom), le CryEngine (Far Cry) et le Source Engine de Valve (HalfLife). Depuis, nous pourrions ajouter Unity qui est devenu un outil robuste de création de jeux vidéo et qui se rapproche de plus en plus de Unreal Engine en terme de fonctionnalités, surtout avec la version 5.

L'Unreal Development Kit (UDK) est une version gratuite d'Unreal Engine 3. Si vous le connaissez, c'est très bien, mais l'interface a considérablement changé (notamment Kismet). Même le langage de script, l'UnrealScript a été abandonné au profit du C++ et de Blueprint.

Avant de commencer, assurez-vous que vous disposez d'un ordinateur suffisant pour faire fonctionner convenablement l'outil. Les préconisations affichées sur le site officiel sont:

- Windows 7 64-bit ou plus (Windows 8+) ou Mac OS X 10.9.2 ou plus
- Une carte graphique supportant DirectX 11 ou plus
- 8 Go de mémoire vive (RAM), même si cela peut fonctionner avec moins

S'il est possible aujourd'hui de compiler les sources de l'éditeur sous Linux, il est toutefois conseillé de ne pas l'envisager pour une mise en exploitation. Pour plus d'informations, consultez la page suivante:

https://wiki.unrealengine.com/Linux_Support

Toutefois, sous Windows et OSX, il est possible de packager ses jeux pour Linux.



Illustration 1: Éditeur Unreal Engine (version 4.7.3 sous Windows)

TÉLÉCHARGEMENT ET INSTALLATION

Pour télécharger la dernière version d'UE4, il vous suffit de vous inscrire sur le site d'Unreal Engine (voir plus loin). Cet enregistrement donne accès à la dernière version de l'éditeur, à son code source, toute la documentation, les tutoriaux et exemples, ainsi qu'au forum d'entraide.

Pour commercialiser des jeux réalisés avec UE4, vous devrez vous acquitter des royalties s'élevant à 5% des bénéfices (sur le montant dépassant 3000\$ par trimestre). Par exemple, si vous vendez une application sur Google Play et qu'elle vous rapporte 4500\$ au second trimestre 2015 (hors frais de Google Play), alors vous devrez vous acquitter de 5% de (4500-3000)\$, soit 75\$. C'est un partenariat gagnant-gagnant! À noter que les projets de film ou d'architecture n'auront pas besoin de reverser ces royalties.

Pour voir les modalités de licences, aller à cette adresse:
<https://www.unrealengine.com/eula>



Epic a annoncé la gratuité de son éditeur lors de la Game Developer Conference qui s'est déroulée du 2 au 6 mars 2015 à San Francisco. Avant, les développeurs devaient souscrire un abonnement mensuel de 19\$/mois, cela a duré moins d'un an. Auparavant, il était payant et seules les grosses sociétés pouvaient se le permettre. C'était déjà une première approche réussie vers la démocratisation de l'outil. Depuis l'annonce par Epic, plusieurs autres éditeurs ont annoncé le passage à la gratuité également: Unity 5 qui n'est «quasiment» pas bridé (à part le splashscreen) dans la «personnal edition» et Source 2 de Valve. Une bataille fait rage chez les éditeurs et les gagnants sont les studios de création de jeux vidéo qui vont pouvoir proposer des jeux de qualité AAA sans devoir investir de grosses sommes dans l'outil de développement.

Tim Sweeney, le fondateur d'Epic, explique ce passage au gratuit «par une volonté de libérer la créativité et de permettre au jeu vidéo, en tant qu'art, de continuer à se développer». Il résume cette opération par «If you love something, set it free» (littéralement, «si vous aimez quelque chose, rendez-le libre»).

Pour télécharger l'éditeur:

1. Aller à l'adresse suivante: <https://www.unrealengine.com/register> et remplir les différents champs. Le mot de passe doit contenir au moins 7 caractères, avec au moins un chiffre et au moins une lettre, sans espace. Cliquer sur «Sign Up» pour créer le compte.
2. Alors, le «Dashboard» donne accès au téléchargement de l'éditeur (encadré rouge). Selon son équipement, cliquer sur «Download» (Windows ou Mac).



The screenshot shows the Unreal Engine dashboard interface. At the top, there's a navigation bar with links for ABOUT, LEARN, COMMUNITY, MARKETPLACE, ACADEMIA, ACCOUNT, LOG OUT, and a search icon. The user account information is displayed: Grégory Gosselin De Bénicourt [edit profile], Account Balance €30.00, and the handle benicourt. Below the navigation, there are tabs for Download, Profile, Billing, Transactions, Redeem Code, and Password. The Download tab is active. Under the Download tab, there are three main sections: 1. Get Unreal Engine: A large orange 'Download' button is highlighted with a red border. Below it, text says 'or choose your platform: Windows | Mac'. 2. Get Unreal Tournament: An orange 'Download' button with 'Windows | Mac' below it. 3. Get UE4 Full Source Code: Text explaining how to link a GitHub account to download the source code. At the bottom right of the dashboard, there's a 'NEED HELP?' link with a question mark icon.

Illustration 2: "Dashboard" sur le site d'Unreal Engine

3. Une fois le fichier d'installation téléchargé, le lancer.
4. Dans un premier temps, choisir le répertoire de l'installation (comptez 8 Go minimum). Ensuite, entrer les éléments du compte (email et mot de passe choisi préalablement).
5. Enfin, le téléchargement de l'éditeur pourra commencer.

Il s'agira de la dernière version officielle. Cependant, il est possible un peu plus tard de télécharger d'autres versions du moteur: des versions antérieures pour assurer une parfaite compatibilité avec des projets existants, et des versions en «avant-première» à tester selon ses envies.

L'INTERFACE DE LANCEMENT

Voilà, l'éditeur est installé et si vous souhaitez tout de suite le lancer pour apprendre à l'utiliser, vous pouvez sauter la section suivante. Nous préférions dans un premier temps vous présenter la petite interface qui permet de lancer l'éditeur, et bien d'autres choses. Voyez la comme un compagnon bien pratique qui vous permet de rester connecté à la communauté d'Unreal.

Lorsque vous ouvrez l'interface de démarrage (illustration 3), vous arrivez tout de suite sur la section «Learn» (Apprentissage). Vous avez ainsi un raccourci rapide pour consulter la documentation, regarder les vidéos de tutoriels et participer au wiki. Le Wiki est un site web collaboratif où vous pouvez trouver de nombreuses informations, mais également participer au contenu (chaque page peut-être éditée et vous pouvez ajouter ou corriger des éléments sous le contrôle de la communauté, un peu comme sous Wikipédia).

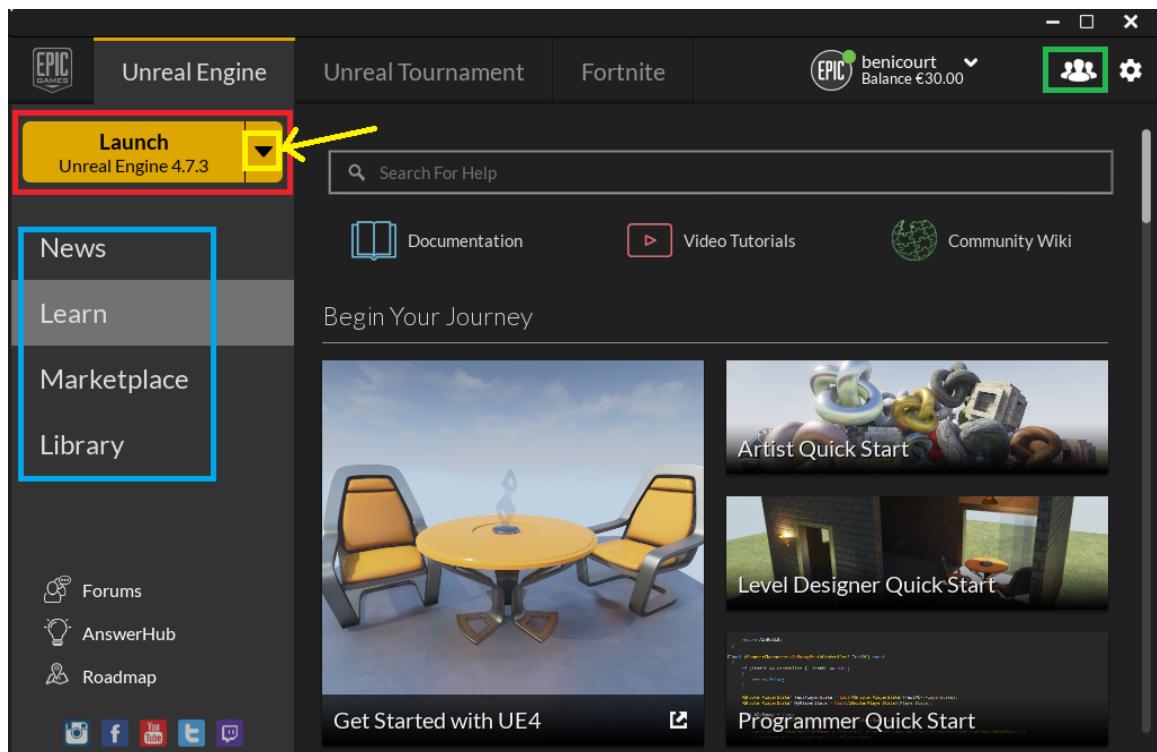


Illustration 3: Interface de lancement d'Unreal Engine

Les «Samples» constituent des exemples que vous pouvez télécharger et qui sont relativement bien documentés, le principal étant «Content Examples» qui contient une visite guidée des principales fonctionnalités du logiciel, classé sous forme de «levels» (niveaux). Nous reviendrons sur cela un peu plus loin. Mais vous pouvez d'ores et déjà l'installer. L'exemple «Matinee» est probablement l'un des plus spectaculaires: vous assistez à un combat filmé comme si tout cela provenait d'une séquence de film – sauf qu'il s'agit bien ici d'une démo exécutée en «temps-réel».

Si vous faites défiler la fenêtre, vous arriverez sur des exemples de jeu complets.

Tous ces éléments ne sont pas encore sur votre ordinateur. L'interface vous propose de télécharger chaque élément. Une fois téléchargé, il arrive dans votre librairie («**Library**») et vous pourrez, soit créer un projet à partir de cet exemple, soit, s'il s'agit d'un asset, l'ajouter à l'un de vos projets existants. La «library» vous permet donc de consulter tout ce qui a été téléchargé avec cette interface, ainsi que la liste des projets qui ont été créés. Elle permet également d'ajouter ou de retirer des versions de l'éditeur (voir illustration 4).

Notez qu'il est possible de télécharger plusieurs choses en même temps: les éléments viendront grossir les rangs de la file de téléchargements.

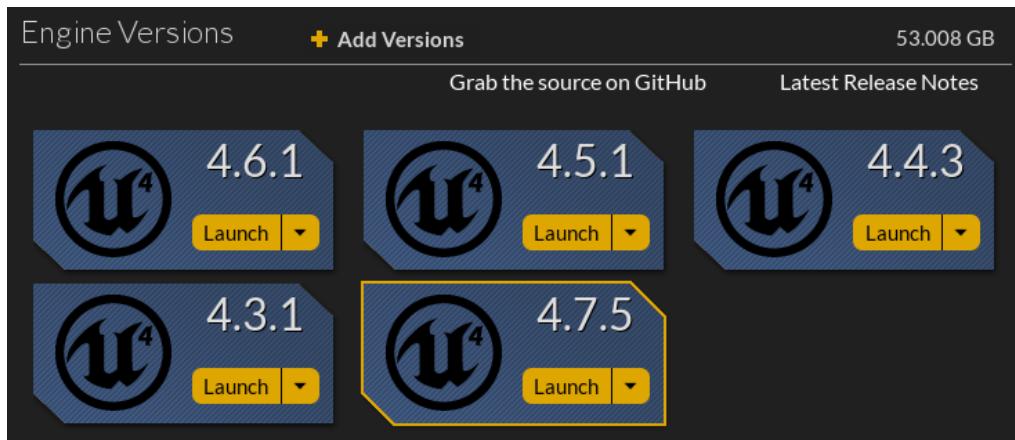


Illustration 4: Ajout et suppression de versions de l'éditeur

Enfin, la section «**Marketplace**»: S'il est possible de trouver plusieurs assets gratuitement, il est aussi possible d'en acheter... et d'en vendre! Remarquez le lien en haut à droite de l'écran «submit content». Il s'agit en fait d'un lien vers le site vous permettant de proposer votre propre asset dans le «marketplace». Pour plus d'informations à ce sujet, vous pouvez consulter la page suivante:
<https://www.unrealengine.com/marketplace/marketplace-submission-guidelines>



Remarquez également le petit outil de discussion (encadré vert de l'illustration 3) qui vous permet de vous mettre en relation avec d'autres utilisateurs sous la forme d'un chat.



Un **asset** est un pack contenant, au plus simple, juste une musique, un modèle 3D ou une texture et, plus généralement, un ensemble d'éléments: modèles, animations, programmes. Vous pouvez par exemple ajouter à votre projet un personnage qui sera accompagné de plusieurs animations, d'une intelligence artificielle (capable de suivre un ennemi et de tirer), de sons (tirs, bruits de pas, etc.), voire d'effets spéciaux. Ce sera un asset «personnage». C'est donc une notion assez large qui correspond à une compilation plus ou moins importante de ressources qui peuvent être utilisées tels quels dans notre projet, juste en l'important.

L'ÉDITEUR DE NIVEAU

A partir de l'interface de démarrage, il est aussi possible de lancer l'éditeur avec le bouton «Launch» (encadré rouge de l'illustration 3). C'est la porte d'entrée de l'éditeur quand on souhaite créer un nouveau projet ou ouvrir un projet existant (on pourra également passer par «Library»). Si vous avez installé d'autres versions de l'éditeur, vous pourrez sélectionner cette version grâce au bouton de l'encadré jaune de la même illustration.

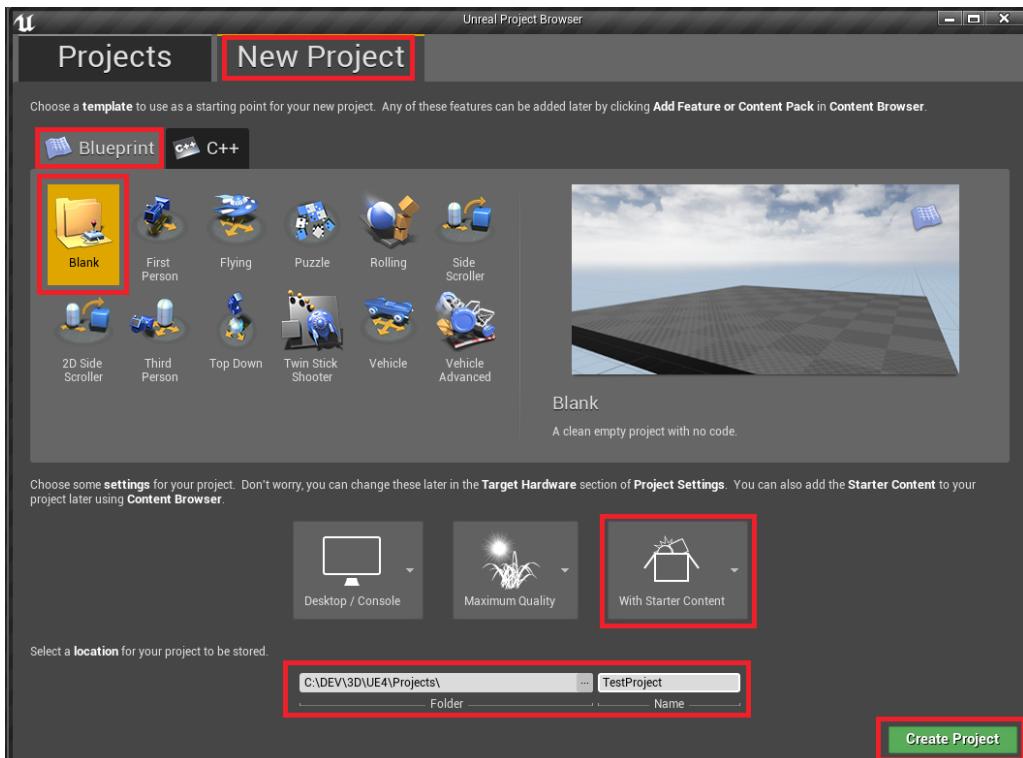


Illustration 5: Création d'un nouveau projet de type Blueprint / Blank

Création d'un nouveau projet:

1. Lancez la version de l'éditeur qui a été téléchargée lors de l'installation du logiciel en cliquant sur «Launch». Au bout d'un petit moment, l'interface principale de l'éditeur invite à ouvrir ou créer un nouveau projet.
2. Cliquez sur le volet d'onglet «New project» (illustration 5)

2 types de projets peuvent être créés:

- les projets de type «Blueprint»: il n'y a aucune ligne de programme, il s'agit de programmation utilisant des «nœuds» visuels dans un graphe. Pour les connaisseurs d'UDK, c'est une version grandement améliorée de Kismet.
- les projets de type «C++»: ces derniers nécessitent la présence d'un éditeur de code comme Visual Studio de Microsoft. Nous réaliserons un petit programme d'exemple dans la suite de ce livre pour vous faire découvrir cette approche.

Blueprint permettant d'accéder à la quasi-totalité des fonctions d'Unreal Engine, il n'est donc pas nécessaire d'utiliser le C++ (sauf cas relativement rares). Mais nous reviendrons sur ce sujet quand nous aborderons la programmation.

3. Sélectionnez le projet de type «Blank»

C'est le modèle de projet (template) qui contient la base minimale pour démarrer tout type de projet. Les autres modèles peuvent être utilisés pour créer des jeux de tir en vue subjective («First Person»), des jeux d'avion avec 6 degrés de liberté («Flying»), des jeux de plates-formes 2D («2D Side Scrolling») ou même une course de voitures («Vehicle»). L'avantage du template, c'est qu'il est tout de suite jouable: on dispose d'un modèle, de sa logique programmée, d'un petit environnement de test et d'une caméra paramétrée.

4. Sélectionner un répertoire pour ce premier projet de test qu'on appellera tout simplement «TestProject». Nous garderons l'option «With Starter Content» car elle permet d'ajouter un pack de démarrage contenant divers objets et effets spéciaux. C'est une bonne façon de s'initier à l'éditeur.
5. Enfin, cliquez sur le bouton «Create Project» pour lancer la création du projet.



Astuce: Le «StarterContent» peut facilement être ajouté à n'importe quel moment du projet. Dans la fenêtre «Content Browser», cliquez sur «Add New», puis sur «Add Feature or Content Pack». C'est l'une des nouveautés de la version 4.7, il nous est proposé deux types de pack: l'un standard, et l'autre optimisé pour les mobiles.

Découverte de l'interface

Nous arrivons alors sur l'interface principale, l'**éditeur de niveau** (illustration 7), avec quelques éléments qui ont déjà été ajoutés au niveau courant (une table, 2 chaises, un sol, une musique d'ambiance, des lumières, un ciel, etc.). Avant de vous expliquer à quoi correspondent les différentes fenêtres :

1. Cliquez sur le bouton «Play» (cerclage jaune).
2. Cliquez au centre de la scène, puis déplacez votre souris en maintenant le clic enfoncé. Avec les **flèches** de votre clavier, vous pourrez vous déplacer dans la scène.

A tout moment, vous pouvez récupérer votre souris en appuyant sur «**Shift + F1**». Ce système de déplacement n'est pas figé: il s'agit d'une configuration de touches qui est proposée par défaut mais qui peut-être modifiée. Pour sortir, pressez **Esc**.

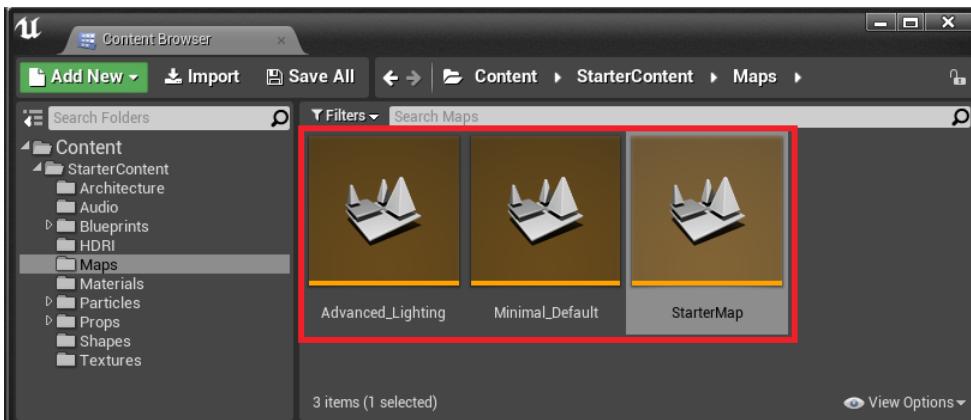


Illustration 6: Ouvrir un niveau du jeu à l'aide du menu

Vous venez d'expérimenter le déplacement en mode jeu. Maintenant, nous allons essayer la **navigation en mode édition**: tout d'abord :

3. cliquez dans la fenêtre de la zone 7 (illustration 7) et tout en maintenant le bouton enfoncé, déplacez votre souris.

Vous pouvez constater que si vous dirigez votre souris vers le haut, vous avancez, vers le bas, et vous reculez. Pour le mouvement droite/gauche, votre vue va s'orienter vers la droite ou la gauche par un mouvement de rotation. Si vous utilisez le bouton droit de la souris, c'est uniquement un mouvement de rotation que vous obtiendrez. La molette de la souris vous permettra d'avancer ou de reculer. Si vous maintenez le bouton central de la souris, vous pourrez bouger vers haut, le bas, à droite ou à gauche. Si vous ne disposez pas d'un bouton central, alors utilisez le bouton droit et gauche de la souris simultanément pour obtenir le même résultat. Notez bien les différences d'utilisation. Ensuite, on peut aussi utiliser les flèches: flèche du pour avancer/reculer, pour déplacement horizontal, et pour monter/descendre.

Le répertoire «Maps» de «StarterContent» donne accès à 3 levels. Celui qui a été chargé par défaut est «**Minimal_Default**». Si nous n'avions pas choisi d'ajouter le «Starter Content», nous aurions juste eu un sol et un ciel. Les deux autres niveaux sont «**Advanced_Lighting**» et «**StarterMap**». Si vous le souhaitez, vous pouvez les ouvrir pour les tester comme le précédent. Pour cela, vous pouvez passer par le menu («File» → «Open Level») et sélectionner un des niveaux contenus dans le répertoire «StarterContent» du jeu (illustration 6). Nous verrons un peu plus loin qu'il y a une autre façon de procéder.

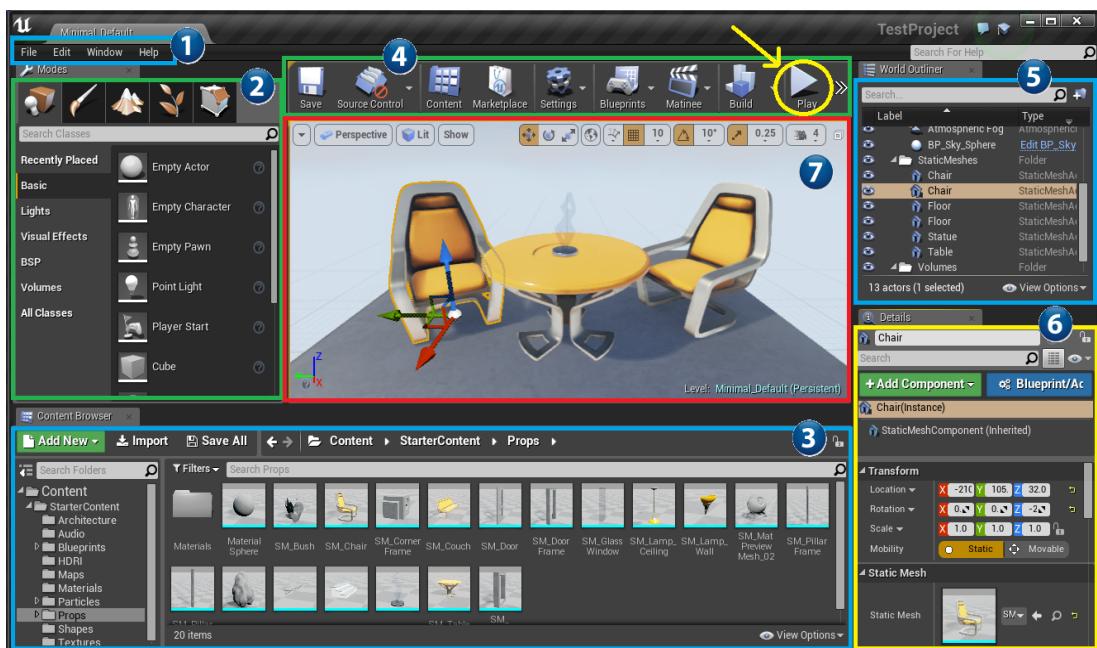


Illustration 7: Interface principale de l'éditeur

Reprendons maintenant l'interface principale de l'éditeur (illustration 7). On distingue 7 zones:

- **Zone 1:** C'est l'accès au menu principal – Le menu «File» permet d'accéder aux différents levels, mais également d'ouvrir d'autres projets. Il permet d'importer des assets ou de les exporter, ainsi que de packager le jeu pour telle ou telle plate-forme. Le menu «Edit», en dehors des traditionnels «couper-copier-coller», permet l'accès aux préférences de l'éditeur et du projet, mais ce n'est pas la seule façon d'y accéder. Le menu «Window» permet d'accéder à des fonctions plus particulières que nous verrons un peu plus loin. Enfin, le menu «Help» donne accès à la documentation en ligne, à la description des API, le forum, etc.
- **Zone 2:** La fenêtre «Modes» (**MD** par la suite) donne accès aux principaux outils de l'éditeur 3D. Juste en dessous de «Modes», 5 icônes se comportent comme des volets d'onglets:
 - **«Place»:** principales primitives (cube, sphère, plan), lumières, caméras et types d'objets qu'on peut trouver dans la scène. Ces objets se nomment «Actors» sous Unreal.

- «**Paint**» permet d'ajuster la couleur et la texture d'un objet de la scène. Ainsi, on peut peindre directement un objet comme si on disposait d'une bombe à taguer.
- «**Landscape**» permet de modeler un paysage, créer des collines et des crevasses.
- «**Foliage**» permet d'ajouter des objets qui ne seront pas considérés comme des actors. C'est le cas des feuilles d'un arbre, d'herbe ou de fleurs. Mais dans l'absolu, tout objet peut être ajouté de cette manière: des roches, des immeubles, etc.
- «**Geometry Editing** » permet de modifier la géométrie de certains objets comme les BSP.
- **Zone 3:** Le «**Content Browser**» (ou **CB**) est l'explorateur de contenu. Il représente l'ensemble des assets disponibles pour le projet, des Blueprints aux modèles 3D en passant par les animations, les effets spéciaux, les widgets, etc. Ce n'est ni plus ni moins qu'une vue organisée sous la forme d'un répertoire. Si on ouvre son explorateur de fichier et qu'on ouvre le répertoire contenant les fichiers du projet, sous «content», on retrouvera exactement les mêmes éléments (voir illustration 8). C'est à partir de cette fenêtre que nous créerons les différents Blueprint et que nous importerons les assets du projet. C'est en quelque sorte le réservoir d'objets que l'on peut ajouter dans un niveau – il ne correspond pas aux objets d'un niveau! Par exemple, vous pouvez y trouver un modèle de chaise, mais tant qu'il n'est pas placé dans un level, ce n'est pas un «actor». De même, à partir d'un modèle de chaise, vous pouvez ajouter dix «actors» chaises. Quand ils sont ajoutés au level, on parle d'«actors» et on y accède avec un clic gauche sur l'objet dans la fenêtre «perspective», ou en passant par le «World Outliner» de la zone 5.

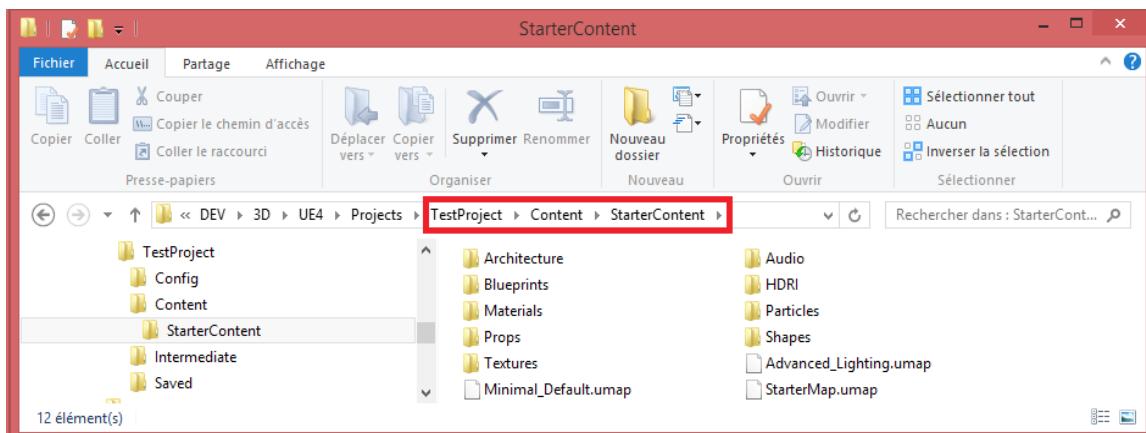


Illustration 8: Répertoire du projet vu sous l'explorateur de Windows 8

- **Zone 4: Barre d'outils** permettant d'accéder à divers raccourcis, mais aussi de construire le projet et de lancer le jeu, que cela soit sous l'éditeur, dans un programme séparé ou directement sur votre tabletPC:
 - «**Source Control**» permet d'utiliser un gestionnaire de version externe comme Github, subversion ou perforce. Pour travailler plus facilement à plusieurs sur un projet et/ou sauvegarder plusieurs versions de ce dernier.
 - «**Content**» permet de faire réapparaître la fenêtre **CB** si elle est fermée.
 - «**Marketplace**» donne directement accès à la boutique permettant d'acheter des assets.
 - «**Settings**» donne accès à divers éléments de configuration. Il y a un raccourci rapide vers les paramétrages du projet («Project Settings») et les paramétrages du jeu («World Settings»). Si vous trouvez, par exemple, que l'éditeur est un peu lent ou que la qualité n'est pas optimale, changer les paramètres dans «Engine Scalability Settings».
 - «**Blueprints**»: permet de créer ou d'ouvrir un Blueprint, mais on passera plus facilement par le **CB**. Toutefois, c'est très utile pour ouvrir le Blueprint lié au niveau chargé («Open Level Blueprint»).

- «**Matinee**»: éditeur d'animations permettant, entre autres, de réaliser de magnifiques cinématiques et des mouvements de caméra. Nous l'étudierons un peu plus tard.
 - «**Build**»: permet de «construire» le niveau en pré-calculant un certain nombre d'informations, comme les ombrages statiques ou les chemins pour l'Intelligence Artificielle. «Map Check» donne accès au log des messages générés durant la compilation de la scène.
 - «**Play**»: pour lancer le jeu de différentes façons.
 - «Selected viewport» permet de lancer le jeu dans la fenêtre active (zone 7),
 - «Mobile preview» émule le lancement du jeu sur un mobile (Android ou iOS),
 - «New Editor Window» lance le jeu dans une autre fenêtre
 - et «Standalone game» compile une version exécutable du projet puis la lance.
 - Le mode «Simulate» ne lance pas vraiment le jeu: on peut naviguer dans le jeu comme si on était en mode édition, mais il n'y a pas de «Player», donc pas d'interaction avec le joueur.
 - «**Launch**»: permet de lancer le jeu sur d'autres supports que la machine de développement (ex: une tablette ou un smartphone).
- **Zone 5:** «**World Outliner**» (**WO**) est le «scenegraph» du level actif, c'est à dire l'arborescence de tous les «Actors» du level. La hiérarchie des objets est préservée: si un actor «fils» est lié à un actor «père», alors il apparaîtra sous le père avec un certain décalage. On peut sélectionner un objet dans la fenêtre 3D avec un clic gauche sur son nom, ou cacher un objet avec un clic gauche sur l'icône «œil» par exemple.
 - **Zone 6:** «**Details**» (**DT**) permet de modifier les caractéristiques d'un actor préalablement sélectionné dans la fenêtre «perspective» ou dans **WO**. Il peut s'agir de sa position, de sa rotation, mais également de sa couleur s'il s'agit d'une lumière, de sa focale s'il s'agit d'une caméra, etc.
 - **Zone 7:** Il s'agit de la fenêtre «**Viewport**» qui effectue le rendu de la scène **en temps réel** (**RT**) pour passer en mode immersif). L'affichage est en 3D (mode «Perspective») mais peut basculer en 2D (en choisissant une vue de dessus par exemple) ou encore en mode 4 vues. C'est une vue interactive: plusieurs éléments animés en témoignent. Si vous ajoutez un effet «feu» par exemple, il sera animé dans cette fenêtre. Un petit menu intégré sous forme d'une barre d'outils (le «viewport UI») en haut de la vue permet d'accéder à diverses fonctions: choisir entre rotation, translation ou redimensionnement, le type d'affichage, ce qu'on veut y voir affiché (afficher ou non les volumes de collision par exemple), etc.



Astuce: Vous pouvez gagner un peu d'espace à l'écran en **masquant le nom des fenêtres**.

Pour cela, ⌂ sur le nom de la fenêtre et choisissez «hide». Le nom disparaît et un petit coin jaune apparaît. Il suffira de cliquer sur le coin jaune pour revenir à l'affichage standard avec le nom. Masquer le nom n'est possible que si les fenêtres ne sont pas affichées sous forme de volets d'onglet.

Effectuons quelques manipulations d'objets



Illustration 10: Outil de déplacement sélectionné

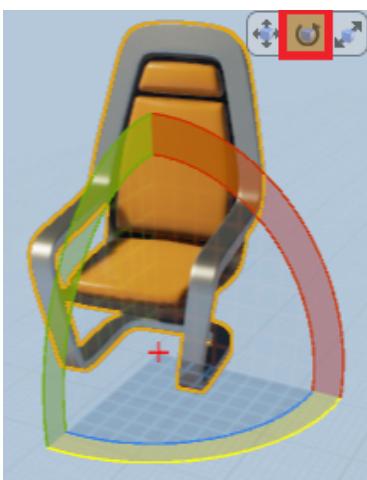


Illustration 9: Outil de rotation sélectionné



Illustration 11: Outil de rotation en cours d'utilisation

1. Maintenant que vous avez fait connaissance avec l'interface, allez dans **CB** (zone 3), et développez «Starter Content» si ce n'est déjà fait. Cliquez sur «Props», puis sur «SM_Chair». En réalisant un glissé-déposé (Cliquez sur «SM_Chair», gardez le bouton enfoncé, bougez la souris vers **le viewport** puis relâchez le bouton), et ajouter une chaise.

Par cette opération, vous venez de créer un nouvel «actor» dans le niveau, basé sur le modèle «SM_Chair» (chaise de bureau livrée avec le «Starter Pack»).

2. Lorsqu'on clique sur un objet, il passe en **surbrillance**. Si on presse **F**, la vue se focalise sur l'objet, c'est un raccourci utile.

Selon l'outil de manipulation utilisé (déplacement, rotation, redimensionnement), il devient possible d'appliquer directement des transformations sur l'objet. Sur l'illustration 10, l'outil de déplacement est sélectionné. Les 3 petites icônes en haut de l'illustration permettent de changer le type de transformation. Dans **le viewport**, ces icônes se situent dans la barre d'outil dont nous avons préalablement parlé. Pour déplacer l'objet, il suffit de placer le pointeur de la souris sur l'un des 3 axes (rouge pour l'axe X, vert pour l'axe Y et bleu pour l'axe Z) et de réaliser un clic gauche tout en maintenant le clic et en déplaçant la souris. Si on utilise la petite sphère blanche au croisement des axes, on réalise la même opération, mais dans l'espace.

3. Sélectionner l'outil «rotation» en cliquant sur l'icône correspondante (illustration 9, encadré rouge). Les axes de rotation sont symbolisés d'une façon très différentes. En cliquant sur la zone bleue (rotation d'axe Z), cette dernière devient jaune comme dans l'illustration. En maintenant le bouton de la souris enfoncé et en bougeant la souris (droite-gauche par exemple), on réalise la rotation comme dans l'illustration 11.
4. L'outil de redimensionnement fonctionne comme les 2 autres outils; testez-le également. Pour redimensionner l'objet, vous pouvez utiliser le petit cube blanc au centre des axes. Pour déformer l'objet selon un seul axe, il suffit de choisir l'axe tout simplement.

Des touches de raccourcis peuvent permettre de passer rapidement d'une transformation à une autre: **W** pour le déplacement, **E** pour la rotation et **R** pour le redimensionnement.

Toutes ces transformations peuvent être réalisées **en entrant directement la valeur souhaitée**: c'est quelques fois plus pratique quand on suit un plan métré par exemple. Pour cela :

5. Sélectionnez un objet et rendez-vous dans la fenêtre «détails» (zone 6, illustration 12).

«Location» correspond à la position de l'objet et «Scale» au dimensionnement (scale 1,1,1 : aucune transformation n'est appliquée, scale 0,5,0,5,0,5 : l'objet est 2 fois plus petit qu'à l'origine). Notez le petit «cadenas» qui, s'il est fermé, oblige à réaliser un redimensionnement proportionnel, c'est à dire identique sur tous les axes. Rien n'empêche de faire d'abord un scale 1,1,2 cadenas ouvert, de fermer le cadenas, puis de faire un scale de 2 sur X: automatiquement le redimensionnement passera à 2,2,4.



Illustration 12: Fenêtre "détails", panel "Transform"

Un autre touche à connaître est **[End]** («snap to floor»): elle déplace tout objet sélectionné vers le bas jusqu'à ce qu'il rencontre un autre objet. Plus simplement, elle permet de «coller» au niveau du sol n'importe quel objet, ce qui peut être très pratique.



Astuce: Nous vous invitons à parcourir les différents raccourcis claviers que vous trouverez dans les préférences de l'éditeur, rubrique «General», section «Keyboard Shortcuts». Vous y trouverez de nombreux raccourcis qui vous simplifieront la vie et vous permettront d'être beaucoup plus performants sous l'éditeur.

La fenêtre de rendu

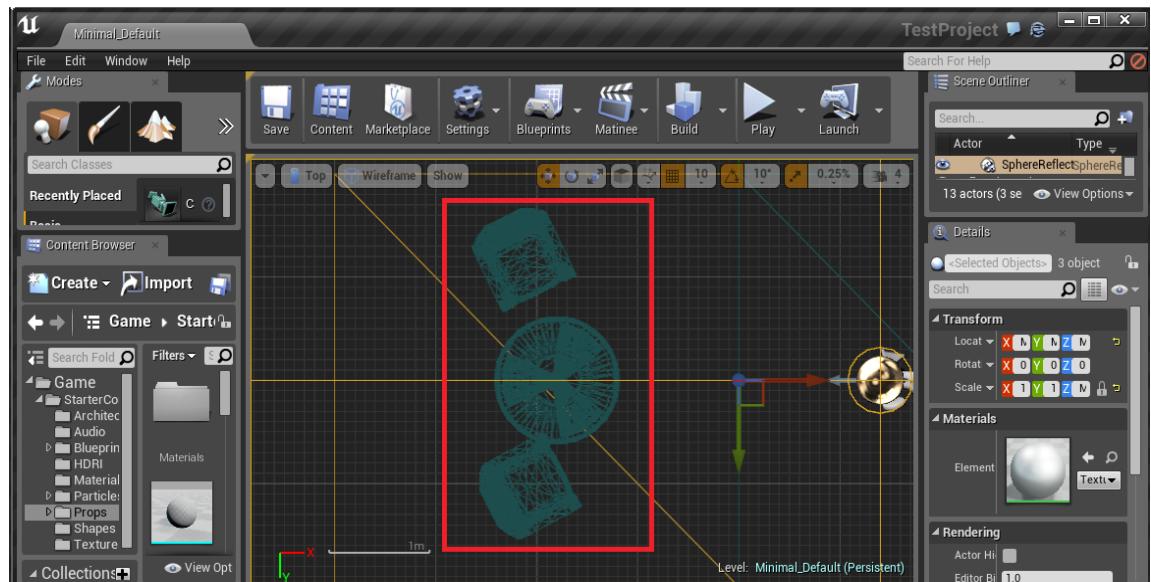


Illustration 13: Affichage en mode vue de dessus (Top)

Nous avons vu comment nous déplacer dans le **viewport**, voyons maintenant quelques manipulations:

1. Cliquez sur le bouton «Perspective» et choisissez la vue orthographique «Top» (ou **Alt + J**).

Il s'agit de la vue de «dessus» de la scène, c'est à dire que l'axe Z disparaît de l'affichage et que l'on passe en 2D. Le déplacement dans cette vue est très différent: le bouton droit de la souris permet de se déplacer, la molette permettant de gérer le niveau de zoom (la hauteur en réalité).

2. Centrez la vue pour obtenir une disposition proche de l'illustration 13.

3. Placez le pointeur de la souris en haut à gauche puis cliquez tout en gardant le bouton enfoncé et déplacez le curseur en bas à droite pour obtenir un rectangle tel que l'encadré rouge, puis relâchez.

Vous venez de sélectionner les 3 objets, ainsi que le plan se situant juste en dessous. De la même façon que vous avez appris à déplacer un objet, faites une rotation ou une redimensionnement, il est toujours possible de réaliser ces opérations sur tous les objets sélectionnés.

4. Appuyez sur **Ctrl + A**, vous venez de sélectionner tous les objets de la scène. Enfin, **Esc** permet de désélectionner tout.

5. Cliquez sur la table du milieu: seule la table est sélectionnée. Maintenez la touche **Shift** enfoncée et cliquez successivement sur les deux chaises. **Vous venez de sélectionner les 3 objets.**

6. Maintenez maintenant la touche **Ctrl** et cliquez de nouveau sur la table: vous venez d'exclure la table de la sélection.

7. Le bouton central de la souris sert à mesurer les distances: cliquez avec ce bouton sur l'écran et déplacez la souris, le résultat s'affiche en temps-réel durant le déplacement. Il est exprimé en unités «UU». Par défaut, ce sont des centimètres.

Le temps est venu de vous montrer la différence entre transformations globales et locales. Dans l'illustration 13, vous constatez qu'il y a deux chaises. Ces deux chaises, par défaut, possèdent une rotation.

8. Cliquez sur la première chaise pour la sélectionner. Puis cliquez sur l'icône permettant le changement de mode (illustration 14, encadré rouge). Observez le changement sur les axes de déplacement de la chaise.

Le petit «globe» signifie que vous êtes en mode transformations «globales» - c'est un système commun à tous les objets. Le petit «cube» (comme dans l'illustration 13) signifie que vous êtes en mode transformations «locales». Dès qu'un objet a subi une rotation, ses axes seront différents d'un mode à un autre. Notez qu'il n'y a aucun changement si vous êtes en «redimensionnement».

Quelques mots sur la notion de grille: sur l'illustration 14, encadré vert, vous noterez que la grille est active (fond orange). Si ce n'est pas le cas, activez-la en cliquant sur la grille. Juste à sa droite, c'est le pas (valeur d'espacement minimal dans la grille), ici il est de 10.

9. Changez-le et indiquez un pas de 100.

Notez immédiatement que la grille gris clair qui correspond au fond de la scène s'est immédiatement modifiée: des petits carrés, nous sommes passés à de plus grands.

10. Maintenant, déplacez une des chaises.

Notez qu'il est impossible de la placer exactement là où on souhaite. C'est la grille qui fait cela. Désactivez la grille en cliquant sur l'icône, et là vous pouvez placer l'objet où vous souhaitez. Le système de grille magnétique est très pratique quand il s'agit de déposer des objets en respectant un certain alignement.



Illustration 14: Barre d'outils de la fenêtre de rendu (viewport)

Sur l'illustration 14, encadré bleu, il y a deux outils supplémentaires: ils jouent exactement la même fonction pour la rotation (ici, un pas de 10°) et pour le redimensionnement (ici, un pas de 0.25). Jouez un peu avec les valeurs et faites quelques tests de rotation et de redimensionnement pour acquérir ces notions.

11. A droite de la barre d'outil, vous avez une petite icône – cliquez sur cette dernière et vous passez en mode 4 vues simultanées, la célèbre vue des «modeleurs» 3D.

Vous avez ainsi la possibilité d'afficher la vue «side» (vue de coté, axe Y) ainsi que la vue «front» (vue de devant, axe X).

Différents types d'affichage

Le passage de la vue «perspective» à la vue «Top» n'a pas seulement changé la position et l'orientation de la caméra, nous sommes également passés du mode d'affichage «Lit» au mode «Wireframe». C'est ce qui est indiqué juste à coté du nom de la vue:

- **Lit:** rendu final de l'objet, tous les matériaux et lumières sont appliqués
- **Unlit:** comme Lit, mais sans les lumières
- **Wireframe:** seule la géométrie des objets est rendue, sous la forme de polygones, sans textures ni matériaux
- **Detail Lighting:** un matériau «neutre» est appliqué à tous les objets, incluant les normal maps
- **Lighting Only:** la même chose sans les normal maps

Nous pourrions lister tous les autres modes, mais il nous semble inutile d'aborder tout de suite des notions complexes qui sont liées pour la plupart aux optimisations que l'on peut faire dans une scène. Cela nous obligerait à aborder tout de suite les notions de shaders, de lumières statiques ou dynamiques, de map de réflexion, etc.



Le «**normal mapping**» est une technique qui consiste à simuler un effet de relief sur la surface d'un objet en utilisant une texture spécifique appelée «normal map». Plus particulièrement, il s'agit de modifier la façon dont la lumière se comporte en chaque point du matériau grâce à la «normal map» qui contient en chaque point la valeur du vecteur normal à la surface. Ainsi, on peut «dévier» la lumière et faire croire que l'objet a un relief particulier. C'est un procédé très peu coûteux en ressources qui apporte un réalisme important au matériau d'un objet. Créer un normal map peut être réalisé à l'aide d'outils spécifiques (comme «crazy bump» ou le plugin «insane bump» pour «The Gimp») qui arrive à «simuler» ce type d'effet à partir de textures simples. L'idéal reste d'être en possession de véritables textures numérisées à partir d'un scanner 3D.

Configuration des fenêtres de l'éditeur

L'environnement de l'éditeur, c'est à dire la position de chaque sous-fenêtre, n'est pas figé. Vous pouvez très facilement réorganiser l'ensemble des fenêtres. Nous allons faire un essai: cliquez sur le nom «Content Browser» (zone 3) et en maintenant le bouton enfoncé déplacez la souris vers le centre du viewport.

Une sorte de grille va s'afficher, comme dans l'illustration 15: il s'agit de l'ancre. Selon la zone (1 à 5) où vous allez déplacer le pointeur et relâcher le bouton de la souris, la fenêtre **CB** va s'ancre de différentes façons dans **le viewport**.

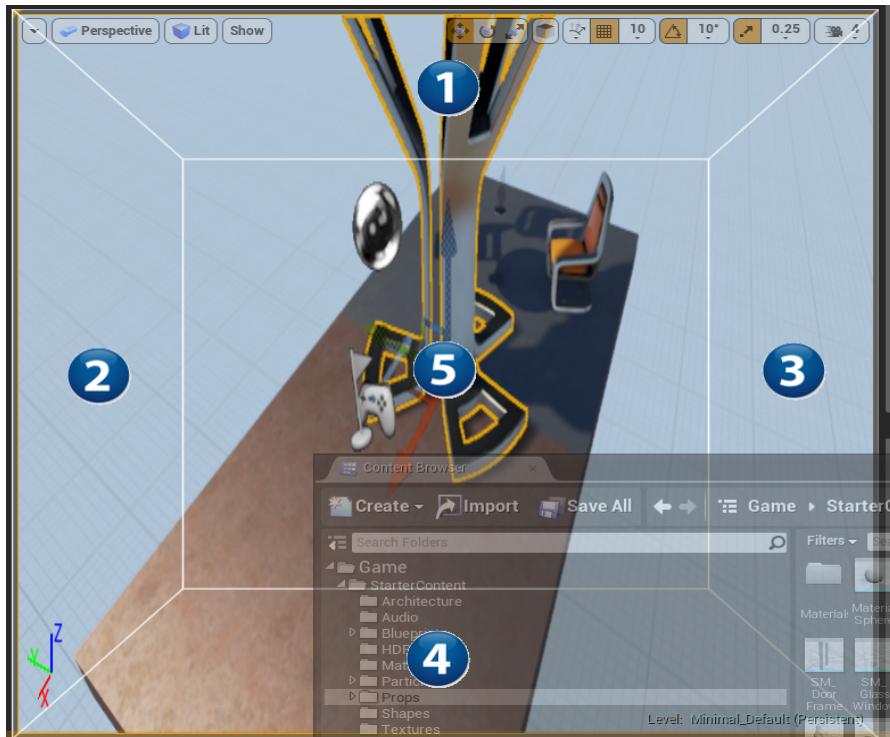


Illustration 15: Déplacement et ancrage des fenêtres

Si par exemple vous choisissez la zone 4, alors la fenêtre d'origine (viewport) se sépare en 2 zones: la première en haut contiendra toujours le viewport et la seconde, en bas, contiendra **CB**. C'est un coup à prendre, mais faites plusieurs essais avec différentes fenêtres et vous pourrez constater que toutes les combinaisons sont possibles en quelques clics.

Si toutefois vous préférez avoir une fenêtre volante, il suffit de choisir la zone 5. C'est très utile si vous travaillez avec plusieurs écrans par exemple.

Autre possibilité, ancrer la fenêtre comme volet principal de l'éditeur: il suffit de réaliser la même opération, mais en relâchant la fenêtre dans la zone 1 de l'illustration 7, et ce, juste à côté du nom du volet actif «Minimal_Default». C'est vrai avec **CB**, mais vous ne pourrez pas faire de même avec toutes les fenêtres.

D'ailleurs, allez faire un tour dans le menu principal «Windows», et affichez la fenêtre «Developper Tools → Output Log». Cette fenêtre peut s'ancre également dans la liste des volets principaux. Comme vous pouvez le constater, il y a de très nombreuses fenêtres listées dans le menu «Windows», ce qui signifie de nombreuses configurations possibles. **Heureux celui qui possède 3 écrans** pour travailler avec UE – vous vous en rendrez encore plus compte en utilisant les Blueprints!

Quand vous en avez assez, vous avez deux possibilités:

- La configuration est meilleure que celle d'origine (attendez peut-être d'en savoir plus), dans ce cas vous pouvez **sauver la configuration** : menu «Windows» → «Save Layout».
- Pour **retrouver la config d'origine**: menu «Windows» → «Reset Layout».

Le système d'aide intégré

Le système d'aide intégrée à l'éditeur est relativement bien fait car il est accessible de bien des façons.

A tout moment, vous pouvez presser **F1** pour accéder à l'aide en ligne (aide contextuelle).

L'illustration 16 montre l'enchaînement d'apparition des bulles d'aides : la première bulle apparaît lorsqu'on survole, par exemple, «Empty Actor» dans la fenêtre «Modes» de l'éditeur de niveau. Si on presse **Ctrl + Alt**, une autre bulle remplace la première (avec généralement plus de détails ou un lien vers la documentation de l'objet).

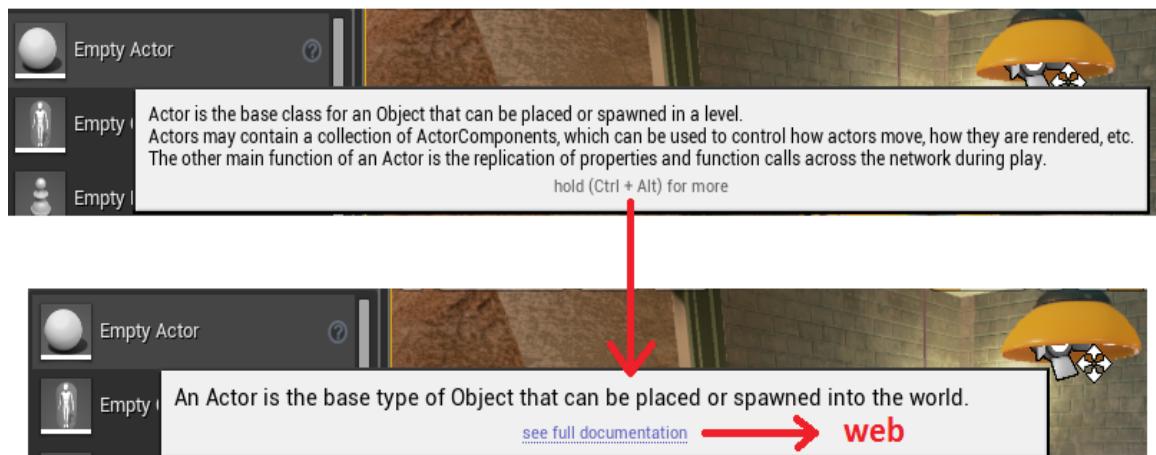


Illustration 16: Bulles d'aides

A partir de la barre de menu, vous pouvez également accéder au **menu «Help»** donnant accès à différentes sections de l'aide:

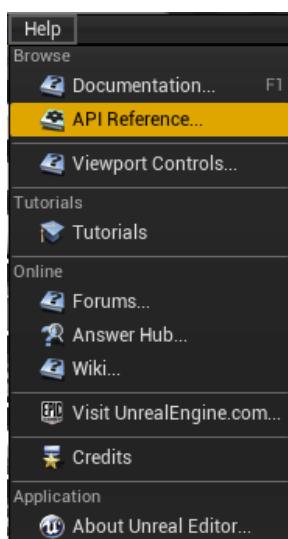


Illustration 17: Help Menu

- **Documentation:** équivalent du **F1** vu précédemment.
- **API Reference:** accès direct et en local au fichier d'aide (.HLP) installé sur votre ordinateur (ce qui peut être pratique en cas de coupure internet). On y trouvera toutes les fonctions utilisées dans les Blueprints ou en programmant en C++
- **Viewport Controls:** raccourci rapide vers l'aide en ligne correspondant aux différentes touches de raccourcis ou contrôles réalisés à la souris sur la zone de rendu 3D au centre de l'écran.
- **Tutorials:** permet d'accéder aux différents tutoriaux classés par thème (nous verrons un peu plus bas une autre façon d'y accéder).
- **Forums:** très utile quand vous rencontrez un problème que vous n'arrivez pas à résoudre, c'est un raccourci rapide vers le forum de discussion en ligne officiel d'Unreal Engine.
- **Answer Hub:** autre façon de recevoir de l'aide. On arrive sur une section du site d'Unreal où l'on peut poser une question et recevoir de l'aide d'un des techniciens d'Epic.
- **Wiki:** documentation non-officielle d'Unreal Engine. Non-officielle, dans le sens où vous, ainsi que tous les internautes, êtes invités à participer à la rédaction de cette documentation (sur le mode de Wikipédia).

Le menu «Help» que nous avons ici (figure 17) est celui de l'éditeur de niveau. Il sera différent pour l'éditeur de matériaux, de composants, etc.

A coté de certains éléments, vous trouverez également une petite icône «?» (illustration 18) vous invitant à consulter la documentation associée.



Illustration 18: icône "?" d'aide

On peut aussi rechercher directement par mot clé en utilisant le champ de recherche en haut à droite de l'écran (zone «Search for Help»). L'illustration 19 nous montre un exemple de recherche sur les mots clés «Set Light Color» : la recherche s'est faite à partir de plusieurs sources. Ici nous avons des réponses sur la documentation en ligne, mais également sur le Answerhub que nous avons vu précédemment.

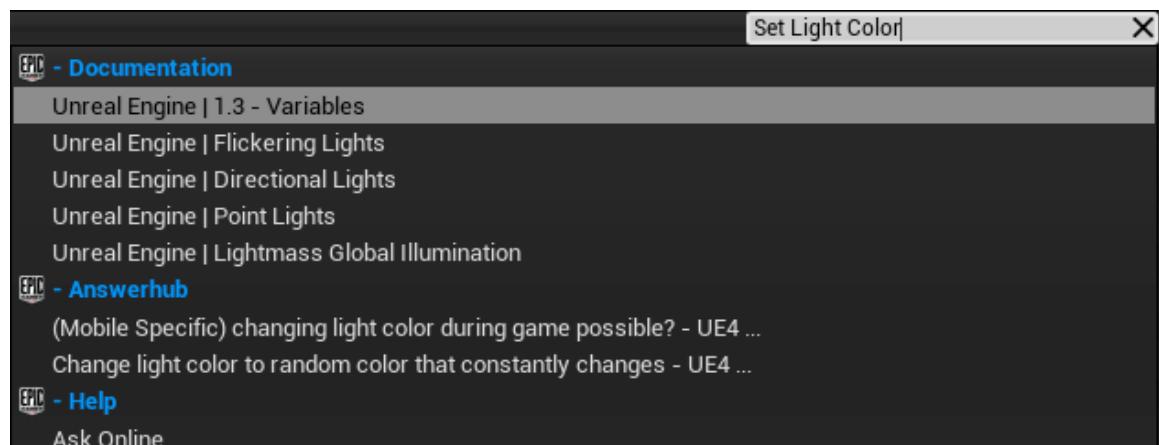


Illustration 19: Aide, Recherche par mots clés – exemple sur «Set Light Color»

L'éditeur est également doté d'une sorte de compagnon qui nous invite à exécuter l'un des tutoriaux sous forme guidée. Il suffit de cliquer sur le bouton cerclé de rouge de l'illustration 20 pour faire apparaître une nouvelle fenêtre. Ensuite, suivez le guide en parcourant le texte et en cliquant sur «Next» (suivant).

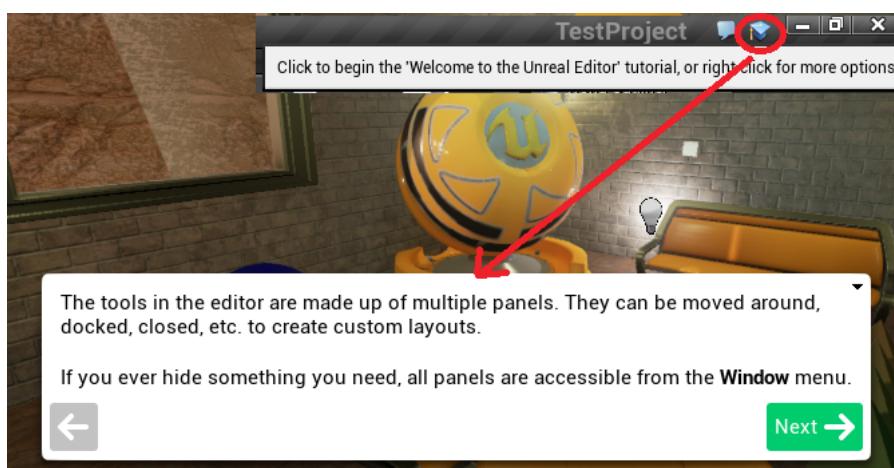


Illustration 20: Système d'aide type compagnon

Vous pouvez aussi ajouter vos propres actors d'aide («**Documentation Actor**») dans votre scène et faire un lien vers la documentation en ligne ou toute autre lien hypertexte vers la page d'un site.

CRÉATION D'UN NIVEAU

Nous venons de faire un tout petit tour d'horizon de l'éditeur. N'hésitez pas à jouer un peu avec ce dernier.

Dans **CB**, vous trouverez divers matériaux sous le répertoire «Materials». Il vous suffit de les «glisser-déplacer» sur un objet comme le sol du niveau pour que ce matériau s'applique automatiquement.

Testez également les «Particles»: vous pouvez ajouter de la même façon un feu, de la fumée ou même une explosion (testez, vous ne risquez rien, c'est virtuel). Nous reviendrons bien sûr sur toutes ces notions, mais n'hésitez pas à vous faire plaisir, c'est le principe même d'un bon apprentissage.

Le niveau d'un jeu peut être composé d'un paysage et d'un ensemble d'assets (rochers, arbres, plan d'eau, etc.). Toutefois, nous n'en sommes pas encore là. L'idée est ici de vous montrer les outils de création que sont les «brushes». Vous allez voir que pour certaines opérations, vous n'êtes pas obligé de passer par un éditeur externe comme 3DSMax, Blender ou Maya. Vous pouvez déjà créer des formes relativement complexes. L'éditeur possède un certain nombre d'outils qui vous permettent de créer des bâtiments, des escaliers, des routes...

Passons à la pratique:

- 1.** Ouvrez votre précédent projet «TestProject».

Quand on ouvre l'éditeur (launch «unreal Engine» à partir de l'interface de démarrage), il est listé dans le volet «Projects» sous la catégorie «My Projects». Vous pouvez également passer par l'interface de lancement et aller dans la section «Library».

- 2.** Menu «File» → «New Level», choisissez «Default».

Examinez **WO**: Le niveau n'est pas complètement vide (cela aurait été le cas si nous avions choisi «Empty Level»). On retrouve les éléments de l'illustration 21 :

- Tout d'abord, l'actor «**Atmospheric Fog**» permet d'ajouter un brouillard léger. Pour simplifier, plus un objet est loin, plus il prend la couleur définie par le brouillard.
- «**BP_Sky_Sphere**» est le ciel animé, mais c'est aussi la position du soleil dans le ciel.
- «**Light Source**» est une lumière directionnelle représentant le soleil – elle possède un angle, mais pas d'origine précise.
- «**Player Start**» représente la position du joueur au lancement du jeu.
- Et enfin «**SM_Template_Map_Floor**» est un «StaticMesh», c'est à dire un modèle qui n'est pas animé – c'est le plan sur lequel le joueur peut marcher, c'est à dire le sol.

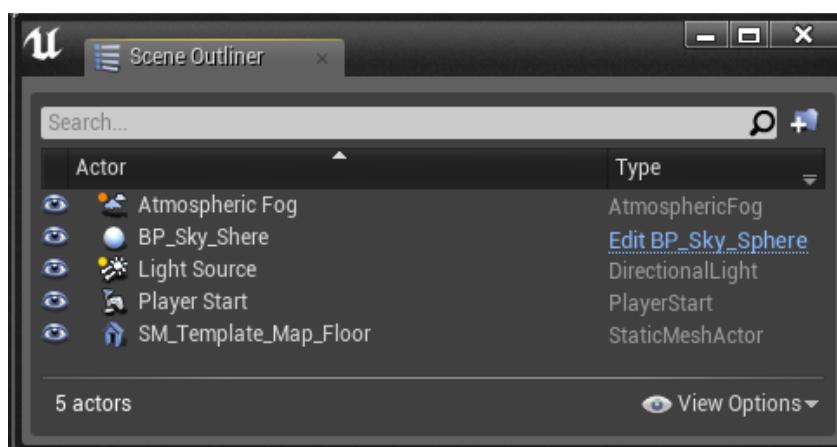


Illustration 21: "World Outliner" (ou **WO**) d'un niveau de type "Default"

3. Pour choisir le niveau est chargé par défaut lors de l'ouverture du projet: dans la barre d'outils, au dessus de la vue «Perspective», cliquez sur «Settings» et choisissez «Project Settings» (Illustration 22).
4. Cliquez sur «Maps & Modes» sous «Project» dans le menu de gauche, et allez dans «Default Maps».
 - «Game Default Map» est le niveau chargé lors du lancement du jeu.
 - «Editor Startup Map» est le niveau chargé lors de l'ouverture du projet sous l'éditeur.

Pour les deux, choisissez «MyFirstLevel», puis fermez la fenêtre. «MyFirstLevel» sera le niveau chargé dans les 2 cas.

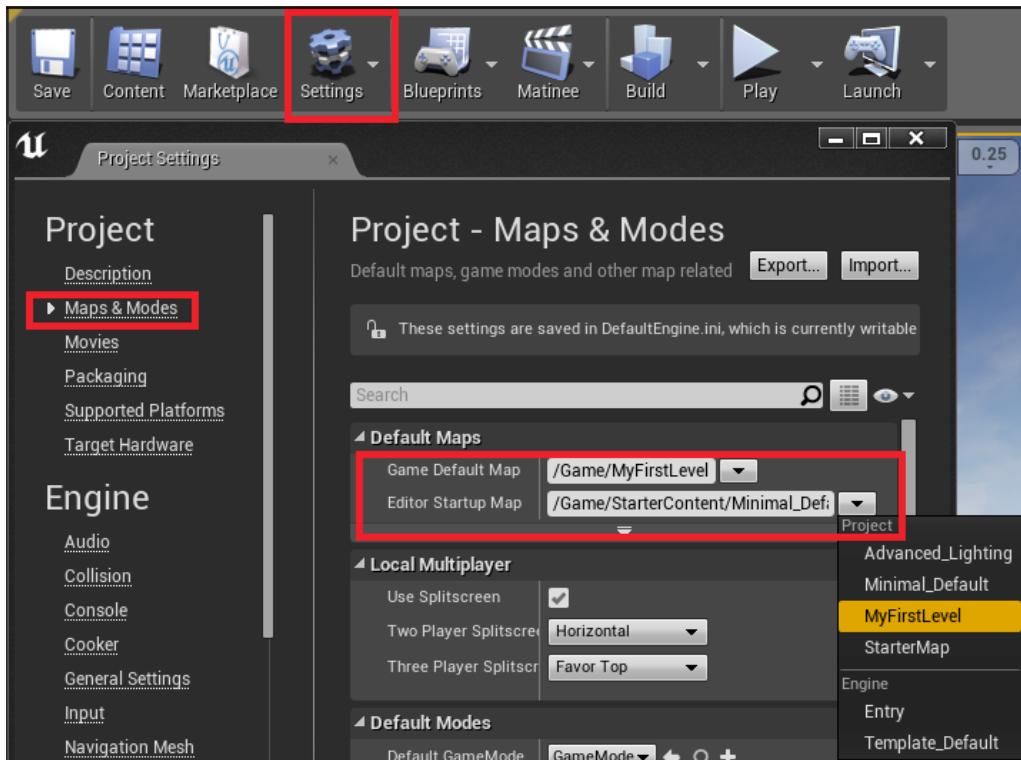


Illustration 22: Propriétés du projet, choix du niveau de départ

Nous allons commencer par sauver notre niveau vide:

5. Menu «File» → «Save», nommez le niveau «MyFirstLevel» et cliquez sur le bouton «Save».
- Il sera enregistré directement au niveau de «Game», alors que les autres sont situés dans «StarterContent». A partir de maintenant, vous pouvez fermer le projet puis le rouvrir, vous arriverez directement sur ce niveau.

Modélisation d'un premier bâtiment

Avec ce premier exemple, nous allons apprendre à utiliser plusieurs outils de modélisation:

1. Dans **MD**, choisissez «BSP» puis «Box» et glissez-déposez la box dans le viewport.
2. Choisissez une «grille» de 50 et **activez** cette grille.
3. Dans **DT**, panel «Brush Settings», indiquez les valeurs «X:800», «Y:800», «Z:400». Nous avons créé une structure de 8m de large, 8m de long et 4m de hauteur.

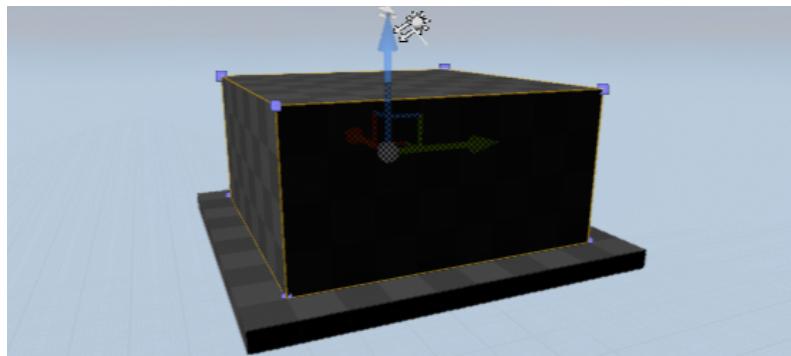


Illustration 23: Premier jet de notre bâtiment

4. Passez en vue «Top» et déplacez la structure au centre du plan représentant le sol.
5. Ensuite, en vue «Side», placez la structure juste au dessus du sol. Vous aurez besoin de passer la grille en pas de 10 pour réaliser cette opération.
6. Puis, revenez en vue «Perspective». Vous obtenez un visuel proche de l'illustration 23.
7. Sélectionnez le nouvel objet créé, voyez que dans **WO** il se nomme par défaut «Box Brush». En effet, il ne s'agit pas d'un objet, mais d'un volume qui sert à modéliser pour le moment.
8. Dans **MD**, cliquez sur le volet «Geometry Editing». Sélectionnez la face du dessus en cliquant sur cette dernière. Elle se colorie pour indiquer qu'elle est bien sélectionnée.
9. Cliquez sur «**Extrude**» puis utilisez l'outil déplacement et la flèche rouge pour déplacer la face sélectionnée vers le haut. Déplacez la d'un quart de la hauteur du bâtiment environ.

En relâchant, vous remarquez qu'au lieu de déplacer la face et agrandir la structure vers le haut, nous avons créé une sorte de boîte supplémentaire (Illustration 24).

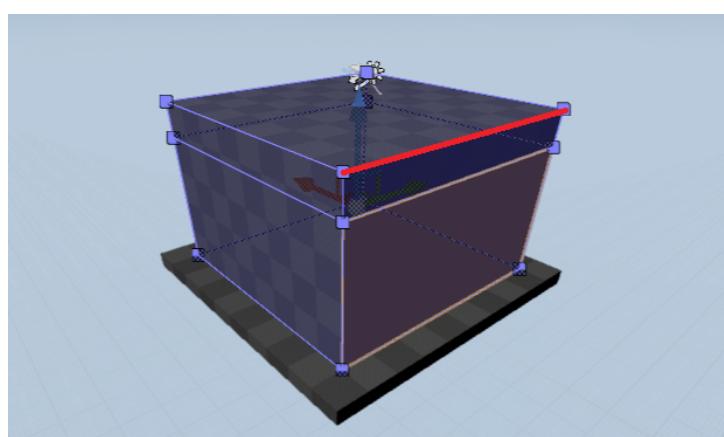


Illustration 24: Utilisation de l'outil "Extrude"

Il n'y a pas que les faces que l'on peut sélectionner. Essayez avec une des arrêtes (en rouge sur l'illustration 24) - ce n'est pas toujours évident car il faut bien viser, mais une fois que l'arrête est sélectionnée, elle se colore en conséquence.

10.Dans **MD**, cliquez sur «**Split**» (diviser).

Automatiquement, le bâtiment sera divisé en deux, ou plus particulièrement vous verrez apparaître de nouveaux points et de nouvelles arrêtes, car de l'extérieur, l'objet n'a pas changé.

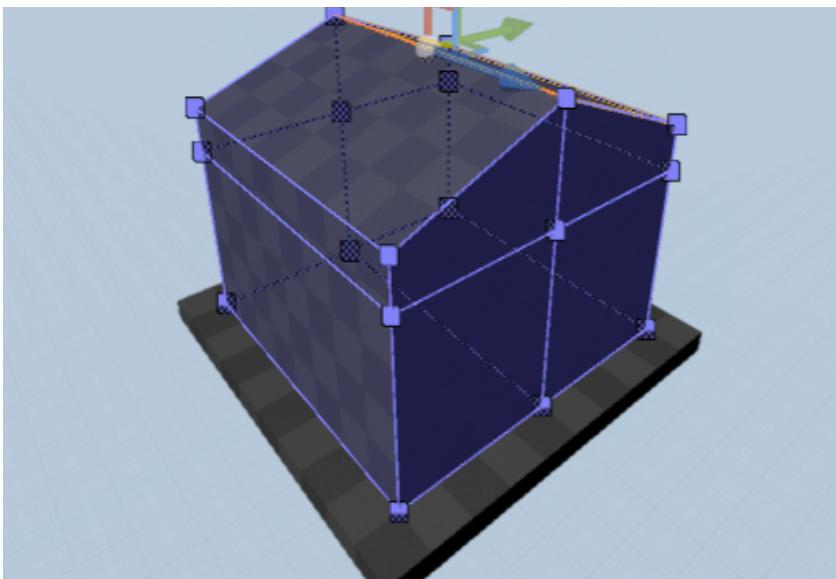


Illustration 25: Utilisation de l'outil "Split"

11.Sélectionnez ensuite l'arrête qui sépare la face au dessus du bâtiment. Puis, avec l'outil «déplacer», elevez l'arrête d'un quart de la hauteur du bâtiment également. Vous obtenez une structure qui ressemble à une maison (illustration 25).

12.Sélectionnez l'objet «Box Brush» en utilisant **WO**. Puis, faites **Ctrl+C** (copier) et enfin **Ctrl+V** (coller).

Vous ne voyez rien de plus à l'écran, pourtant, vous venez de créer une copie du premier volume de l'objet (sans les extrudes et autres manipulations). Pour s'en convaincre, examinez de nouveau **WO**: vous voyez qu'un objet «Box Brush 2» vient d'apparaître. S'il n'est pas sélectionné, sélectionnez-le.

13.Dans **DT**, panel «Brush Settings», modifiez les éléments suivants: «Brush Type: **Subtractive**», et «Y:750».

Ce que nous venons de faire, c'est de soustraire le second objet au premier. En modélisation, on appelle cela une opération booléenne. Visuellement, le bâtiment est ouvert sur 2 pans.

14.Déplacer légèrement l'objet «Box Brush 2» selon l'axe des X pour faire réapparaître le mur du fond. Vous allez obtenir le même visuel que l'illustration 26.

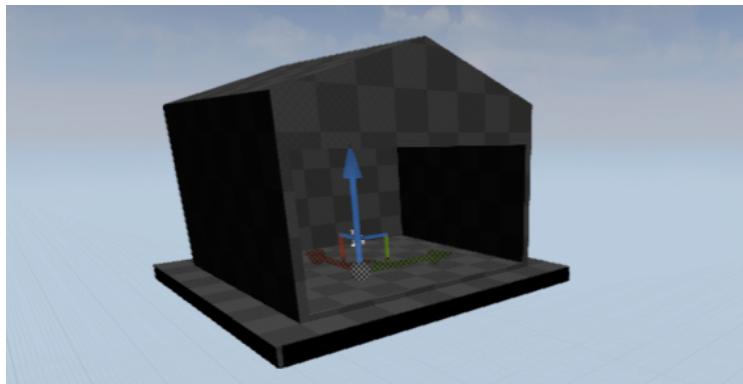


Illustration 26: Application d'une brush "subtractive"

Nous allons ajouter un autre type Brush: un escalier. Pour cela :

15. Allez dans **MD**, toujours dans «BSP», sélectionnez «Linear Stair» et glissez-déposez sur la scène. Notez au passage qu'il existe 3 types d'escalier.
16. Avant d'effectuer le positionnement, allez dans **DT**, panel «Brush Settings» et indiquez «Step length: 60» (profondeur des marches), «Step Height: 40» (hauteur des marches), «Num Steps: 20» (nombre de marches).

Ce que vous pouvez constater, c'est que ce type de «Brush» est relativement configurable.

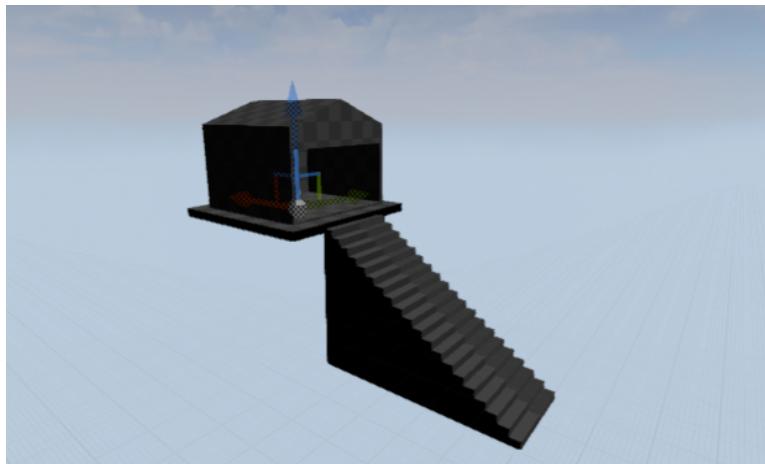


Illustration 27: Ajout d'un escalier

17. Passez en vue «Top» et «Front» pour positionner les marches juste devant la maison comme sur l'illustration 27. Utilisez l'outil de redimensionnement pour agrandir la largeur des marches (nous aurions aussi pu modifier «Step Width» dans **DT** panel «Brush Settings», les deux démarches ne sont pas incompatibles).
18. Sélectionnez «Player Start» dans **WO** et déplacez-le pour le positionner en bas des escaliers.
19. Testons le jeu: dans la barre d'outils du viewport, cliquez sur «Play».

Le joueur est placé dans la pièce principale, essayez de rejoindre l'escalier. Par défaut, vous volez! Ce n'est pas très pratique pour tester un escalier. Plus tard, nous verrons comment ajouter un véritable acteur, mais nous en sommes pour l'instant à la modélisation!

Nous allons maintenant **mettre un peu de couleurs** dans tout cela en appliquant des matériaux sur les surfaces du bâtiment:

20. Dans **CB**, «Materials» (de StarterContent), sélectionnez le matériau «M_Cobblestone_Rough».
21. Cliquez sur l'une des marches pour la mettre en surbrillance, effectuez un glisser-déplacer du

matériau vers la marche. Celle-ci va automatiquement intégrer le nouveau matériau.

22.Sélectionnez une autre marche, «**Shift**+clic gauche» sur cette dernière.

C'est un raccourci pour appliquer le dernier matériau sélectionné. Toutefois, vous vous doutez que nous n'allons pas devoir répéter l'opération pour chaque marche!

23.Sélectionnez une marche, puis dans **DT**, panel «Geometry», cliquez sur «Select» et choisissez «Select Matching Brush»: toutes les faces de la brush active sont sélectionnées. Retenez le raccourci clavier qui est bien pratique **Shift**+**B**.

24.Répétez l'opération en effectuant un glisser-déplacer du matériau vers l'escalier. Ainsi, toutes les marches prennent le bon matériau (illustration 28)

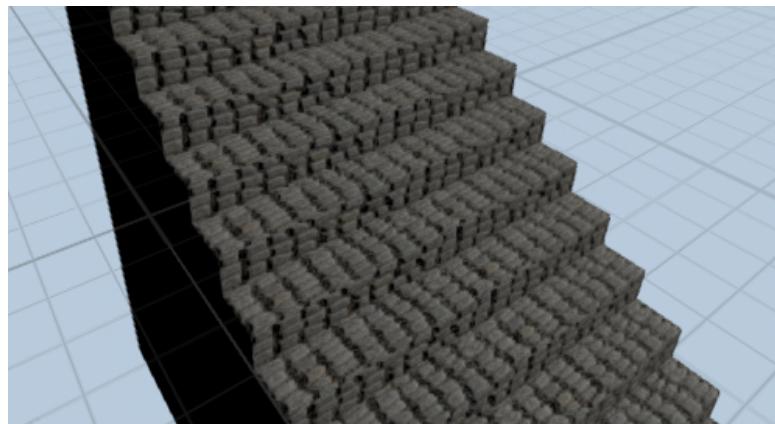


Illustration 28: Application du matériau sans travaux sur les UVs

Ce n'est pas forcément très joli, nous allons travailler la façon dont la texture est appliquée (c'est à dire changer l'«UV Mapping» pour les connaisseurs).

25.Tout en ayant les marches sélectionnées (sinon, refaire un **Shift**+**B**), allez dans **DT**, panel «Surface Properties», modifiez «scale U: 4» et «scale V:4», puis cliquez sur «Apply».

La texture appliquée a été zoomée d'un facteur de 4. Si nous avions utilisé «0.5» comme valeur, le zoom aurait été divisé par deux.

26.Toujours dans «Surface Properties», cliquez sur «Rotate 90»: la texture est alors tournée avec un angle de 90°, ce qui est plus joli pour notre escalier.

Toutefois, on peut encore faire un petit réglage:

27.cliquez sur l'un des «pan: 1/256» et observez le résultat.

Cela déplace la texture sur la surface. Appuyez plusieurs fois afin d'obtenir le meilleur placement des pierres sur les marches (illustration 29)

Nous allons maintenant nous occuper des côtés de l'escalier. Pour cela :

28.Sélectionnez une face sur l'un des côtés de l'escalier, et **Shift**+**C** pour sélectionner toutes les surfaces coplanaires (nous aurions aussi pu passer par **DT**, panel «Geometry», bouton «Select» et «Select All Coplanar Surfaces»).

29.Puis, appliquez le matériaux «M_Rock_Basalt».

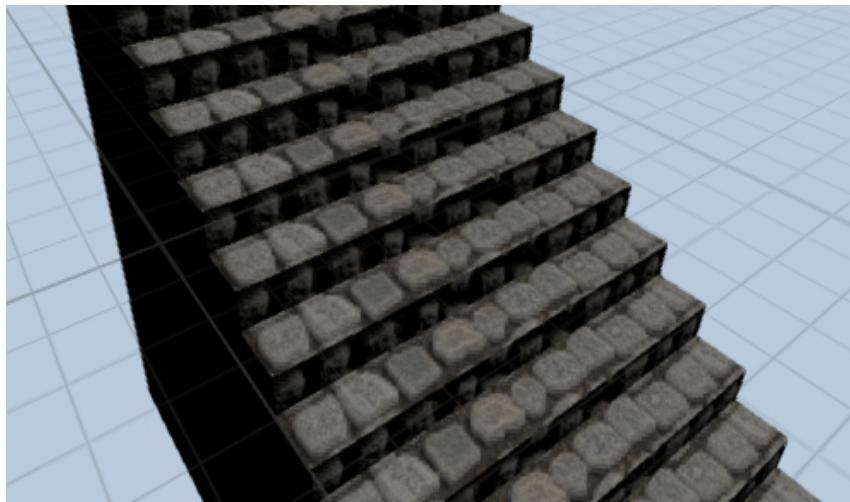


Illustration 29: Matériaux après un "scale", "rotate" et "pan"

Nous aurions préféré que l'escalier semble taillé d'un même bloc – pour cela :

30. Allez dans **DT**, panel «Geometry», cliquez sur le bouton «Alignment» et sélectionnez «Align Surface Planar».

Ainsi, toutes les surfaces sont traitées comme une seule lors de l'application de la texture. Faites de même pour l'autre côté de l'escalier. Si vous rencontrez des difficultés à cause des sources lumineuses, n'hésitez pas à passer en mode «Unlit» grâce à la barre d'outils le temps d'effectuer les opérations.

Désormais, vous connaissez les principales techniques permettant d'appliquer les matériaux sur les objets. Vous pouvez donc réaliser ces mêmes opérations pour le reste du bâtiment.

31. Cliquez sur «Play» et allez visiter votre pièce.

Quelque chose ne fonctionne pas... les murs sont étrangement éclairés, ce n'est pas joli... et ce n'est pas normal! Quittez le mode jeu avec **Esc**. Vous venez d'expérimenter un problème lié à l'éclairage – nous n'avons pas encore réalisé un «Build». Il s'agit principalement des calculs de lightmaps (simulation de lumière en utilisant des textures) et diverses opérations de précalculs qui permettent des optimisations pour un rendu temps-réel. Mais avant de réaliser un «Build» :

32. sur le petit combo juste à coté du bouton «Build» de la barre d'outils.
33. Un menu se déroule, sélectionnez «Lighting quality», puis «Production», pour avoir le meilleur calcul d'éclairage possible.
34. Cliquez sur «Build».

Après un certain temps, le message «Lighting Build Complete» apparaît.

35. Cliquez sur «Play» et allez visiter votre pièce de nouveau. Un peu sombre au fond de la pièce... mais nous allons y remédier.
36. Sélectionnez l'actor «Light Source» et allez dans **DT**, panel «Transform» et dans «Rotation» indiquez les valeurs: «X:45», «Y:-15» et «Z:-15».
37. Puis relancer un «Build».

Nous avons déplacé la source lumineuse correspondant au soleil. Remarquez tout de suite une modification sur l'ensemble de l'environnement: le soleil étant plus bas, tout se teinte d'une couleur orangée symbolisant le coucher de soleil. L'effet est plutôt réussi.



Une **texture** est une image qui est «plaquée» sur un objet pour donner une «impression» de détail. Par exemple, une texture «herbe» peut être plaquée sur un plan pour représenter une surface végétalisée. Les textures doivent être correctement mappées, c'est à dire dotées d'un UV-Mapping adapté: c'est le procédé du dépliage UV qui permet d'obtenir un bon UV-mapping. De nombreux paramètres sont liés à la texture: indice de transparence, influence sur les couleurs, déformation géométrique (équivalent du bump map, mais plus élaboré en déformant littéralement l'objet dans sa géométrie), etc. Il faut noter qu'on peut appliquer plusieurs niveaux de texturage et gérer la façon dont les textures se mélangent et s'influencent.

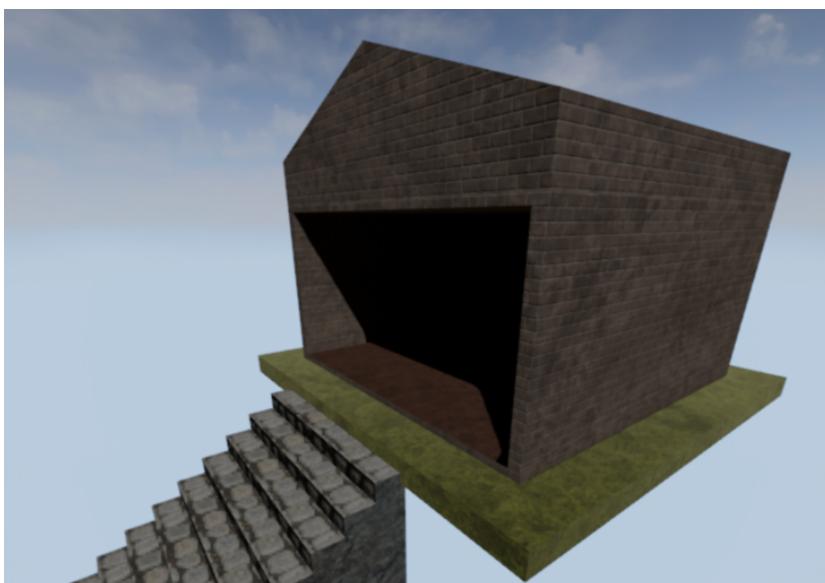


Illustration 30: Bâtiment après application des matériaux et calculs d'éclairage



Un **matériau** (Material) définit certaines propriétés de la lumière renvoyée par l'objet (diffuse, spéculaire, etc.), mais aussi ses caractéristiques physiques. Il peut être composé de plusieurs textures qui peuvent être combinées entre-elles.

Aménageons un peu la scène

Si vous souhaitez vous entraîner un peu, vous pouvez ajouter une table, quelques chaises, un lampadaire, des fenêtres... vous pouvez faire votre marché dans le répertoire «Props» de **CB**. L'illustration 31 est une proposition d'aménagement.



Illustration 31: Exemple d'aménagement de la pièce

Vous l'avez compris, créer un niveau de jeu nécessite de commencer par modéliser un environnement et importer un certain nombre d'assets. Mais il ne faut pas forcément les voir comme des objets séparés. Un bon exemple est celui du rocher. Pour réaliser l'illustration 32, nous avons utilisé 8 actors issus de l'objet «SM_Rock» que nous avons assemblé, en effectuant diverses rotations et redimensionnements. Il faut un peu d'expérience pour réaliser un bon assemblage, mais on arrive rapidement, avec peu d'assets à réaliser des scènes extraordinaires. Pour manipuler facilement l'ensemble des rochers, nous les avons groupés, au lieu de les sélectionner et de presser «**Ctrl**+**G**». pour créer un groupe d'objets, nous utilisons une nouvelle fonction des versions 4.7 : à partir du deuxième rocher, il suffit de le glisser-déposer dans **DT** comme sur l'illustration 33. Au final, on obtient un seul Actor composé de 8 Static Meshes.



Astuce: Il est possible de **sauvegarder le point de vue** de votre choix dans l'éditeur. Exemple: une vue de loin, une vue du bas des escaliers et une vue de l'intérieur du bâtiment. Pour cela, choisissez votre premier point de vue et cliquez sur la petite flèche à gauche de «Perspective» dans la barre d'outils du viewport, rubrique «Bookmarks» → «Set Bookmark» → «Bookmark 0». Prenez votre second point de vue et répéter l'opération avec «Bookmark 1». Pour revenir au premier point de vue, il suffit de faire «Bookmarks» → «Jump to Bookmark 0». Si vous souhaitez effacer les points de vues, vous pouvez sélectionner «Bookmarks» → «Clear All Bookmarks»



Illustration 32: Assemblage d'un même rocher

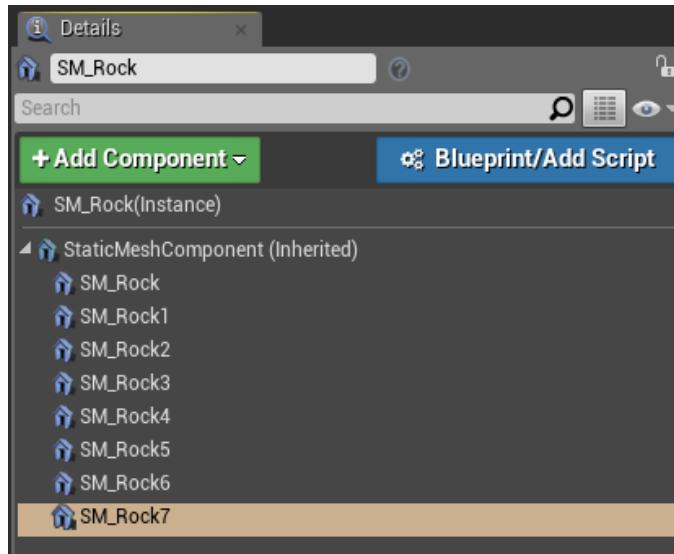


Illustration 33: Nouveauté 4.7: assemblage de plusieurs composants dans un même Actor

CRÉATION DE COMPOSANT PROGRAMMABLE

Nous allons bientôt réaliser notre tout premier programme avec Blueprint. Nous allons **ajouter une lampe qui s'anime dès que le joueur se trouve à proximité**. Cela peut sembler complexe à première vue, mais il s'agit en fait d'un des programmes les plus simples.

1. Dans la fenêtre **CB**, cliquez sur «Game», puis sur «Create» et enfin sur «New Folder». Renommez en «Blueprints».
2. Dans **CB** → «StarterContent» → «Props», sélectionnez l'objet «SM_Lamp_Ceiling» et , sélectionnez «Asset Actions» puis «Create Blueprint using this...».

Une fenêtre s'ouvre et vous invite à choisir la destination:

3. Sélectionnez le nouveau répertoire «Blueprints», dans «name» indiquez «BP_Lamp_Ceiling_Auto» (on nommera toujours les Blueprints en commençant par BP_ afin de mieux s'y retrouver) et cliquez sur «Ok».

Une nouvelle fenêtre d'édition s'ouvre (illustration 34).

Que venons-nous de faire? Plutôt que d'importer directement l'objet dans la scène afin d'obtenir un actor de type «StaticMesh», nous avons créé un composant programmable. Nous allons donc pouvoir intégrer un certain nombre d'interactions à notre composant, voire même un comportement. L'idée est ici de lui ajouter un spot ainsi qu'un «volume déclencheur» (Volume Trigger). Ce déclencheur est une zone qui va générer un événement quand nous allons la franchir et un autre quand nous allons en sortir.

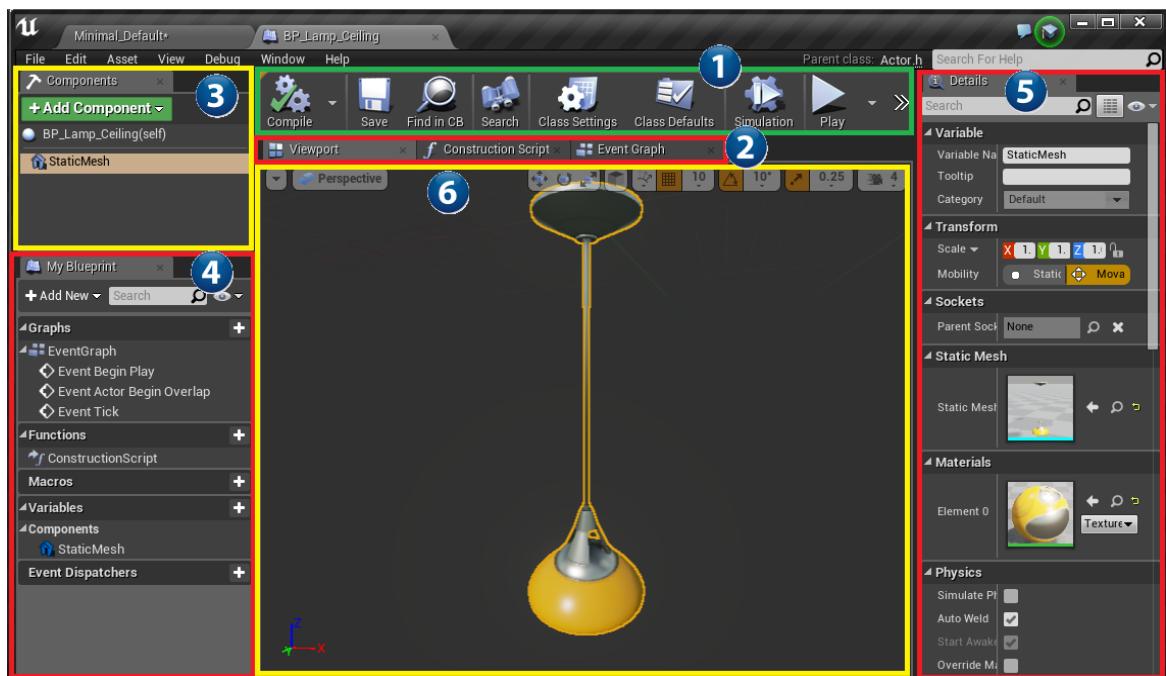


Illustration 34: Éditeur de composant

L'éditeur de composant est probablement le second éditeur que vous utiliserez le plus souvent. Voici les principales zones de cette fenêtre:

- **zone 1:** barre d'outils principale avec les boutons:
 - **«Compile»:** quand vous modifiez un composant, il faut le compiler pour que ces modifications apportées soient appliquées. Concrètement, le composant est un fichier C++ compilé pour être utilisé de façon «optimisée» au lancement du jeu. Il ne s'agit pas d'un script interprété. Vous pouvez donc indépendamment coder en C++ ou utiliser Blueprint, vous n'aurez pas de perte de performance.
 - **«Save»** permet d'enregistrer les modifications. Il est donc possible de tester une modification en compilant, sans enregistrer cette modification – mais l'inverse est vrai aussi. N'oubliez donc pas de compiler systématiquement si vous modifiez votre composant!
 - **«Find in CB»** permet de sélectionner directement le composant dans le **CB**.
 - **«Search»** pour rechercher une fonction ou un événement dans le blueprint en utilisant son nom. Une fois trouvée, il est possible de cliquer sur son nom pour ouvrir le graphe associé.
 - **«Class Settings»** pour accéder au paramétrage de la classe qui est accessible dans **DT**. Il est ainsi possible de modifier la classe parente et d'ajouter une interface par exemple. Dans le cas présent, le Blueprint est dérivé de la classe Actor («Parent class» est rappelé

- juste au dessus de la zone 1). On parle de «dérivation» quand un objet hérite des propriétés de son parent, mais en gardant la possibilité d'avoir d'autres caractéristiques qui lui sont propres.
- «**Class Defaults**»: permet d'accéder au paramétrage du blueprint – cela revient à cliquer sur «BP_Lamp_Ceiling(self)» dans la fenêtre «Components» (zone 3).
 - **Zone 2:** boutons de sélection des volets de l'onglet de la **zone 6**. Au minimum, il y a 3 volets accessibles:
 - «**Viewport**»: volet actuellement ouvert dans l'illustration. C'est la représentation 3D du composant.
 - «**Event Graph**»: contient le programme sous forme de nœuds interconnectés. Nous allons nous pencher sur la partie «Graph» très rapidement.
 - «**Construction Script**»: contient le graphe permettant de construire la classe
 - **Zone 3:** représentation hiérarchique de notre composant. On parle d'un objet «parent» et d'objets «enfants». Tout simplement, cela signifie que les transformations apportées à un parent (déplacement, rotation ou redimensionnement) seront apportées également aux enfants. Mais l'inverse n'est pas vrai. Ici, nous n'avons qu'un seul objet, mais nous allons justement ajouter 2 autres objets par la suite en respectant une hiérarchie. En fonction de classe parente, le composant peut hériter d'un certain nombre d'éléments qui se retrouveront automatiquement ici.
 - **Zone 4 :** Cette zone est liée à la programmation du Blueprint. Nous y reviendrons un peu plus loin.
 - **Zone 5:** La fenêtre **DT** (comme dans l'éditeur de niveau). Ce sont les détails de l'objet sélectionné dans la zone 3.

Pour rappel, vous pouvez ancrer cette fenêtre juste à coté de celle de l'éditeur de niveau en glissant-déplaçant le nom de la fenêtre (ici «BP_Lamp_Ceiling_Auto») juste à coté de «MyFirstLevel» comme nous l'avons expliqué précédemment. C'est plus pratique si vous ne possédez qu'un seul écran.

Continuons notre paramétrage de composant:

1. Dans la fenêtre «**Components**» (zone 3), sélectionnez l'objet «staticmesh» et renommez-le «myLamp» en modifiant «Variable name» du panel «Variable» de **DT**.
2. Tout en sélectionnant l'objet «myLamp» dans la fenêtre «Components», cliquez sur «Add Component» et choisissez «Spot Light».

Automatiquement, ce nouvel objet sera parenté à l'objet précédent. Renommez-le en «SpotLight». Sinon, il est toujours possible de parenter un objet à un autre en cliquant sur le nom du fils, puis en le glissant sur le nom du père.

3. Positionnez le spot juste au niveau de l'ampoule et assurez vous que le spot soit orienté vers le bas comme dans l'illustration 35. Vous disposez ici des mêmes outils pour opérer les transformations que dans la vue «Perspective» de l'éditeur de niveau.
4. Dans **DT**, panel «Light», changez «intensity:50000» (pour une lumière plus forte) et «outer cone angle: 30» afin de diminuer la largeur du cône du spot, «Attenuation radius:5000» (permet d'augmenter l'intensité de la lumière à distance) et «indirect Ligthing intensity: 5» pour que la lumière soit reflétée par les objets sur d'autres objets de façon plus importante. Vous pouvez également choisir une couleur un peu particulière pour mieux observer la différence.
5. Dans le panel «Rendering», décochez la case «Visible». Par défaut, le «SpotLight» est invisible, ce qui revient à avoir une ampoule éteinte.
6. Cliquez à nouveau sur «myLamp» et ensuite sur «Add Component» puis sélectionner «Box» (n'hésitez pas à utiliser le champ de recherche pour vous faciliter la tâche). Renommez en «BoxTrigger», il s'agit de notre déclencheur.
7. Modifiez les caractéristiques de la «box» en allant dans **DT**, panel «Transform», modifiez «scale» en «X: 10», «Y:10» et «Z:10».

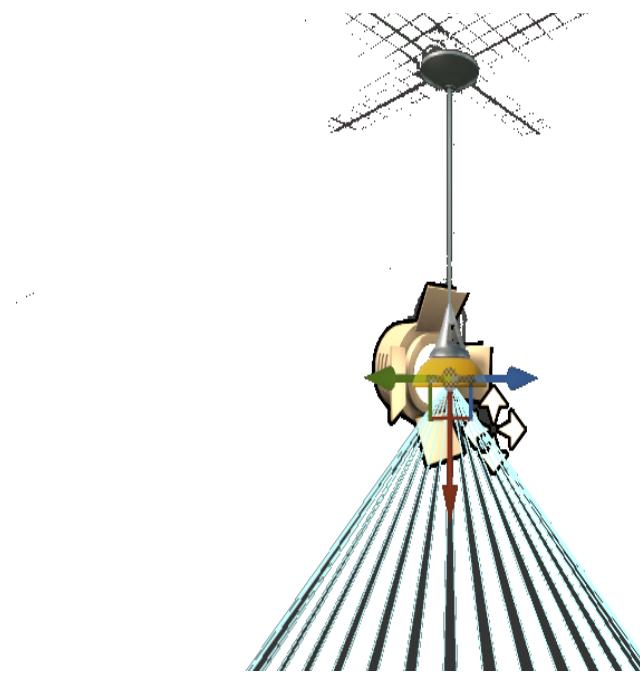


Illustration 35: Spot orienté vers le bas et positionné au niveau de l'ampoule



Astuce: Pour placer une lumière, vous pouvez utiliser une autre approche en prenant le contrôle de cette dernière et en vous déplaçant dans l'éditeur avec les mêmes contrôles que ceux de la caméra principale. Pour cela, dans le viewport, cliquez sur «viewport option», puis sur «Lock Viewport to Actor» et enfin, sélectionnez la lumière. Un petit cadenas va apparaître. Pour relâcher le contrôle, il suffira de prendre le même chemin et de sélectionner «Unlock from ...».

PROGRAMMATION BLUEPRINT



DÉCOUVERTE

Que vous soyez un développeur C++ aguerri ou un graphiste peu intéressé par la programmation, vous devriez trouver la programmation visuelle Blueprint extraordinaire: toute l'API C++ (ou presque) est exposée dans les Blueprints. Quand on travaille proprement (avec des «sequences» et des «functions»), on fait de jolis graphes faciles à maintenir et à partager. Après, dans certains cas, cela peut-être utile de développer ses propres classes C++ et de les appeler via Blueprint: les deux se marient très bien.

Blueprint ne produit pas un code plus lent car, en réalité, les nœuds seront transformés en ligne de code de façon totalement transparente, puis ces dernières seront compilées comme le reste du code C++. D'ailleurs, on peut très bien démarrer un projet Blueprint et ajouter du code C++ ou inversement.



L'API (Application Programming Interface) d'un logiciel est l'ensemble des fonctions de programmation ouvertes à un programmeur externe. Dans le cadre d'UE4, le moteur est programmable en Blueprint ou en C++ grâce à son API.

C'est une sorte de dictionnaire de toutes les fonctions permettant de programmer sous ce moteur. L'API d'UE4 est organisée principalement sous la forme d'un ensemble de classes d'objets. Une référence est disponible à cette adresse:

<https://docs.unrealengine.com/latest/INT/API/index.html>



Voilà, tout est prêt, il ne reste plus qu'à **programmer le composant**:

1. Cliquez sur «Event Graph» dans la zone 2.
2. Sélectionnez «BoxTrigger», puis ⚡ dans la zone réservée au graph («Right-Click to Create New Nodes») et sélectionnez «Add On Component Begin Overlap» comme dans l'illustration 36.

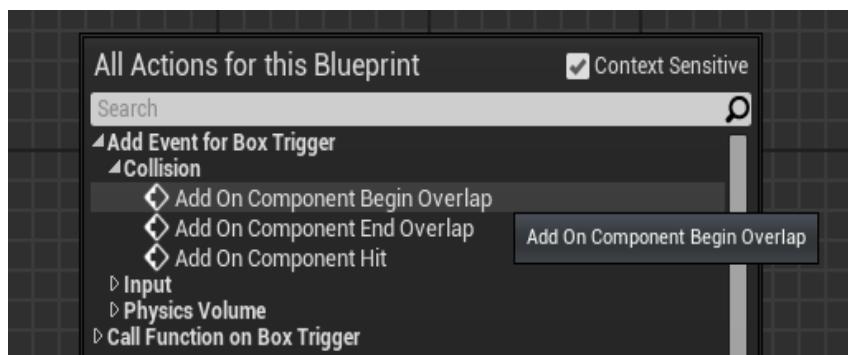


Illustration 36: Déclenchement lorsqu'on entre dans la zone

Un nouveau nœud apparaît – notez qu'à coté du type d'événement, on trouve l'objet concerné (BoxTrigger) entre parenthèses.

3. Observez l'illustration 37: vous allez devoir cliquer sur la petite flèche («▶» cerclé de rouge) qu'on nommera «sortie exec» et tout en maintenant le clic déplacez le pointeur de la souris sur la droite puis relâchez quand vous arrivez sur une zone libre (ce qui est le cas de la presque totalité du graph).



Illustration 37: Création du premier lien

Dès que vous relâchez le clic, une fenêtre s'affiche et vous invite à choisir une action.

4. Sélectionnez «Set Visibility (SpotLight)».

Le nouveau nœud apparaît comme dans l'illustration 37. Il est lié au premier à partir de son entrée «exec».

5. Cochez la case «New visibility» qui signifie que l'objet «SpotLight» sera visible.
6. Répéter les 2 opérations précédentes avec l'événement «OnComponentEndOverlap» et laissez «New visibility» décochée.

Cela signifie que dès qu'on sort de la zone, l'objet «SpotLight» disparaît.

7. Cliquez sur «Compile», puis sur «Save» et retournez dans l'éditeur de niveau.
8. Ajoutez le nouveau composant à la scène et positionnez le au dessus de la table.
9. Effectuez un nouveau «Build» et lancez le jeu.

Déplacez-vous et lorsque vous arrivez dans la zone de déclenchement, la lumière s'allume. Quand vous vous éloignez, elle s'éteint! Un instant magique non? Si ce n'est pas le cas, vous allez voir, il y a une façon très simple de contrôler que le déclencheur fonctionne bien.



Illustration 38: La lampe s'allume ! Fiat Lux !



Astuce: On peut changer la couleur de la lampe par Blueprint en utilisant la fonction «SetLightColor».

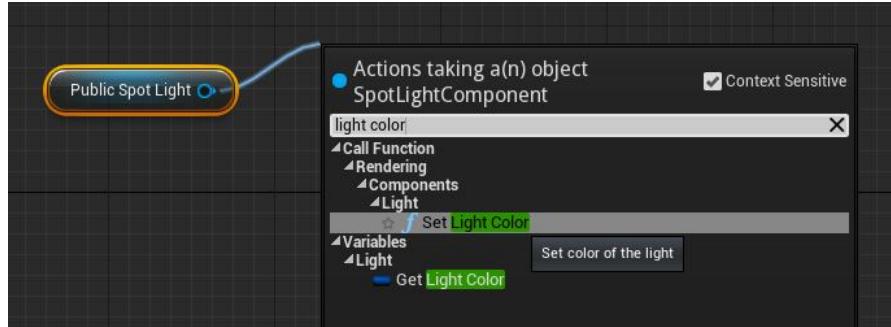


Illustration 39: Fonction Set Light Color applicable aux différentes lampes

Nous ne nous étendons pas d'avantage sur ce graph, même s'il y a beaucoup à dire. Nous préférerons aborder ces différentes notions ultérieurement pour éviter une indigestion.

Vous verrez que Blueprint est très puissant et qu'il peut gérer quasiment tous les cas de figure – ce qui est surprenant lorsqu'on est habitué à coder en C++ et à manipuler des classes d'objet complexes d'une très grosse API.

DÉBOGAGE DU JEU

Introduction au «mode debug»

Si votre programme fonctionne du premier coup, c'est parfait. Mais cela ne sera pas toujours le cas, croyez-moi. Heureusement, Unreal Engine fournit les outils permettant d'aller à la chasse aux bugs.

1. Ouvrez le Blueprint «BP_Lamp_Ceiling_Auto» dans une fenêtre volante et positionnez-la de façon à voir également la scène du jeu.
2. Lancez le jeu

Bien évidemment, quand on possède au moins 2 écrans, c'est plus simple d'avoir les 2 fenêtres visibles. Mais ça reste gérable avec un seul écran.

Examinez l'illustration 40: lorsqu'on pénètre dans la zone, le lien reliant «OnComponentBeginOverlap» et «Set Visibility» s'activent (encadré rouge), puis se désactivent. Et lorsqu'on sort de la zone, c'est au tour du lien reliant «OnComponentEndOverlap» et le second «Set Visibility» de s'activer puis de se désactiver.

Si les liens s'activent mais que la lumière ne s'allume pas, c'est probablement parce que le «Visible» n'est pas coché dans le «Set Visibility» lié à «OnComponentBeginOverlap», ou que le «spotlight» est mal paramétré (intensité trop faible par exemple).

Comme nous n'avons instancié qu'un seul Actor à partir de «BP_Lamp_Ceiling_Auto», c'est celui-ci qui est utilisé pour le débogage. Mais **si vous disposez de plusieurs actors issus de la même classe Blueprint**, sélectionnez celui qu'il faut déboguer.

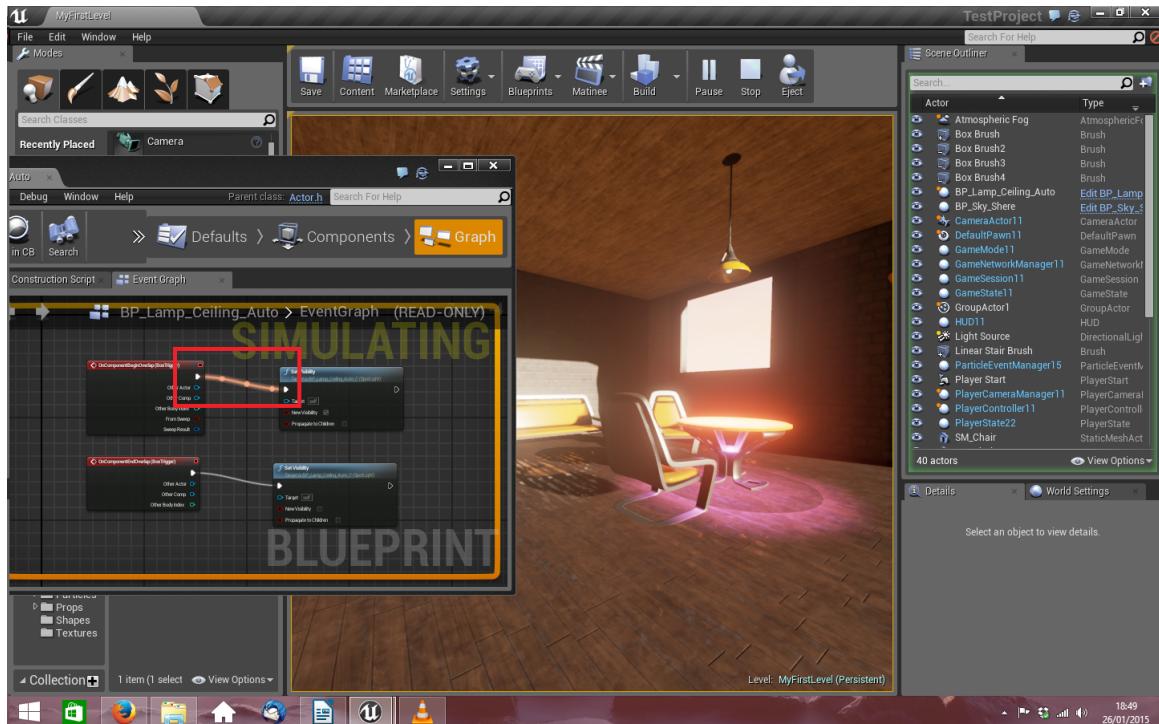


Illustration 40: Mode jeu avec simulation au niveau du graph

Dans l'exemple de l'illustration 41, nous avons 3 actors instanciés à partir de cette classe. Il suffit de sélectionner l'instance que l'on souhaite observer pour observer les résultats dans l'«EventGraph».

Au passage, remarquez la **présence du bouton «Eject»** disponible uniquement quand le jeu est lancé. Il permet de changer de contrôleur de jeu (une notion que nous verrons juste après) et de revenir au contrôleur standard. A cette étape du projet, cela ne change rien, mais lorsqu'on utilise un personnage comme contrôleur, cela permet de revenir en «Fly Mode» et de tourner autour du personnage. Le bouton «Eject» est alors remplacé par le **bouton «Possess»** permettant de revenir au commandant du contrôleur précédent.

Cette approche visuelle du débogage est très intéressante car quand le nombre de nœuds devient important et que le programme ne semble pas faire ce qu'on attend de lui, cela permet d'observer en temps-réel les différentes activations, et de trouver où le problème commence.

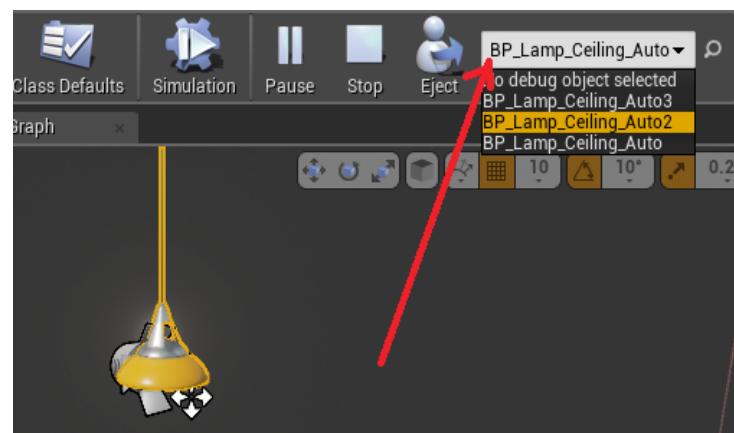


Illustration 41: Sélection de l'instance à visualiser

Les breakpoints

Les programmeurs sont habitués à la notion de «point d'arrêt» ou «breakpoint». C'est un petit marqueur que l'on dépose devant une instruction et qui permet à l'éditeur de mettre le jeu en pause dès que ce marqueur est atteint. La programmation Blueprint dispose de ce genre d'outil.

Pour l'utiliser:

3. Allez sur l'EventGraph de «BP_Lamp_Ceiling_Auto»
4. ⚡ sur le premier «Set Visibility» et choisissez «Add breakpoint»

Un petit rond rouge apparaît en haut à gauche du nœud «Set Visibility».

5. Lancez le jeu et approchez de la zone qui va activer la lampe.

Automatiquement, le jeu est mis en une sorte de «pause». Examinons l'illustration 42 :

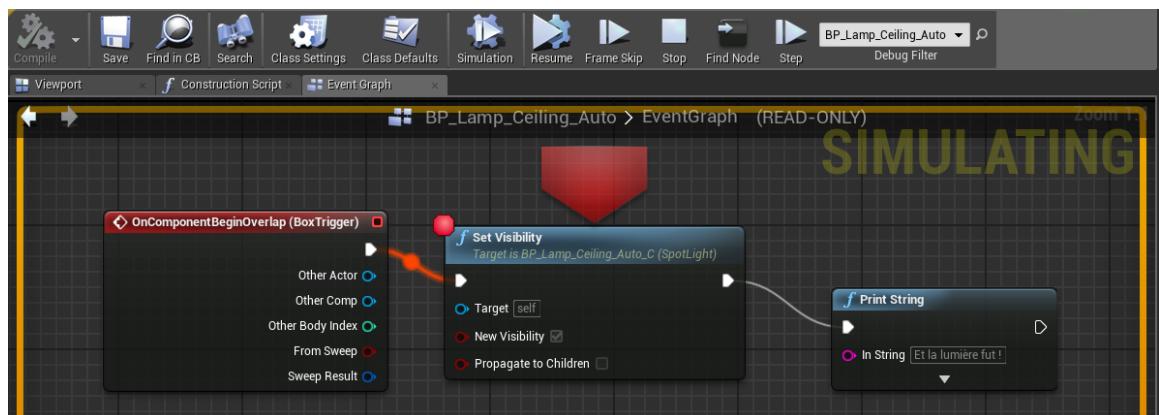


Illustration 42: Ajout d'un breakpoint et lancement du débogage

Ce n'est pas une pause comme lorsqu'on appuie sur le bouton «pause», mais cela revient au même. L'affichage du jeu est gelé, et on ne peut qu'observer le programme et la valeur des différentes entrées et sorties de chaque nœud. Par exemple, survolez la sortie «Other Actor» du nœud «OnComponentBeginOverlap» pour obtenir sa valeur sous la forme d'une **bulle d'aide**.

Pour observer une ou plusieurs variables en temps-réel, déposez cette variable sur le graphe et ⚡ sur cette variable en sélectionnant «Watch This Value». Une petite bulle contenant la valeur de la variable apparaît juste au-dessus du nœud et restera présente tant qu'on aura pas fait un «Stop watching this value».

Le nœud actif, c'est à dire l'instruction sur laquelle le jeu s'est arrêté avant même de l'exécuter, est symbolisé par une **flèche rouge** dirigée vers le bas.

A partir d'ici, on peut reprendre le cours du jeu en cliquant sur «Resume», avancer d'une seule frame en cliquant sur «Frame Skip» (ce qui ne change rien à notre débogage actuel) ou cliquer sur «Step» pour avancer vers l'instruction suivante. Dans ce cas, c'est comme si nous avions placé un point d'arrêt sur le nœud suivant «Print String». La flèche symbolisant le nœud actif devient grise mais se déplace bien vers le nœud suivant.

Quand le graphe est très grand, le bouton «Find Node» permet de centrer la vue sur le nœud actif en cours de débogage.

Pour retirer un point d'arrêt, il suffit de ⚡ sur le nœud en question et de choisir «Remove Breakpoint».

«Toggle breakpoint» ou ⌘ permet de désactiver le point d'arrêt s'il y en a un ou d'en poser un s'il n'y en a pas.

La fenêtre de débogage

Dans le menu «Window», «Developper Tools», vous trouvez «Blueprint Debugger» (illustration 43):

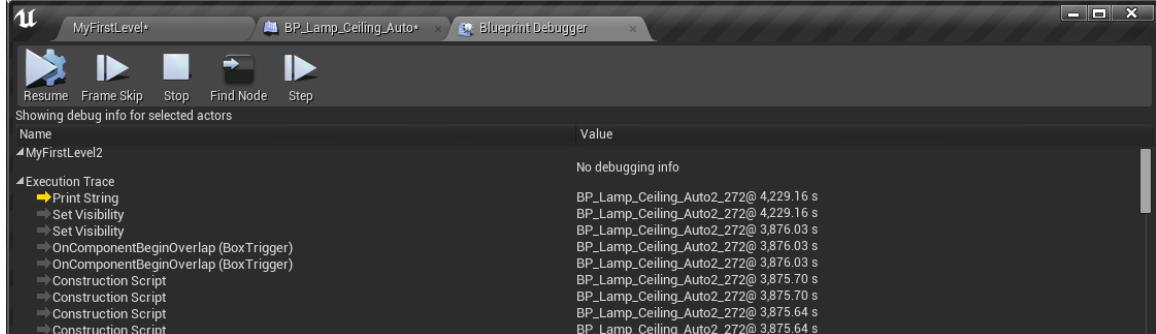


Illustration 43: Fenêtre "Blueprint Debugger"

Comme vous pouvez le constater, cette fenêtre retrace l'exécution de chaque instruction dans le temps.

A l'ancienne...

Une **autre façon de déboguer** peut se faire en affichant des messages à l'écran. Pour cela, modifiez le graphe comme sur l'illustration 44.

Procédez comme précédemment pour créer les liens.

Comprenez que le lien qui relie les «►» signifie qu'on déclenche une action. Ici, lorsque l'événement «OnComponentBeginOverlap» est activé, il déclenche une première action («Set Visibility»), puis une fois l'action terminée, une nouvelle action «Print String» qui permet d'afficher un message à l'écran. Repérez bien cette succession car un graphe peut contenir un nombre très élevé de liens.

Nous aurions pu, par exemple, afficher le contenu d'une variable en reliant la borne «In String» à la valeur de cette variable. Le lien aurait pris la couleur «rose» comme la borne (ce qui signifie «chaîne de caractère»: la couleur dépend du type de la borne).

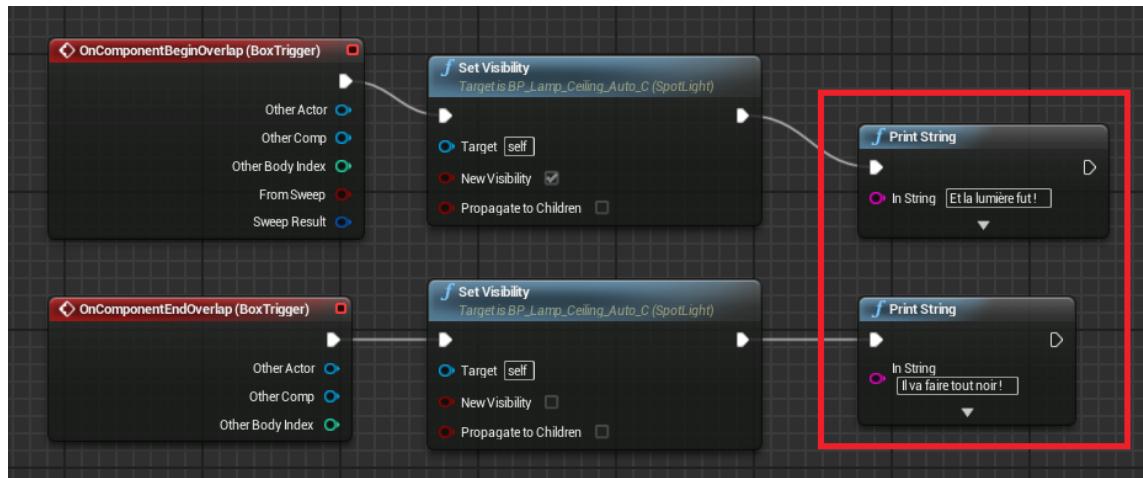


Illustration 44: Affichage d'informations de débogage dans la fenêtre de rendu grâce à la commande "Print String"

CRÉATION D'UN BOUTON D'ALLUMAGE DE LAMPES

Ajoutons une première interaction avec des objets: pouvoir cliquer sur un bouton pour allumer une lampe :

1. Ajoutez 4 points lumineux de type «**Point Light**» dans la pièce et répartissez-les le long des murs. Renommez «Lamp1», «Lamp2», «Lamp3» et «Lamp4».
2. Ajoutez l'asset «Shape_Cube» du répertoire «Shapes» de «StarterContent». Renommez-le en «LightButton» et
3. Redimensionnez en «X:0.2», «Y:0.2» et «Z:0.1».
4. Placez-le le long d'un mur, il symbolise le bouton pour allumer et éteindre les lumières.

Les différents éléments graphiques sont prêts, mais avant de passer à la programmation Blueprint, nous devons créer un nouveau **PlayerController**: ceci va nous permettre d'afficher le curseur souris et d'activer la gestion des événements comme le clic sur un objet:

5. Dans **CB**, sélectionnez le répertoire «Blueprints» créé précédemment et cliquez sur «Create», puis sur «Blueprint», et enfin sur «PlayerController». Renommez le fichier en «BP_MyPlayerControl»
6. Ouvrez le Blueprint et allez dans le volet «Default».
7. Dans le panel «Mouse Interface», cochez «Show Mouse Cursor» et «Enable Click Events».
8. Sélectionnez «Hand» dans «Default Mouse Cursor».

Un PlayerController est l'interface permettant de contrôler un avatar («pawn»). Si nous arrivons à nous déplacer dans l'espace, c'est parce que, par défaut, on nous a assigné le contrôle d'une caméra. Mais nous pourrions tout aussi bien déplacer un personnage ou un véhicule. Pour l'instant, nous allons continuer à bénéficier de cette petite caméra mobile dont nous avons affiné le fonctionnement pour traiter les clics de souris.

Pour utiliser ce PlayerController, créons un nouveau «**GameMode**»:

9. Dans **CB**, sélectionnez ce même répertoire «Blueprints» et créez un nouveau «Blueprint» de type «GameMode». Renommez le fichier en «BP_MyGameMode»
10. Ouvrez le Blueprint et allez dans le volet «Default». Dans le panel «Classes», modifiez «Player Controller Class» en «BP_MyPlayerControl» afin d'utiliser le nouveau contrôleur.
11. Menu «Windows» → «World Settings», panel «Game Mode», «Gamemode Override»: sélectionnez «BP_MyGameMode».

Si vous testez maintenant, la seule différence que vous observerez est que le pointeur de la souris est actif et qu'il se transforme en «main» dès que l'on utilise l'un des boutons de la souris. Maintenant, nous allons **programmer l'interaction**:

12. Dans **WO**, sélectionnez l'interrupteur «LightButton», puis dans la barre d'outils principale, cliquez sur «Blueprints» et sélectionnez «Open Level Blueprint».
13. ☰ sur le graph et ajoutez «**Add On Clicked**»: c'est l'événement qui est appelé quand on clique sur l'actor «LightButton».
14. A partir de la sortie exec du nœud, créez une action «**Toggle Visibility**» en décochant «Context Sensitive» lors de votre recherche.

Lorsqu'elle est cochée, seuls les éléments liés au contexte (ici, nous sommes dans un Blueprint de type level, pas dans le Blueprint d'un véhicule par exemple) sont proposés. Le contexte est aussi défini par le type de données à partir duquel on crée le nœud. Quand vous ne trouvez pas l'élément proposé, décochez temporairement la case pour trouver ce que vous recherchez.

15. Dans **WO** de l'éditeur de niveau, sélectionnez les actors «Lamp1», «Lamp 2», «Lamp3» et «Lamp4» (maintenez **Ctrl** enfoncee lors du clic pour sélectionner plusieurs actors), puis revenez sur le Blueprint du niveau.
16. ☰ sur le graphe et ajoutez «Create reference to 4 selected Actors». 4 nouveaux nœuds apparaissent et représentent nos actors.
17. Prennez la sortie «Point Light Reference» du nœud «Lamp1» et reliez-la à l'entrée «Target» du nœud «Toggle Visibility».

Vous avez remarqué que le nom des entrées et sortie apparaissent sous la forme d'une bulle d'aide lorsque la souris les survole. Un nouveau nœud intermédiaire se crée. L'avantage d'utiliser «Toggle Visibility» c'est que si l'objet est visible, il devient invisible et s'il ne l'est pas, il redevient visible. Ainsi, nous ne sommes pas obligé de conserver une variable booléenne pour stocker l'état des lampes, ou interroger leur état.

- 18.Répétez l'opération pour les 3 autres lampes.
- 19.Faites un peu le ménage (illustration 45).

Vous pouvez déplacer facilement les nœuds en cliquant sur le nom du nœud, en maintenant le clic et en déplaçant la souris. La molette de milieu permet de gérer le niveau de zoom et vous pouvez déplacer tout le graphe en utilisant , en maintenant le clic et en déplaçant la souris. Vous pouvez sélectionner plusieurs nœuds en cliquant sur le coin supérieur gauche de la sélection et en agrandissant jusqu'au coin inférieur droit. Vous pouvez aussi les sélectionner individuellement en utilisant  pour faire une sélection multiple. Lorsque plusieurs nœuds sont sélectionnés, déplacer l'un deux permet de déplacer l'ensemble.

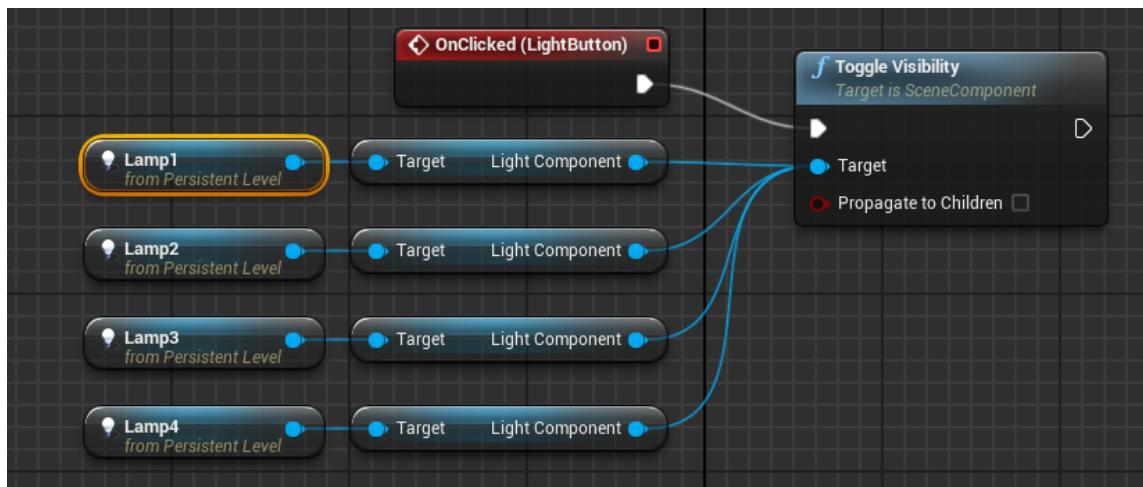


Illustration 45: Blueprint permettant d'allumer ou éteindre 4 lampes

- 20.Compilez et sauvez votre Blueprint, effectuez un «Build» pour prendre en compte les nouveaux points lumineux et testez votre jeu.

Si tout se passe bien, quand vous appuyez sur l'interrupteur une première fois, toutes les lampes s'éteignent. Si vous répétez l'opération, elles se rallument toutes.

- 21.Faites une copie de l'actor «LightButton» et positionnez-le à un autre endroit de la pièce pour faire un va-et-vient.
- 22.Testez de nouveau.

Cela ne fonctionne pas. Pourquoi? Si nous avions réalisé un composant et que le Blueprint était lié au composant même, cela aurait fonctionné (il aurait fallu partager les lampes toutefois). Mais là, notre événement «OnClicked» est bien lié à «LightButton» et pas à toutes ses copies! Pour remédier à cela, vous devrez sélectionner «LightButton2», puis dans le graphe, ajouter un événement «OnClicked» lié à ce nouvel acteur et relier sa sortie exec à l'entrée exec du «Toggle Visibility» actuel. Ainsi, cela fonctionnera pour les 2 boutons.

- 23.Sélectionnez tous les éléments de votre graphe et appuyez sur : nommez «Toggle Lights with Interruptors».

Ainsi, quand vous aurez plusieurs éléments sur votre graphe, vous pourrez plus facilement vous y retrouver.

Dans cet exemple, nous n'avons pas créé de composant particulier. Il s'agissait juste de programmer la logique d'un niveau, alors que dans le premier cas, nous avons créé un composant lampe réutilisable qu'on pourra importer dans un autre projet. Ce sont deux approches différentes, tout dépend de notre besoin.

PREMIÈRE ANIMATION AVEC MATINEE

Dans un premier temps, refermons le bâtiment et créons une porte:

1. Sélectionnez l'objet «Box Brush 2» et redimensionnez Y à 750. Déplacez l'objet selon l'axe des X vers le fond du bâtiment pour faire apparaître le mur de devant. Appliquez les matériaux désirés.
2. Utilisez une nouvelle brush de type «Box» pour réaliser une ouverture dans le mur de la taille d'une porte.

Nous allons créer un composant «porte» doté de son propre système de détection du joueur:

3. Dans **CB**, répertoire «Props» sélectionnez «**SM_Door**» et → «Asset Actions» → «Create Blueprint using This». Choisissez le répertoire «Blueprints» créé préalablement, puis renommez en «**BP_Door**»
4. Sélectionnez votre nouveau Blueprint dans **CB** et ajoutez-le à la scène. Déplacez et redimensionnez pour que la porte viennent combler la nouvelle ouverture créée, poignée vers la gauche de l'extérieur.
5. Ouvrez le Blueprint «**BP_Door**» en édition, allez dans le volet «Components».
6. Dans la fenêtre «Components», sélectionnez la porte. Cliquez sur «Add Component» → «**Box**». Renommez la «Box» en «**Box_interior**»

Le but est d'afficher un message quand on approche de la zone, nous allons donc créer une boîte de collision que nous allons placer et dimensionner comme sur l'illustration 46 (remarquez le sens de la poignée, nous sommes à l'intérieur).

7. Sélectionnez «**Box_Interior**», puis cliquez sur «Add Component» → «Text Render». Renommez en «**ExitText**». Dans le panel «Text», indiquez «Text»: «Press 'space' to Exit».

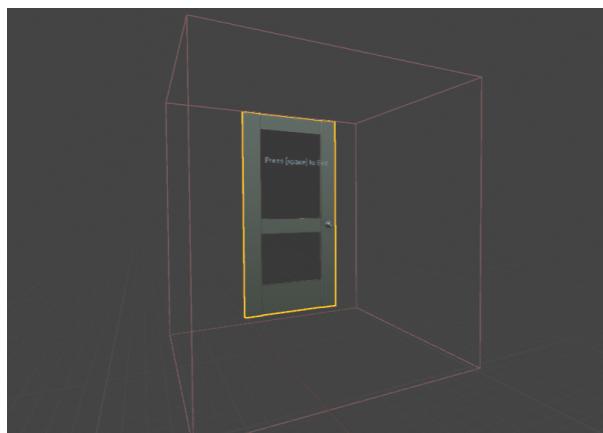


Illustration 46: Composant porte: un StaticMesh, un Trigger Box et un Text

Maintenant, créons l'animation de la porte sous Matinee»:

Revenez sous l'éditeur de niveau et ajoutez un nouvel actor matinee:

8. Dans la barre d'outils, cliquez sur «Matinee» → «Add Matinee».

Une nouvelle fenêtre s'ouvre comme sur l'illustration 47. Nous reviendrons sur cette fenêtre après avoir réalisé l'animation.

9. Dans la zone 3 «Tracks», dans la zone encadrée en vert, et sélectionnez «Add New Empty Group», nommez «**OpenDoorGroup**».
10. Revenez dans l'éditeur de niveau et sélectionnez «**BP_Door**». Revenez ensuite dans Matinee et sur «**OpenDoorGroup**»: Sélectionnez «Actors», puis «Add Selected Actors».

11. ⌂ sur «OpenDoorGroup»: Sélectionnez «Add New Movement Track».

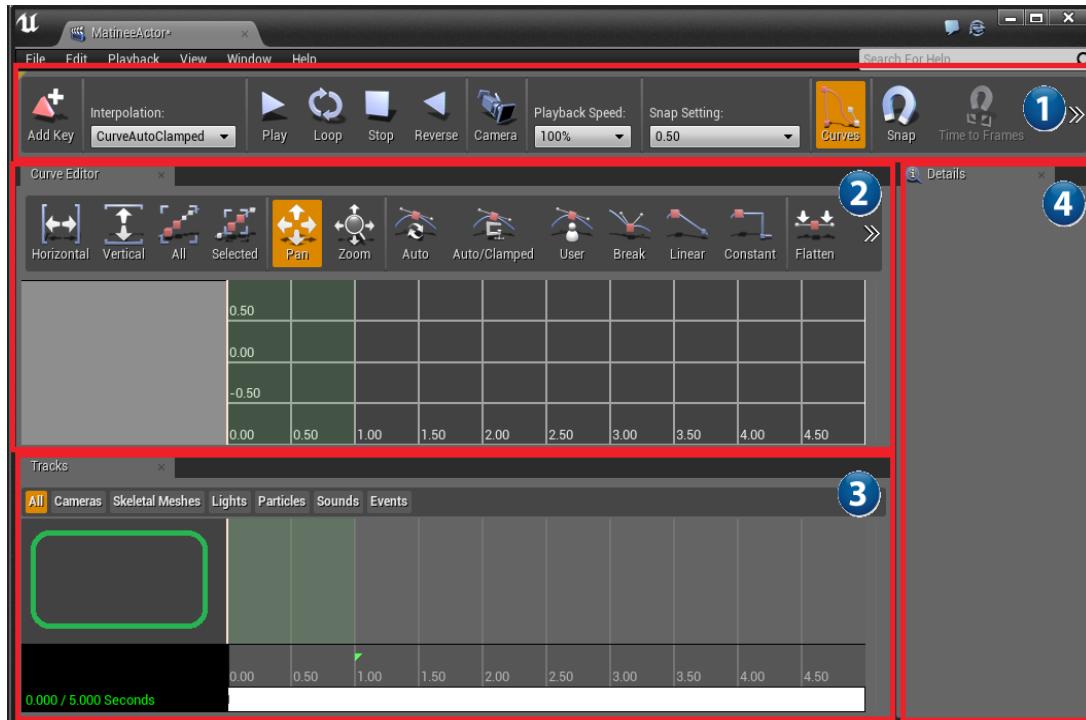


Illustration 47: Fenêtre d'édition Matinee

Dans la timeline, c'est à dire le graphique juste à droite de «Tracks» (zone 3), vous avez des petites flèches vertes et rouges.

12. Comme dans l'illustration 48, cliquez sur la petite flèche verte de droite (celle dans le cercle gris) et en maintenant le clic déplacez la de 1s à 2s.

La molette de la souris permet de gérer le zoom sur le graphe. Cliquez sur le graphe et tout en maintenant le bouton, faites un mouvement «droite-gauche» et vous déplacez la partie visible de la timeline.

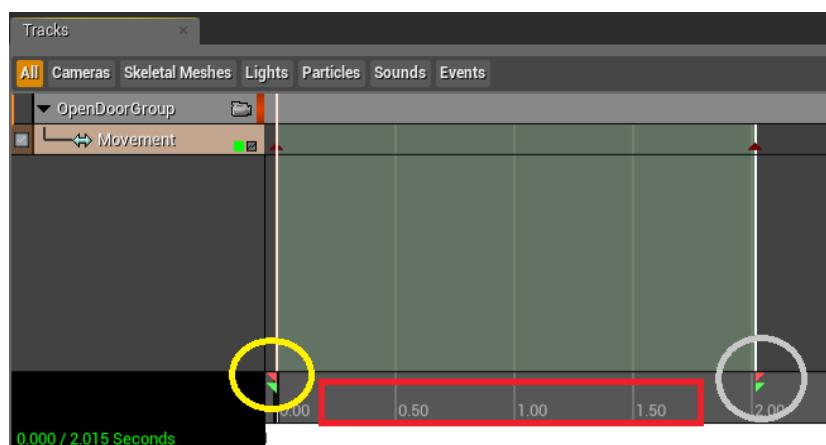


Illustration 48: Volet "Tracks" de l'éditeur Matinee

13. De la même façon, attrapez la flèche rouge de droite (cercle gris encore) et déplacez la de 5s vers 2s.
14. Placez le curseur de la timeline en 2s. Pour cela, il suffit de cliquer dans la zone encadrée en rouge.
15. Puis cliquez sur «**Add Key**» de la barre d'outils (zone 1).
16. Revenez dans l'éditeur de niveau. Sélectionnez l'actor «BP_Door» et effectuez une rotation d'axe Z de -90°.
17. Revenez ensuite sur l'éditeur Matinee et cliquez sur «Loop» de la barre d'outils.

Nous lançons l'animation en boucle. Revenez sous l'éditeur de niveau et vous allez constater que la porte est animée comme sur l'illustration 49.



Illustration 49: Animation de la porte en cours sous "Matinee"

18. Revenez ensuite sur l'éditeur Matinee, cliquez sur «Stop», puis dans le menu «File» → «**Save All**». Un message vous indique que l'éditeur Matinee doit être fermé: acceptez en cliquant sur «Yes».
19. Vous retrouvez le nouvel actor «MatineeActor» dans le **WO** de l'éditeur de niveau. Renommez-le «MatineeDoor».

Vous pouvez aussi le déplacer au niveau de la porte et le parenter à «BP_Door» dans le **WO**: ce n'est pas obligatoire, mais c'est pour ranger un peu votre scène et retrouver rapidement les informations visuellement.

PROGRAMMATION BLUEPRINT DE L'ANIMATION

Inclure l'animation au composant n'est pas possible, un actor Matinee ne peut pas être directement intégré à un actor Blueprint. Plus simplement, un actor ne peut être intégré à un actor sous la forme d'un composant. Toutefois, depuis la version 4.7, tout élément du **CB** peut-être associé à un actor existant en l'ajoutant sous la forme d'un composant («Add component»). Ce n'est malheureusement pas le cas des animations réalisées sous Matinee.

La solution consiste, ici, à ajouter à notre Blueprint un paramètre permettant de référencer l'animation et la déclencher directement à partir du Blueprint.

Commençons déjà par la visibilité du texte:

1. Ouvrez «**BP_Door**» en édition et allez dans le volet «Graph». Dans la fenêtre «My Blueprint», cliquez sur «+ Variable» pour ajouter une nouvelle variable. Nommez-la «IsPlayerNear» (signifie «Est-ce que le joueur est proche?»). Dans «Details», «Variable Type» est «Boolean» (Vrai/faux).
2. A vous de jouer maintenant: **réalisez le graphe** de l'illustration 50.

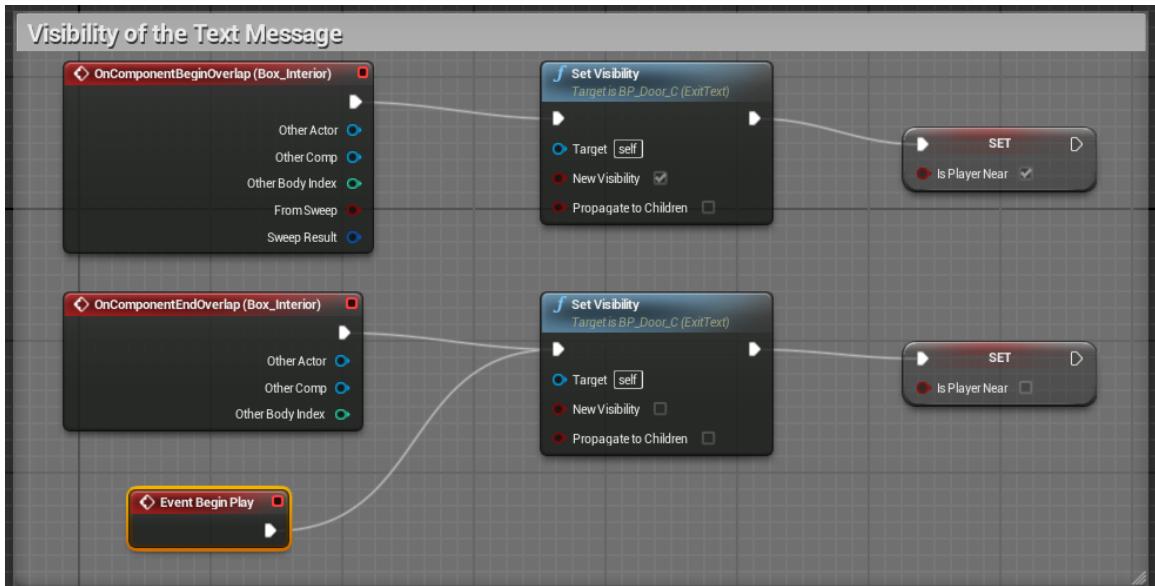


Illustration 50: Blueprint pour cacher ou afficher un actor en fonction d'un trigger

Vous connaissez déjà ce programme, la seule différence est la présence de la variable «**IsPlayerNear**». Pour l'obtenir sur le graphe, il suffit de la glisser-déplacer de la fenêtre «My Blueprints» vers le graphe. Il nous est demandé si on souhaite un «Get» (pour lire son contenu) ou un «Set» (pour modifier son contenu). Dans le cas de notre graphe, il s'agit de modifier la variable, donc un «Set». Cette variable est à «Vrai» quand le player est dans la zone «Box-Interior» et à «Faux» sinon.

Ensuite, nous avons besoin de **déclencher l'animation** si le joueur est proche et qu'il appuie sur «espace». Nous allons en profiter pour créer un «custom event», c'est à dire un événement personnalisé. Il faut le comprendre comme un ordre ou un message. On indique au composant «**BP_Door**» qu'il doit ouvrir sa porte.

Voici comment procéder:

3. Ajoutez une nouvelle variable booléenne «**IsDoorOpened**» qui nous permet de savoir si l'animation d'ouverture a commencé ou si la porte est ouverte.
4. Ajoutez une autre variable, de type «**Matinee Actor**», nommez-la «**Animation**». Notez le petit «œil» juste à droite du nom et cliquez dessus pour le rendre actif.

Cela revient à cocher la case «**editable**» dans la fenêtre «**Details**». Cela signifie que la variable pourra être visible et donc accessible en dehors du Blueprint. Ce sera la référence de notre actor Matinee. Vous vous souvenez: nous ne pouvons pas l'intégrer au composant, mais nous pouvons l'utiliser en référence. Vous allez comprendre avec la suite.

5. **Compilez** votre Blueprint pour que les nouvelles variables soit prises en compte.
6. Revenez dans l'éditeur de niveau, sélectionnez «**BP_Door**» et rendez-vous dans la fenêtre «**Details**». Dans le panel «**Default**», vous avez un nouveau champ «**Animation**». Sélectionnez «**MatineeDoor**».
7. Revenez dans l'éditeur de «**BP_Door**», au niveau du graphe, et réalisez le programme de l'illustration 51.

Comme précédemment, les variables des encadrés verts proviennent d'un glisser-déposer de la fenêtre «**My Blueprints**» des variables du même nom - il s'agit d'un «**Get**» cette fois-ci, nous voulons tester leur contenu. «**OpenTheDoor**» est un «**Custom Event**», un événement personnalisé. Pour le créer, il suffit de cliquer sur le graphe et d'entrer «**Add Custom Event**», puis de le renommer en «**OpenTheDoor**». «**Not**» (Boolean), «**And**» (Boolean) et «**Branch**» se trouve facilement en utilisant ces mots.

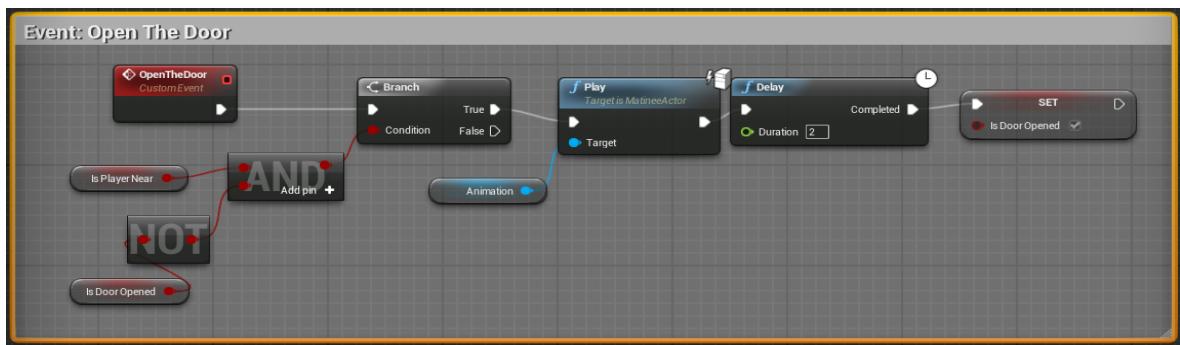


Illustration 51: Événement permettant d'ouvrir la porte

Comment lire et comprendre ce graphe ?

Le point d'entrée est l'événement «**OpenTheDoor**». Il est déclenché de façon externe, nous verrons comment par la suite.

Un test est réalisé: Si la porte n'est pas ouverte (**isDoorOpened** + NOT) et (AND) que le joueur est dans la zone (**IsPlayerNear**) alors (Branch) lancer l'animation (Play) et positionner **IsDoorOpened** à Vrai.

«**Branch**» est très important car vous l'utiliserez très souvent, c'est ce qui permet le test conditionnel. Il prend en entrée une variable booléenne et si cette dernière est «**Vrai**», il enclenche l'action sur la sortie «**True**» ou l'action sur la sortie «**False**» dans le cas contraire. A la place d'utiliser un Et logique («**AND**»), nous aurions pu utiliser 2 «**branch**» en cascade, mais cela aurait été moins optimisé et peut-être moins facile à lire.

«**Play**» lance l'animation matinee puisqu'en entrée elle reçoit la variable animation que nous avons précédemment positionnée à «**MatineeDoor**». C'est pour cela qu'on parle de référence. Ici, cela ne signifie pas que l'animation va être jouée et qu'ensuite **IsDoorOpened** sera positionné à «**Vrai**». En réalité, la fonction «**Play**» informe le système d'animation qu'il devra traiter cette animation, mais continue ensuite par sa sortie «**exec**» avant même que l'animation soit réellement jouée. Il s'agit d'un traitement qui s'effectue en parallèle: ce n'est pas ainsi que fonctionne les nœuds en général, d'où cette précision. Pour éviter que «**IsDoorOpened**» soit mis à vrai durant toute l'animation, on ajoute un temps d'attente de 2 secondes grâce à la fonction **delay**.

Maintenant, nous allons gérer l'appui sur la touche «**espace**»:

8. Retournez dans l'éditeur de niveau et ouvrez le Blueprint du level. C'est à ce niveau que nous allons traiter l'entrée clavier «**espace**».
9. Ajoutez le graphe de l'illustration 52.
10. Vous pouvez **compiler, sauver et tester** votre jeu.

Rappelez vous que pour obtenir une référence à «BP_Door», il suffit de sélectionner l'actor dans le **WO** de l'éditeur de niveau et de faire un sur le graphe et de sélectionner «Create a reference to BP_Door» (notez que cela n'aurait pas fonctionné pour l'actor Matinee dans le Blueprint précédent car ce n'est pas ainsi qu'on référence un actor matinee dans un autre actor). Toutefois, «Open the door» est naturellement attaché au Blueprint «BP_Door» donc si vous ajoutez le nœud en partant de «Space Bar», il crée automatiquement une référence en entrée à «BP_Door». Tout ceci pour vous montrer qu'il y a plusieurs façons de «dessiner» le graphe.

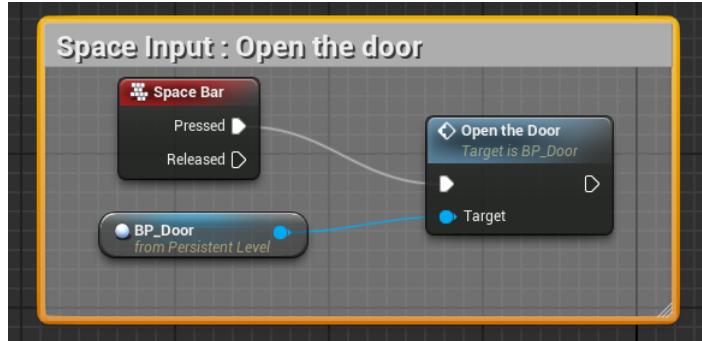


Illustration 52: Gestion de la touche "Espace"

Concrètement, lorsque la touche «espace» est pressée, on déclenche l'événement personnalisé «Open the door».

En vous approchant de la porte, le message apparaît, si vous reculez, il disparaît. Pendant que le message est visible, vous pouvez appuyer sur «espace» et la porte va s'ouvrir... mais uniquement de l'intérieur!

Nous aurions pu étendre la surface de collision et afficher le message des deux cotés de la porte. Mais la méthode que nous utilisons nous permet de savoir de quel côté nous sommes et d'afficher un message personnalisé.

FERMETURE AUTOMATIQUE DE LA PORTE

Poursuivons avec la fermeture de la porte et étudions une autre façon de procéder. Nous allons mesurer la distance entre le joueur et la porte plutôt que d'utiliser une boîte de collision.

1. Modifiez le graphe conformément à l'illustration 53.

L'événement «Event Tick» est appelé à chaque affichage de frame. A chaque frame, on teste donc la distance entre le player et «self», c'est à dire l'actor «BP_Door». Si la distance est inférieure à 350, alors le message est affiché, sinon, il est caché.

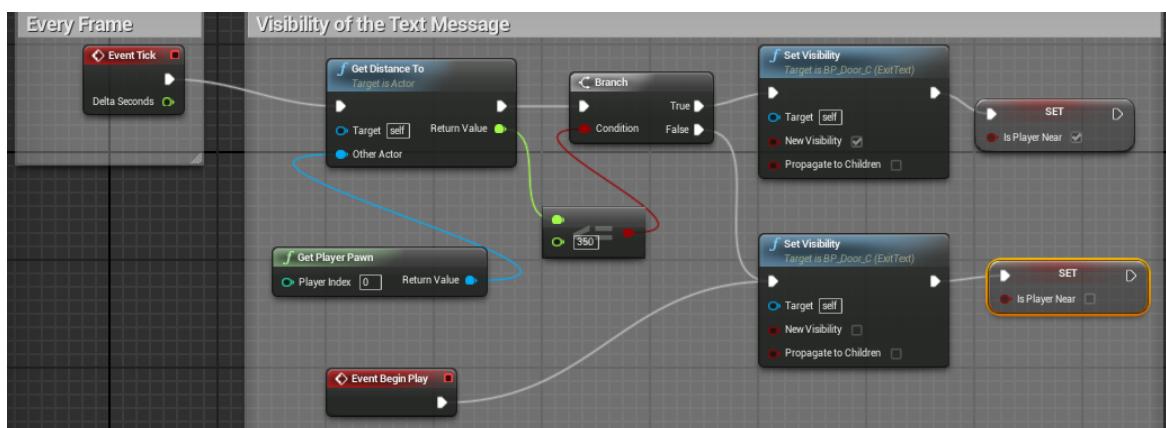


Illustration 53: Modification du graphe BP_Door

Ensuite, nous profitons du dernier nœud en surbrillance: à ce niveau, le joueur est une distance supérieure à 350. Il nous reste à tester si la porte est ouverte et si oui, à déclencher l'animation inverse.

2. Ajoutez à partir de ce nœud le graphe de l'illustration 54.
3. Compilez, sauvez et testez.

Prenez note de la façon de lire en sens inverse une animation: on utilise en complément la fonction «Change Playback Direction».

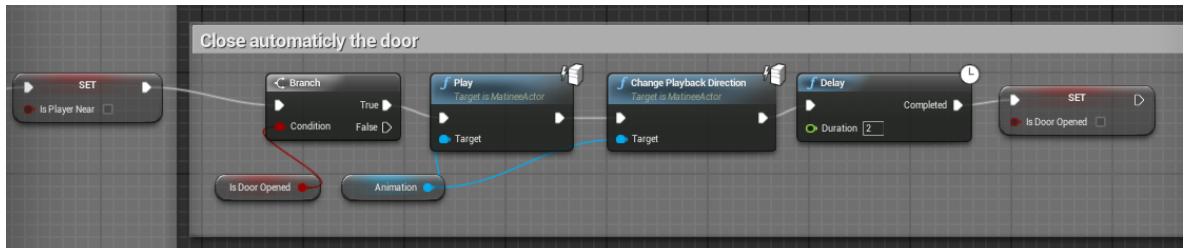


Illustration 54: Fermeture automatique de la porte

Avancez vers la porte, ouvrez-la. Elle reste ouverte tant que vous restez à proximité. Reculez et la porte se referme automatiquement. L'avantage, c'est que cela fonctionne aussi de l'extérieur.

LES BASES DE LA PROGRAMMATION BLUEPRINT

En ayant pratiqué la programmation visuelle au travers de ces quelques exemples, nous espérons que nous avons réussi à éveiller votre intérêt concernant cette approche.

Cette introduction ne serait pas complète si nous ne définissions pas d'avantage la notion de Blueprint, et si nous ne vous donnions pas les éléments de base du langage. Car il s'agit bien d'un langage: il y a des variables, des fonctions, des «mots-clés» en quelque sorte, des boucles, une bibliothèque standard (API)... L'utilisation d'un graphe composé de nœuds permet d'éviter les erreurs de syntaxe, et de rendre plus intuitive la programmation. Mais cela reste de la programmation!

Qu'est-ce qu'un Blueprint?

Nous aurions pu vous donner ces définitions au début du chapitre, mais le fait d'avoir pratiqué avant permet de mettre des images derrière des concepts qui sont parfois complexes et difficile à définir.

Tout d'abord, vous remarquez que nous utilisons le terme «Blueprint» pour définir plusieurs choses:

- La **«programmation Blueprint»**, c'est une programmation visuelle, par opposition à la programmation C++ qui utilise un langage de programmation. Le terme *blueprint*¹ désigne, en anglais, un plan détaillé. Cela représente assez bien cette idée de graphe contenant des nœuds, associant des fonctions et des variables entre elles, ainsi que des éléments de langage.
- Une **«classe Blueprint»**: en programmation orientée objet (POO), une classe est un type d'objet. Elle sert de modèle à une instantiation. Ainsi, la classe BP_Lamp peut être instanciée 3 fois, nous obtenons **3 actors différents**, mais qui héritent tous des propriétés et des fonctions de la classe BP_Lamp. Cette dernière peut également avoir hérité des propriétés d'une autre classe parente: on appelle cela l'héritage.
- Un **«level Blueprint»**: il s'agit de l'**ensemble des programmes associés à un niveau**. Certains programmes sont liés à un objet, les éléments étant contenus dans la «classe blueprint» de cet objet. Ce sont en quelques sortes des programmes réutilisables car ils peuvent servir à plusieurs niveaux, voir à d'autres projets. D'autres sont liés au level: il n'y a que dans le Blueprint du level que l'on peut directement accéder aux actors référencés dans **WO**.

¹ Le terme, signifiant littéralement «impression en bleu», provient d'un procédé d'imprimerie, la *cyanotypie*. Mais, la ressemblance s'arrête à la notion de plan détaillé.

- Une «**interface Blueprint**»: c'est une **collection d'une ou plusieurs fonctions** – on définit un nom, mais sans implémentation. Cela signifie que le programme n'est pas défini à ce niveau. C'est au moment de l'implémentation que le programme est ajouté. L'utilité, comme en programmation classique, est de pouvoir accéder à plusieurs objets différents au travers d'une interface commune. En gros, les interfaces Blueprint permettent à différentes classes Blueprint de communiquer entre elles et d'échanger des données, et ce, sans avoir à passer par un mécanisme d'échange de messages de gestion d'événements.
- Une «**macro librairie Blueprint**» contient une collection de macros ou des graphes autonomes qui peuvent être placés dans d'autres Blueprints. Les macros sont des «morceaux» de graphe, des assemblages de nœuds. **Alors qu'une fonction possède un point d'entrée et un point de sortie, une macro peut en contenir plusieurs.** Il est possible d'y inclure des fonctions de latence comme le delay, alors que c'est impossible dans une fonction. La fonction est compilée une seule fois et peut-être appelée à plusieurs endroits, une macro est compilée dans chaque Blueprint qui y fait appel. En gros, c'est comme si nous avions fait un copier-coller dans chaque blueprint, ce qui fait grossir l'ensemble du code au final. A l'inverse, les macros contenues dans la librairie qui ne sont pas utilisées dans le projet ne seront pas compilées.
- Un «**Blueprint Utility**» ou «Blutility» est un blueprint particulier qui contient un programme spécifique à l'éditeur et dont l'objet est d'étendre ses fonctionnalités.

Comme nous l'avons vu précédemment, une «**classe Blueprint**» contient:

- Un ou plusieurs **composants** (fenêtre «Components»)
- Des **variables** (ce sont les propriétés de l'objet en C++)
- Des **fonctions** (ce sont les méthodes de l'objet en C++)
- Des **macros** (des extraits de graphes)
- Un **graphe de construction** «Construction Script» permettant de réaliser des opérations d'initialisation ou toute autre opération au moment de l'instanciation de la classe. C'est l'équivalent du constructor en C++.
- Un **graphe d'événements** «Event Graph» permettant de programmer le «comportement» de l'objet.

Un «**Level Blueprint**» contient les même éléments qu'une classe, mise à part les composants. Toutefois, il peut accéder directement aux **actors** contrairement aux classes.

Types de données et variables

Une variable est avant tout un moyen de stocker une information. Selon l'information que l'on souhaite traiter, on définit un type de variable.

Les types standards sont les suivants:

Type	Couleur	Exemple	Utilisation
Boolean	Rouge		Vrai/Faux
Byte	Vert pin		Entier non signé entre 0 et 255
Integer	Cyan		Entier signé entre -2147483648 et 2147483647 Ex: 1024

Type	Couleur	Exemple	Utilisation
Float	Verte		Flottants signés (nombres décimaux), de précision 6 de -45646545686719395e10 à 45646545686719395e10 Ex: -15.056667
Name	Mauve Parme		Chaîne de caractères (codage sur 8 octets) Ex: «Je joue avec Unreal Engine!» Utilisé principalement pour les données que l'on souhaite enregistrer sur le disque ou envoyer sur le réseau (plus petits que String).
String	Magenta		Chaîne de caractères (codage sur 16 octets). Possède des fonctions de modification, recherche, remplacement, etc.
Text	Rose		Chaîne de caractères particulière utilisée principalement pour la localisation/internationalisation. Cela permet d'accéder à des jeux de caractères spécifiques à un pays par exemple (codage sur 40 octets)
Vector	Or		Un vecteur contient 3 flottants. Utile pour manipuler des positions (X,Y,Z) ou une couleur (R,G,B). Ex: (0.5,-1.7,6.0)
Rotator	Violet		Comme les vecteurs, mais pour les rotations (Roll, Pitch, Yaw). Ex: (90.0, 180.0, 45.0)
Transform	Orange		Type de donnée contenant la notion de translation, de rotation et de redimensionnement.
Object	Bleu		Objets, incluant les Lights, Actors, StaticMeshes, Cameras, etc.

Il y a aussi des types particuliers comme les «enums» (énumérateur – exemple: «sapin, chêne, platane, pyracantha») que nous utiliserons dans le tome 2.

Une **structure** («Struct») est un assemblage de plusieurs types de variables. Le type «Transform» est une structure contenant 3 vecteurs. Pour exploiter les structures, il faut leur appliquer un break «type».

Exemple: si le type est «Hit Result», on applique un nœud «Break Hit Result» pour obtenir la liste de ses éléments distribués sur chaque sortie du nœud.

Avant d'utiliser un «struct» ou un «enum», il faut d'abord le créer en passant par «Add New» de **[CB]**, puis «Blueprints», en enfin, en choisissant «Enumeration» ou «Structure».

Les variables dont la propriété «editable» est cochée sont accessibles à partir de l'éditeur quand on affiche les détails du Blueprint.

La fonction «SET» permet d'entrer une valeur dans une variable et la fonction «GET» renvoie la valeur de cette dernière.

La **création d'une variable** se fait sous l'éditeur du Blueprint dans la fenêtre «My Blueprint», bouton «+» à coté de variable (à gauche de l'illustration 55). Une autre façon est d'utiliser «Promote to Variable» (↗) sur la sortie d'un nœud: il y a alors création automatique de la variable avec le type déjà sélectionné (à droite de l'illustration 55).

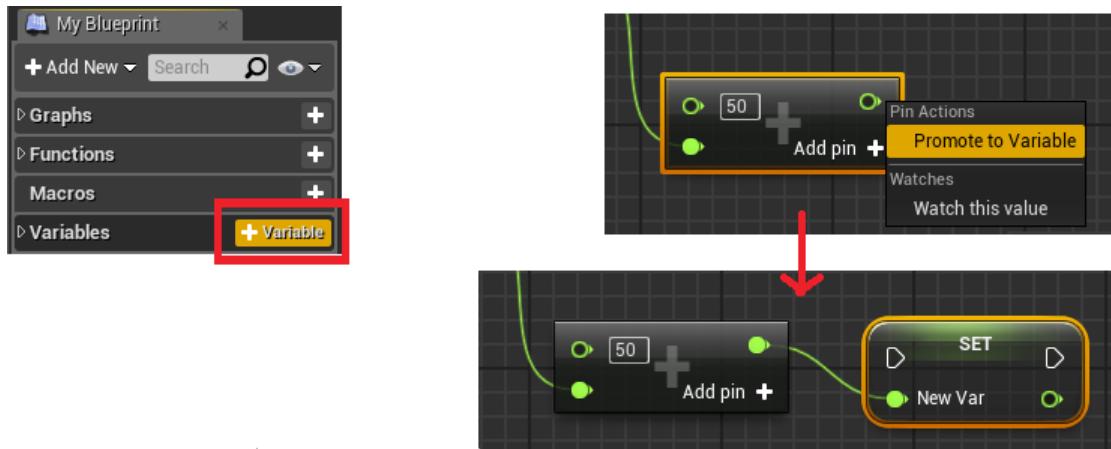


Illustration 55: 2 façons de créer une variable

La plupart du temps, l'éditeur permet de transformer un type en un autre type selon les besoins. Si par exemple on tente d'appliquer un «Print String» sur une variable de type «Float», une conversion «Float vers String» sera automatiquement ajoutée.

Tableaux et boucles

Un autre type de variable est le **tableau**. Lors de la création d'une variable, qu'il s'agisse d'un entier, d'une chaîne ou même d'une structure, il est possible de cocher l'icône «tableau» (encadré rouge, illustration 56).

Dans l'exemple ci-contre, la variable Scores est un tableau de flottants. Exemple: (0.5, -6, 67, 2.3, -102, ...).

Dans un tableau, tous les éléments sont de même type.

Les tableaux sont **dynamiques**: cela signifie qu'ils s'étendent au fur et à mesure des besoins. Il n'est donc pas nécessaire de réservé un certain nombre d'éléments avant de pouvoir l'utiliser. Le premier élément d'un tableau est d'index 0, le suivant 1, etc.

Une fois le tableau créé, il est possible de lui ajouter des éléments en passant par la fenêtre «Default Value». Le bouton «+» (encadré jaune, illustration 57) permet d'ajouter des éléments. On peut ajouter («insert») un élément entre deux valeurs, supprimer («delete») un élément ou copier («duplicate») un élément en utilisant le petit bouton de l'encadré rouge.

Pour travailler avec les tableaux, nous disposons de plusieurs fonctions:

- **ForEachLoop** (illustration 58): permet de créer une boucle et de traiter chaque élément un à un. On entre dans la fonction par «Exec», la sortie «Loop Body» correspond aux actions à réaliser sur chaque élément, ce dernier pouvant être récupéré par la sortie «Array Element». Si on a besoin de l'index de l'élément dans le tableau, on peut le récupérer grâce à la sortie «Array Index». Enfin, quand tous les éléments sont traités, le programme continue au travers de la sortie «Completed».

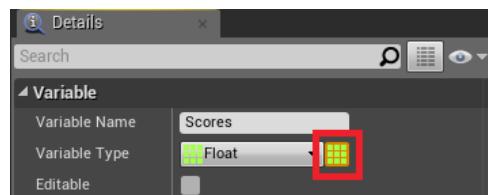


Illustration 56: Tableau de flottants

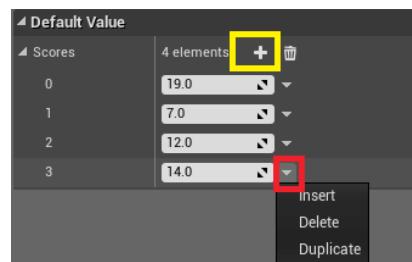


Illustration 57: Ajout d'éléments dans un tableau

- **ForEachLoopWithBreak:** ajoute à «ForEachLoop» une entrée «break» permettant d'arrêter le traitement à tout moment. Par exemple, pour trouver un élément répondant à une condition, si cet élément est au milieu du tableau, il n'y a pas besoin de continuer jusqu'à la fin. On relie la sortie du traitement à l'entrée «Break» pour que la boucle prenne fin et que le traitement continue via la sortie «Completed» comme si la fin du tableau avait été atteinte.

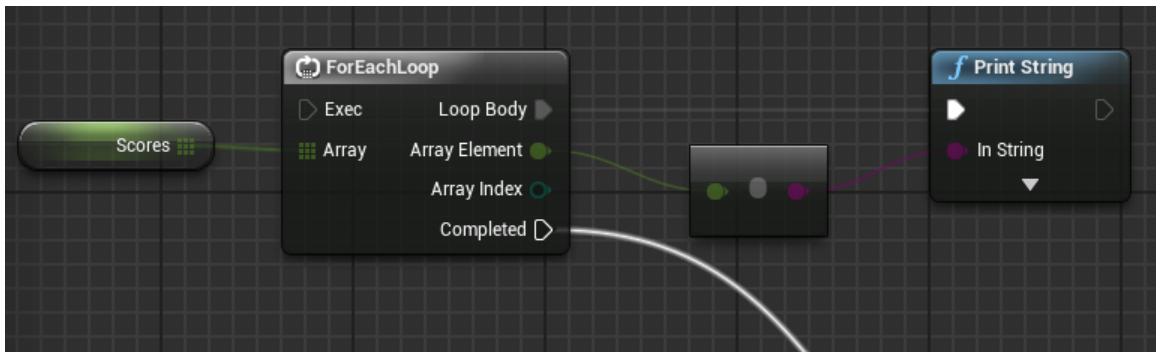


Illustration 58: Affichage de tous les éléments d'un tableau

« Utilities → Array» contient de nombreuses autres fonctions:

- **Add Item:** ajoute un élément à la fin du tableau
- **Clear:** supprime tous les éléments du tableau
- **Contains Item:** recherche dans le tableau une valeur et renvoie vrai si la valeur est trouvée, faux dans les cas contraire
- **Filter Array:** prend en entrée un tableau d'actors et filtre selon un type pour renvoyer un tableau d'actors ne contenant que les éléments de ce type.
- **Find Item:** recherche dans le tableau une valeur et renvoie sa position (index) dans le tableau
- **Get:** prend en entrée un numéro d'index et renvoie l'élément correspondant du tableau
- **Insert Item:** ajoute un élément à une position déterminée dans le tableau
- **Last Index:** renvoie l'index de la dernière valeur du tableau
- **Length:** renvoie la taille du tableau
- **Remove Index:** supprime un élément à une position déterminée dans le tableau
- **Remove Item:** prend en entrée une valeur, recherche cette valeur dans le tableau et la supprime
- **Resize:** redimensionne le tableau. Si la nouvelle taille est plus petite que la précédente alors tous les éléments dont l'index est supérieur à la nouvelle taille sont supprimés. Si toutefois la nouvelle taille est plus grande, alors des éléments vides sont ajoutés automatiquement.
- **Set Array Elem:** Change la valeur d'un élément du tableau à une position déterminée
- **Shuffle:** réorganise le classement du tableau de façon aléatoire

Après, il existe quelques fonctions spécifiques en fonction du type de variable (exemple: «Max of Float Array» qui renvoie la valeur maximale lue dans un tableau de flottants) que nous ne listerons pas ici mais qui vous seront proposées directement par l'éditeur.

Instructions de branchement (flow control)

Nous avons vu quelques instructions de contrôle comme le SI («branch») pour tester une condition. Unreal Engine intègre la plupart des instructions de branchement des langages de programmation:

- **Branch:** prend en entrée une condition et redirige l'exécution suivant que la condition est remplie (sortie «True») ou non (sortie «False»).
- **DoN:** réalise une opération (sur sa sortie «Exit») N fois (N étant passé en entrée) puis bloque l'exécution jusqu'à ce que son entrée «Reset» soit sollicitée.
- **DoOnce:** équivalent à DoN avec N égal à 1.

- **FlipFlop**: au premier appel, exécute la sortie A. Au second appel, la sortie B. Au troisième appel, de nouveau A, etc.
- **ForLoop**: boucle allant de l'index «First Index» (en entrée) à «Last Index» (en entrée).
- **ForLoopWithBreak**: équivalent à ForLoop, mais avec la notion de «break» en plus.
- **Gate** (porte): permet d'ouvrir (entrée «Open») et de fermer (entrée «Close») un flux d'exécution relié à l'entrée «Enter». Si le flux est fermé, l'exécution s'arrête là, sinon, elle continue par la sortie «Exit». L'entrée «Toggle» permet de fermer un flux ouvert ou d'ouvrir un flux fermé: elle inverse l'état de la porte.
- **MultiGate**: permet d'exécuter une suite de plusieurs instructions reliées chacune à une sortie du Multigate. Mais une seule à chaque fois! Au premier appel, on exécute le traitement relié à «Out0», au second appel «Out1», etc. On peut ajouter autant de sorties que l'on souhaite. Si «Random» est activé, alors l'ordre n'est pas respecté et la sortie est tirée au hasard. Si «Loop» est activé, l'appel suivant l'exécution de la dernière sortie reprend à «Out0». L'entrée «Start Index» permet de ne pas forcément commencer à «Out0», mais de choisir l'index de la première exécution.
- **Séquence**: permet d'exécuter une suite de plusieurs instructions reliées chacune à une sortie, de façon séquentielle. On peut ajouter autant de sorties que l'on souhaite. Lorsqu'on entre dans la séquence, on exécute «Then0», puis «Then1», puis «Then2», etc. La différence avec Multigate, est qu'on appelle une seule fois séquence pour exécuter toutes ces instructions, alors que Multigate n'exécute qu'une instruction par appel.
- **WhileLoop**: effectue une série d'instructions en boucle (sortie «Loop Body») tant qu'une condition n'est pas remplie (entrée «Condition»). Quand cette dernière est remplie, le traitement continue par la sortie «Completed».

L'ÉDITEUR DE MATÉRIAUX



3

CRÉATION D'UN NOUVEAU MATERIAU

Pour cela, commençons par déposer un objet particulier sur la scène et jouer avec les matériaux, le «présentoir» de matériaux fourni par Unreal répond bien à ce besoin:

1. Revenez dans l'éditeur de niveau et dans **CB**, sélectionnez l'asset «SM_MatPreviewMesh_02» du répertoire «Props» de «StarterContent» et déposez-le sur votre scène. Renommez l'actor en «MaterialTest».

Dans **DT**, panel «Materials», on voit que l'objet possède 2 matériaux: «Element 0» et «Element 1».

Quand nous avons modélisé l'escalier, nous avons utilisé une «brush», il ne s'agit pas d'un static mesh. Cela signifie que chaque surface possède son propre matériau. Mais dans le cas d'un «StaticMesh», c'est à dire un objet qui a probablement été modélisé avec un logiciel externe comme Blender, 3DSMax ou Maya, l'objet est indivisible.

D'ailleurs, en allant dans le volet «Geometry Editing» de **MD**, vous vous rendez compte qu'il n'y a ni point, ni arrête, ni surface sélectionnable. On ne peut pas modifier la géométrie de l'objet, hormis un redimensionnement.

Toutefois, il est possible d'agir sur les matériaux. Quand l'asset «SM_MatPreviewMesh_02» a été exporté, il possédait 2 matériaux – il pourrait en avoir 10 ou plus. Cela signifie que l'artiste a découpé l'objet de telle façon à appliquer tel matériau sur telle surface, et tel autre matériau sur l'autre. Ainsi, nous voyons 2 materials sous UE4.

Un «Material» sous UE4 est considéré comme un **Asset**. Cela signifie qu'on peut le trouver dans **CB**, s'en servir dans un projet, mais aussi l'utiliser dans d'autres. Il est ainsi possible d'importer des banques de matériaux provenant de l'extérieur, et même de les acheter sur le Marketplace.



Illustration 59: Fenêtre "Details" d'un objet contenant 2 matériaux

Examinons comment créer un nouveau «Material»:

2. Dans **CB**, créez un nouveau répertoire «Materials» directement sous «Game» et sélectionnez ce nouveau répertoire.
3. Puis, Cliquez sur «**Create**», sélectionnez «**Material**» et renommez «Mat1»
4. Appliquez «Mat1» sur l'objet «MaterialTest» (en évitant le «U» d'Unreal) en réalisant un glisser-déplacer de «Mat1» du **CB** vers l'objet dans le viewport.

Vous remarquez qu'il se dote d'une surface en damier qui signifie «non défini» comme sur l'illustration 59.

Dans **DT**, panel «Materials», c'est «Element 0» qui contient ce nouveau matériau.

5. Cliquez sur la petite flèche de l'encadré rouge: vous retrouvez le matériau d'origine de l'asset.
6. Nous recommandons l'opération précédente, mais cette fois-ci en glissant-déplaçant «Mat1» sur la zone encadrée en vert de **DT**.

Nous obtenons le même résultat. Nous aurions pu également sélectionner «Mat1» dans **CB** et utiliser la petite flèche de l'encadré jaune. Ces 3 opérations reviennent au même. Quant à la petite loupe, elle sert à sélectionner le matériau dans **CB**. Retenez bien ces 3 petits boutons car ils servent à bien d'autres endroits et ont toujours cette fonction.

Présentation de la suite Quixel

Avec ses 4 outils dédiés (DDO, NDO, 3DO et Megascans), la suite Quixel vous propose une série d'outils de texturing 3D plutôt impressionnantes: les matériaux sont conçus à partir de textures issues de clichés pris en très haute définition et d'outils maison (par exemple des scanners 3D pour les normal maps).

- Megascans est une librairie en ligne possédant plus de 1000 scans de matériaux ultra-réalistes!
- NDO et DDO sont des outils de texturing pour le jeu vidéo.
- 3DO est un plugin Photoshop gratuit pour avoir un rendu temps réel de matériaux reposant sur la physique.



Site officiel: www.quixel.se



Illustration 60: image provenant d'une modélisation de la jungle sous Unreal Engine 4 utilisant Megascans de Quixel

DÉCOUVERTE DE L'INTERFACE

Nous allons maintenant éditer notre nouveau matériau: pour cela double-cliquez sur «Mat1» dans **CB** pour ouvrir l'éditeur de matériau: une nouvelle fenêtre comme sur l'illustration 61 s'ouvre.

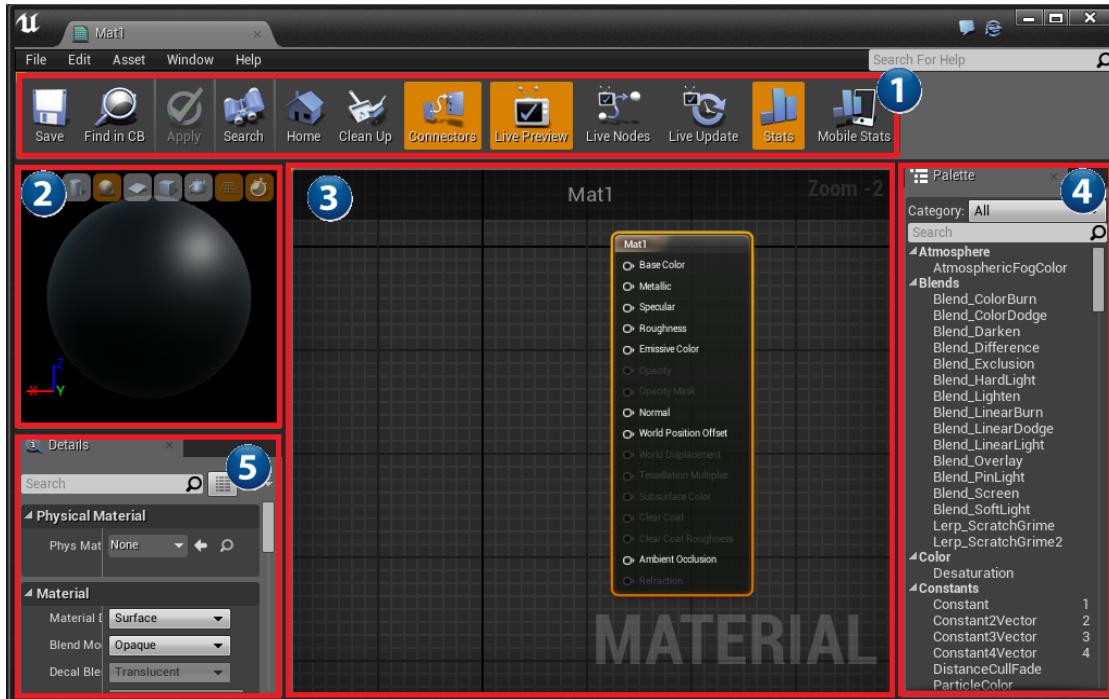


Illustration 61: Éditeur de matériaux à l'ouverture d'un nouveau matériau

Faisons connaissance avec cette nouvelle interface:

- Zone 1: barre d'outils** principale. En dehors des fonctions dont nous avons parlé précédemment, on trouve:
 - «**Home**» permet de recentrer le canevas sur le nœud principal («Mat1» de la zone 3).
 - «**Clean Up**» permet de supprimer tous les liens qui ne sont pas utilisés, c'est à dire non-reliés à notre noeud principal.
 - «**Connectors**», lorsqu'il est actif (orange), affiche les détails des nœuds, c'est à dire les entrées et sorties comme pour un Blueprint.
 - «**Live preview**» permet d'activer l'affichage en temps-réel de toutes les modifications apportées, et ce directement dans la zone 2.
 - «**Live Nodes**» et «**Live Update**» sont deux notions plus complexes que nous verrons un peu plus tard lors d'un exemple pratique.
 - «**Stats**» permet d'afficher une autre petite fenêtre avec les statistiques de performance liées au matériau courant, et «**Mobile Stats**» ajoute des éléments pour les mobiles (Tablets, smartphones, etc.)
- Zone 2:** fenêtre «**preview**» permettant d'afficher en temps-réel les modifications apportées au graphe (si «**Live Preview**» de la barre d'outils est actif).
 - Plusieurs modèles types sont possibles (cylindre, sphère, plan, cube), dont l'affichage directement d'un l'objet symbolisé par la «thière». Pour cela, vous devez sélectionner préalablement un objet dans **CB** (seule l'élément 0 de cet objet sera concerné). Vous pouvez changer le modèle d'affichage en cliquant sur la petite icône correspondante de la barre d'outil intégrée à cette fenêtre.
 - Vous pouvez également activer/désactiver l'affichage de la grille du repère.
 - La dernière icône «**Toggle realtime rendering**» permet d'activer ou de désactiver les rendu

- tempis réel des particules et autres animations.
- Concernant les manipulations sur l'objet: Un clic gauche permet d'effectuer une rotation de l'objet, permet de jouer sur le zoom (tout comme la molette), et le bouton central de déplacer l'objet.
- Vous pouvez déplacer la source lumineuse en pressant «+clic gauche» et en bougeant la souris simultanément.
- **Zone 3:** Il s'agit du canevas principal contenant les différents nœuds qui forme un graphe. Un matériau se défini exactement comme un programme comme vous allez bientôt pouvoir le constater.
- **Zone 4:** fenêtre «**Palette**». Elle contient toutes les fonctions qui peuvent être utilisées pour composer le graphe. Comme vous pouvez le constater, les fonctions sont très nombreuses et lorsqu'elles sont combinées ensemble, elles donnent des résultats fort différents. Créer des matériaux complexes est donc tout un art à part entière (sans compter le travail préalable sur les textures). Il faudrait un livre complet pour aborder toutes les notions de l'édition de matériaux sous UE4, mais nous allons étudier les principales.
- **Zone 5:** fenêtre «**Details**» (ou **DT**). Inutile de vous présenter cette fenêtre, mais il ne s'agit pas des détails du matériau, mais bel et bien des détails du nœud sélectionné.

Le matériau définit certaines propriétés de la lumière renvoyée par l'objet (diffuse, spéculaire, etc.), mais aussi ses caractéristiques physiques. Un matériau peut être composé de plusieurs textures qui peuvent être combinées entre-elles.

Examinons le nœud principal de la zone 2. Plusieurs entrées correspondent au différents paramétrages possibles d'un matériau:

- **«Base Color»:** C'est la couleur de base du matériau
- **«Metallic»:** Capacité du matériau à refléter l'environnement comme un miroir (comme le ferait un métal)
- **«Specular»:** Quand la lumière frappe une surface avec un certain angle, elle est réfléchie au même angle à l'opposé. Si un observateur se trouve sur le chemin de ce rayon réfléchi, il reçoit une plus grande quantité de lumière. C'est d'autant plus fragrant avec le métal.
- **«Roughness»:** La granularité, la rugosité. Une surface ayant une forte granularité va refléter la lumière dans plusieurs directions. Une granularité de 0 offre une matériau très lisse réfléchissant la lumière, alors qu'une granularité de 1 donne un aspect mat.
- **«Emissive color»:** C'est la capacité d'une matière à émettre de la lumière, lumière qui se répercute sur les objets environnants.
- **«Opacity»:** Plus un objet est opaque est moins on voit à travers. Un objet transparent a une opacité proche de 0.
- **«Opacity Mask»:** Permet d'utiliser un masque comme composante d'opacité. Au lieu d'une explication théorique, nous allons l'utiliser un peu plus loin.
- **«Normal»:** c'est la composante des normal maps. Le normal mapping consiste à utiliser une texture pour donner une impression d'un relief à une surface. L'exemple type est la brique. Suivant le déplacement de la lumière, les joints produisent une ombre différente. C'est un procédé très peu coûteux en ressources qui apporte un réalisme important au matériau d'un objet.
- **«World position offset»:** permet de déformer l'objet en utilisant un vecteur directionnel. Ainsi, on peut donner une impression de gonflement ou de réduction.
- **«World displacement»:** Sur la base d'une normal map, permet d'induire une déformation de l'objet. Plus simplement, on crée du relief «réel» là où la normal map ne faisait que dévier la lumière. A utiliser en corrélation avec le paramètre suivant.
- **«Tesselation multiplier»:** multiplicateur de subdivision (voir encadré ci-contre). Il s'agit d'un itérateur: un multiplicateur de 2 signifie qu'on divise chaque face par 2x2, soit 4, un multiplicateur de 3 → 8 fois, un multiplicateur de 4 → 16 fois, etc. Attention donc à ne pas utiliser des valeurs trop

élevées.

- «**subsurface color**»: couleur de l'intérieur de l'objet lorsque la lumière le traverse.
- «**Clear Coat**»: utilisé pour mieux simuler les matériaux multicouches qui ont une couche translucide mince. Exemples: les acryliques, les laques, les films colorés sur des métaux tels que des canettes de soda, de la peinture de la voiture, etc.
- «**Clear Coat Roughness**»: rugosité de la couche précédente.
- «**Ambient Occlusion**»: permet d'agir directement en accentuant ou non l'effet de l'occlusion ambiante. Cela n'agit pas sur la lumière directe.
- «**Refraction**»: la lumière est déviée lorsqu'elle passe d'un milieu transparent à un autre (par exemple: de l'air à l'eau). On observe ce phénomène lorsqu'on regarde une paille dans un verre: celle-ci paraît brisée. La lumière est dite «réfractée».



L'Ambient occlusion (AO) est une technique permettant d'apporter plus de réalisme à un scénario. Plus des faces sont rapprochées, plus la quantité de lumière diminue entre ces faces. C'est un peu l'ombrage qu'un objet projette sur lui-même.

La **tesselation** est un procédé de décomposition des polygones en éléments plus petits.

C'est l'équivalent du «**subdivide**» de Blender ou des outils de «**subdivision surface**» de 3DSMax. En créant des éléments plus petits, on affine davantage le maillage de l'objet, c'est à dire en lissant la géométrie par un procédé d'interpolation linéaire. Si on ajoute le «**displacement mapping**» à cela, on peut obtenir des modèles très réalistes à partir de modèles «Lowpoly».

UN MATÉRIAUX MÉTALLIQUE

Maintenant que vous avez fait connaissance avec l'interface, ainsi que les principales notions, continuons notre création de matériau:

1. Dans la «palette», sous «**Constants**», sélectionnez «**Constant3Vector**» et ajoutez-le sur canevas.
2. Double-cliquez sur le nœud, une fenêtre comme sur l'illustration 62 apparaît et choisissez la couleur de votre choix.

Attention, la couleur obtenue est visible dans «new» (encadré jaune). Il faut donc composer avec la palette de couleur (rond de gauche) et les différents curseurs (encadrés bleus).

- La **couleur** est représentée par une composante Rouge, Verte, Bleue (ou RGB pour «Red Green Blue»). Il est possible d'entrer la valeur directement dans les champs correspondants sous le panel «**Advanced**».
- Le modèle **Teinte Saturation Valeur** (ou HSV pour «Hue Saturation Value») est un autre système de représentation de la couleur. Vous pouvez de la même façon entrer la valeur manuellement.
- Si cette sélection de couleur ne vous convient pas, vous pouvez cliquer sur le bouton de l'encadré vert pour passer en mode «color spectrum» qui est un autre mode de sélection.

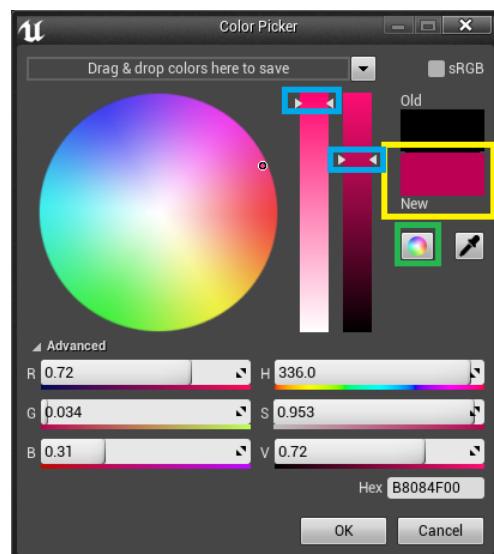


Illustration 62: Fenêtre "Color Picker" permettant de sélectionner une couleur

- A droite de l'encadré vert, vous avez le «**color picker**» qui permet de copier une couleur prise n'importe où dans l'éditeur.
- Enfin, la case «**sRGB**» permet d'activer la correction «gama» en fonction de votre écran, en respectant la norme sRGB. Cela ne change rien au paramétrage de la couleur.

3. Reliez la sortie de ce nœud à l'entrée «Base Color» de «Mat 1». Si «**Live Preview**» est actif, vous verrez tout de suite le résultat de l'opération à l'écran.
4. Dans la «palette», sous «Constants», sélectionnez «Constant» et ajoutez-le sur canevas. Dans **DT**, indiquez «1.0» dans «Value».
5. Reliez la sortie de ce nœud à l'entrée «**Metallic**».

Vous obtenez un matériau qui reflète la lumière, comme le ferait un métal. La valeur maximale est ici «1.0», toute valeur supérieure produira le même résultat.

6. «Copier-Coller» ce dernier nœud (sélectionnez le nœud, puis **Ctrl+C**, puis **Ctrl+V**) et prenez une valeur de 0.
7. Reliez la sortie de ce nœud à l'entrée «**Roughness**».

Vous obtenez un matériaux réfléchissant. On vient de passer d'un métal style «brossé» à un chrome.

8. **Sauver** et revenez dans l'éditeur de niveau. Un «**Build**» va être nécessaire pour prendre en compte ce nouveau matériau.

Vous obtenez quelque chose de sensiblement proche de l'illustration 63. Remarquez l'environnement qui se reflète en temps-réel sur l'objet, ainsi que les lumières provoquant éblouissement (HDR) et la création de «Lens Flare» (Aberration optique due à une diffusion parasite de la lumière à l'intérieur d'un objectif). C'est tout de même du plus bel effet non?



Illustration 63: Matériau de type métallique sans rugosité: comme de l'or.

On pourrait être tenté de créer des miroirs avec ce type de matériaux, mais si vous réalisez cette expérience vous serez déçu. Il y a une approche beaucoup plus intéressante pour cela que nous verrons un peu plus loin.

LES MATÉRIAUX TEXTURÉS

Pour créer une surface texturée, nous avons besoin au minimum d'une image. Mais la plupart des matériaux texturés en utilisent 3: une pour la couleur de base, une pour la sécularité et une dernière pour la normal map.

1. Créez un nouveau matériau «Mat2», appliquez-le à notre actor «MaterialTest» et ouvrez l'éditeur de matériau avec «Mat2».
2. Dans **CB**, sélectionnez les textures «T_Cobblestone_Pebble_D» (Diffuse map), «T_Cobblestone_Pebble_M» et «T_Cobblestone_Pebble_N» (normal map) qui se trouvent dans le répertoire «Textures» de «Starter Content», puis faites un «glisser-déplacer» sur le canevas de l'éditeur de matériaux.

3 nouveaux nœuds apparaissent, de type «Texture Sample».

3. Reliez les nœuds comme sur l'illustration 64.

Plus précisément, commencez par le nœud de la «diffuse map», observez le résultat. Bougez un peu le modèle dans le «preview». Ensuite, ajoutez le second nœud et recommencez l'opération. Observez comment la lumière se déplace sur les pierres donnant un aspect «mouillé» à l'ensemble.

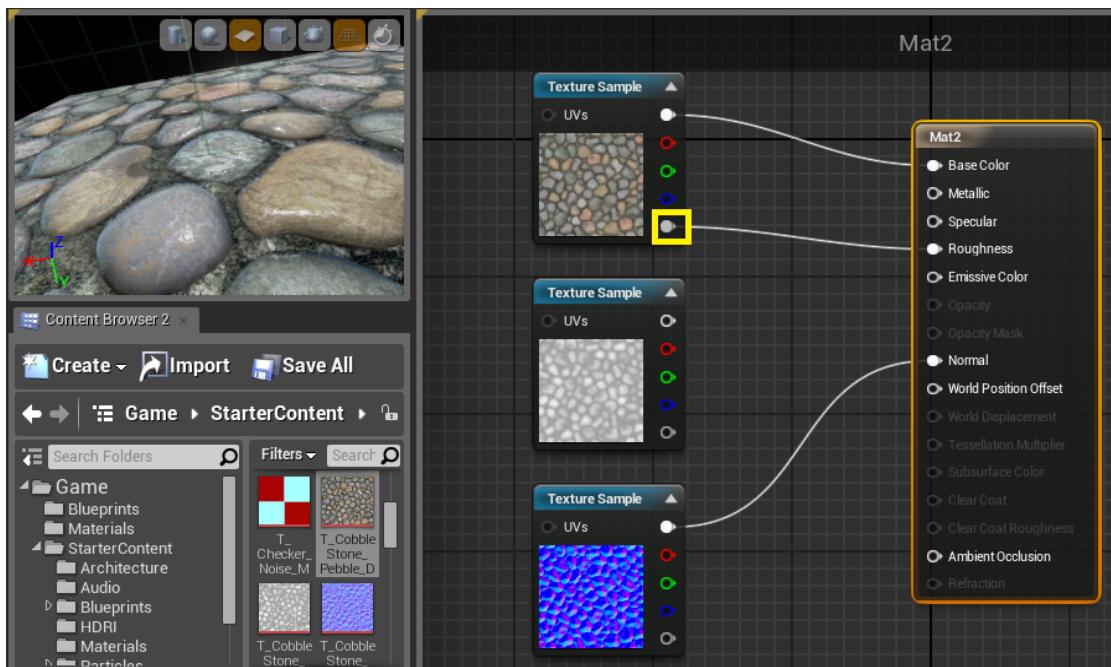


Illustration 64: Effet «mouillé» réalisé à l'aide du canal alpha de l'image de base et la normal map

Pour mieux comprendre:

4. double-cliquez sur la texture «T_Cobblestone_Pebble_D» dans **CB**.

Vous obtenez l'éditeur de textures d'UE4 comme dans l'illustration 65.

5. Cliquez sur le bouton view (encadré rouge) et décochez les canaux «Red», «Green», «Blue» et cochez «Alpha».

Reprenez l'illustration 64 et examinez un nœud «Texture Sample». La 1ère sortie en partant du haut (blanche) correspond à «Tous les canaux». Mais on peut aussi ne choisir qu'un ou quelques canaux en utilisant les sorties correspondantes: rouge: «red», verte: «green», bleue: «blue» et grise: «Alpha».

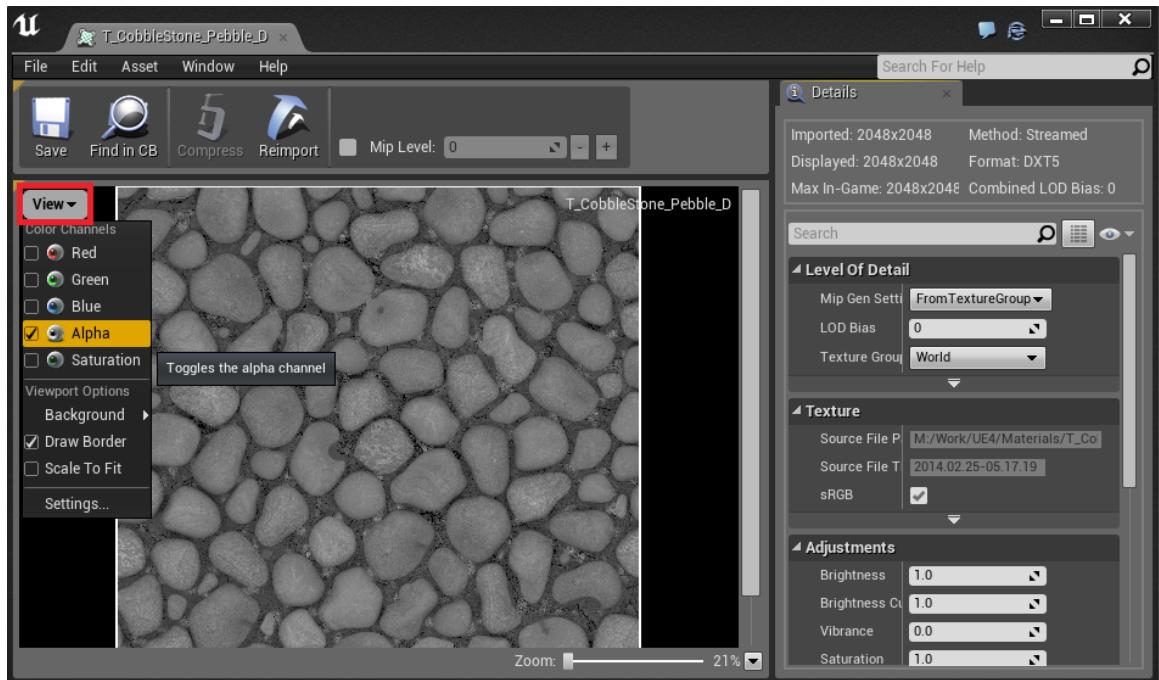


Illustration 65: Editeur de Texture, sélection du canal alpha

Reprenez l'illustration 65: l'image que nous voyons est celle du canal alpha de la texture sélectionnée. Plus un point de l'image est sombre et plus sa valeur s'approche de 0. Plus un point est lumineux, et plus sa valeur se rapproche de 1.

Rappelez-vous comment fonctionne «**Roughness**»: Une granularité de 0 offre une matière totalement réfléchissant comme un miroir, alors qu'une granularité de 1 donne un aspect mat. Appliquée comme nous l'avons fait précédemment, avec une constante, c'est toute l'image qui est concernée par une valeur constante. Appliquée avec cette texture, chaque point de l'image se verra appliquer la valeur correspondante dans l'image de la figure 65. L'effet mouillé est du à cela, et aussi grâce à l'utilisation de la normal map qui permet de gérer le relief de l'objet.

Notez que nous aurions pu utiliser l'ensemble des canaux pour créer le même effet, mais l'explication est plus claire ainsi.

- 6. Désactivez la normal map en cassant le lien (sélectionnez la sortie et → «Break Links») et vous voyez comment fonctionne exactement le «Roughness» avec la texture.
 - 7. Pour renforcer l'effet, utilisez le nœud de type «Texture Sample», et reliez-le à l'entrée «Metallic».
- Les pierres semblent encore plus mouillées: bougez la lumière (avec «+clic gauche») et regardez les pierres scintiller comme si de l'eau parcourait leur surface.

Nous allons maintenant **changer la taille des pierres sur la texture**:

- 8. Attrapez l'entrée «UV's» du premier nœud et ajoutez un nœud de type «TextureCoordinate».
- 9. Sélectionnez ce nouveau nœud et dans «Details», panel modifiez «Utile: 2» et «Vtiling:2».
- 10. Reliez la sortie de ce nœud à l'entrée «UV's» des deux autres nœuds de type «Texture Sample».

Ainsi, on dispose de 4 fois (2x2) plus pierres sur le nouveau matière. Cela devrait vous rappeler les opérations que nous avons réalisées avec les marches. Par contre, ici, au lieu de modifier l'UV Mapping de l'objet, nous modifions directement le matière. Notez la possibilité dans **DT** de réaliser un miroir vertical ou horizontal de l'application de la texture.

Pour créer une variation dans le matériau, nous pouvons combiner deux normal maps:

11. Importer dans un nouveau matériau les textures «T_Brick_Clay_New_D», «T_Brick_Clay_New_N» et «T_RockMesh_N»
12. Reliez les nœuds comme sur l'illustration 66.

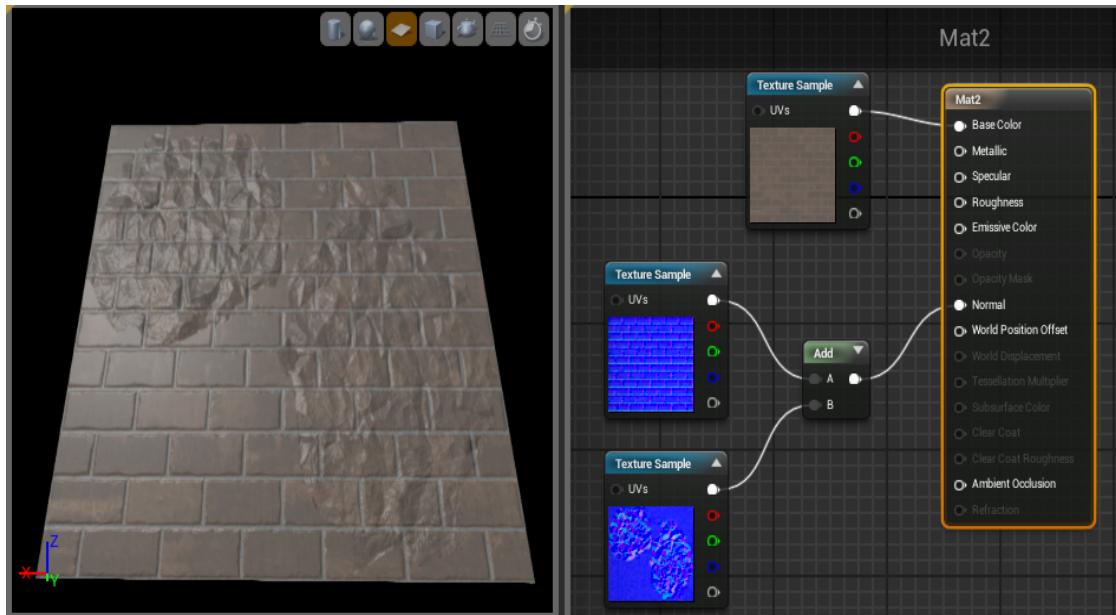


Illustration 66: Addition de deux normal maps

UN MATÉRIAU DE TYPE «MIROIR»

Nous avons vu que l'on pouvait créer un matériau métallique avec un très fort coefficient de réflexion, au point où nous aurions pu vouloir l'utiliser pour créer des miroirs.

Nous allons maintenant utiliser une classe bien particulière «Scene Capture 2D» pour effectuer le miroir, elle peut aussi servir pour créer des caméras de sécurité.

Concrètement, il s'agit d'une caméra que l'on oriente comme on le souhaite et qui envoie le contenu de ce qu'elle filme dans une texture pouvant ensuite être utilisée dans un matériau.

Passons à la pratique:

1. Créez un actor «mirror» en utilisant l'asset «Shape_Plane» contenu dans le répertoire «Props/Shapes» de «StarterContent».
2. Effectuez les rotations et le redimensionnement nécessaires et fixez-le sur un des murs du bâtiment.

Ajoutez-lui un composant pour capturer la scène et la restituer dans une texture:

3. Dans **DT**, cliquez sur «Add Component» et sélectionnez «Scene Capture Component 2D»

Nous aurions pu utiliser la classe «Scene Capture 2D» qui est accompagnée visuellement d'une caméra, mais il est plus commode de réaliser un composant comme nous venons de le faire.

4. Cliquez ensuite sur «Blueprint/Add Script».
5. Sélectionnez le répertoire «Blueprint» et renommez-le en «BP_Mirror», puis «Create Blueprint»

Nous allons créer un Blueprint avec les deux éléments. Il est possible de créer ce Blueprint puis d'ajouter les deux éléments, mais nous avons préféré vous montrer cette démarche qui est rendue possible avec

l'apparition de la version 4.7 de l'éditeur.

6. Dans **CB**, répertoire «Textures», cliquez sur «Add New», «Materials & Textures» puis «Render Target». Nommez-le «**T_Mirror**»
7. Ouvrez «**T_Mirror**» en édition
8. Dans **DT**, panel «**Texture Render Target 2D**», modifiez «**Size X**» et «**Size Y**» en **1024**.

La résolution de l'image du miroir dépend de ces deux paramètres. Si vous utilisez de petits miroirs, vous pouvez baisser la taille, et inversement.

9. Ouvrez le Blueprint «**BP_Mirror**» en édition
10. Dans la fenêtre «Components», sélectionnez «**SceneCaptureComponent2D**»
11. Dans **DT**, panel «**Scene Capture**», «**Texture Target**», sélectionnez «**T_Mirror**»
12. Modifiez «**Field of View**» en **110**

«Field of View» correspond à l'ouverture du champ de la caméra. Plus sa valeur est élevée, et plus le champ est large.

13. Panel «**Transform**», modifiez «rotation X»: -90 et «rotation Y»: 90

Dans la pratique, si votre caméra semble filmer le plafond ou une autre direction, il faut modifier ces valeurs de rotation.

La texture est associée, il reste à créer un matériau pour afficher l'image sur le plan qui nous sert de miroir:

14. Créez un nouveau matériau «**M_Mirror**» et ouvrez-le en mode édition.
15. Ajoutez la texture «**T_Mirror**» en entrée de «**Base Color**»
16. Ajoutez un nœud «**Texture Coordinates**» et reliez-le à l'entrée «**UVs**» de la textures
17. Sélectionnez ce nœud et dans **DT**, modifiez «**UTiling**» en **-1** pour avoir le reflet inversé comme dans un miroir.
18. Sauvez et appliquez «**M_Mirror**» au Blueprint «**BP_Mirror**».



Illustration 67: Test du miroir avec.... mais c'est le personnage du tome 2 ?

Normalement, l'image doit changer en fonction de la position de l'observateur, ce qui n'est pas le cas ici. Qu'à cela ne tienne, vous pouvez adapter la rotation du composant «**SceneCaptureComponent2D**» en fonction de l'observateur!

PEINDRE UN OBJET AVEC LE VERTEX PAINTING

Nous ne pouvions pas aborder la peinture d'un objet, qui correspond à l'outil «Paint» de **MD** sans avoir préalablement vu comment on crée un matériau car les deux sont étroitement liés.

1. Dans **CB**, «StarterContent», «Props», sélectionnez «**Material Sphere**» et ajoutez-le à la scène.

Nous n'avons pas choisi cet asset par hasard. Pour que le vertex painting fonctionne, il faut que l'objet contienne un nombre de vertex minimum. Si vous tentez de le faire ainsi sur un plan, vous ne pourrez pas dessiner à la peinture dessus, car il n'y a que 4 vertex pour faire un plan. L'objet que nous venons d'ajouter contient tout juste suffisamment de vertex pour réaliser l'opération.

2. Créez un nouveau matériau «**MatPaint**» et appliquez-le à ce nouvel actor, puis ouvrez-le avec l'éditeur de matériau.
3. Sous l'éditeur de matériau, ajoutez juste un nœud «**Vertex color**» et reliez-le à l'entrée «Base Color» du matériau, puis appliquez et sauvez.
4. Revenez à l'éditeur de niveau, sélectionnez l'actor précédemment ajouté et allez dans **MD**, volet «Paint» (ou **Shift + F2**)
5. Assurez-vous que «**Vertices**» est coché (et non «Textures»)
6. Choisissez une nouvelle couleur dans «Paint Color»
7. Dans «Radius», indiquez 25, dans «Strength» 0.4 et indiquez un «Falloff» de 0.5

Le «radius» correspond à la taille du pinceau que nous utilisons pour peindre.

«Strength» est la force avec laquelle la couleur s'applique et se mélange à la couleur existante (de 0 à 1).

«Falloff» correspond à ce second cercle que vous apercevez dans la brosse symbolisant le pinceau. La force est d'autant plus importante dans le cercle circonscrit au second qu'au dehors.

8. Puis dirigez votre curseur souris vers l'actor et **commencez à peindre**.

On peint en maintenant le clic gauche. Pour utiliser la couleur d'effacement («Erase Color»), utilisez

«Shift + clic gauche»

9. Changez de couleur et répétez l'opération, vous allez obtenir un objet tel que sur l'illustration 68

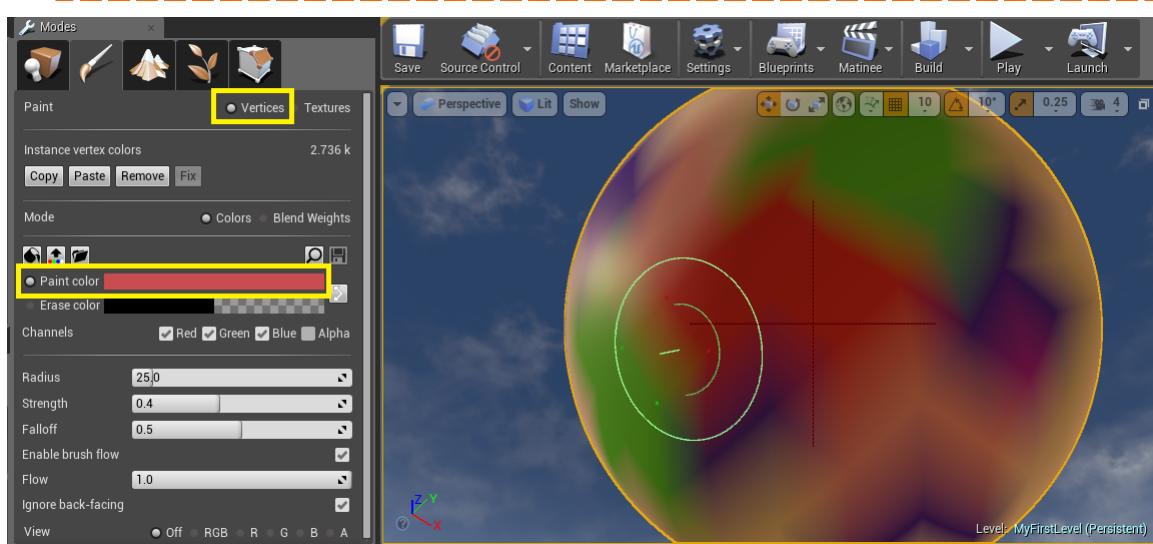


Illustration 68: Utilisation du vertex color painting

Pour recommencer et effacer la peinture réalisée, il suffit de cliquer sur «**Remove**».

«Copy» et «Paste» permettent d'appliquer notre œuvre d'art à un autre objet (mais il faut qu'il soit exactement du même type, avec le même nombre de vertex).

«Fix» permet de corriger une peinture si on importe une nouvelle géométrie de l'actor qui ne contient pas exactement le même nombre de vertex.

Il est aussi possible de peindre au dessus d'une texture. Dans ce cas, il suffit de modifier notre matériau et d'ajouter un multiplicateur comme dans l'illustration 69 :

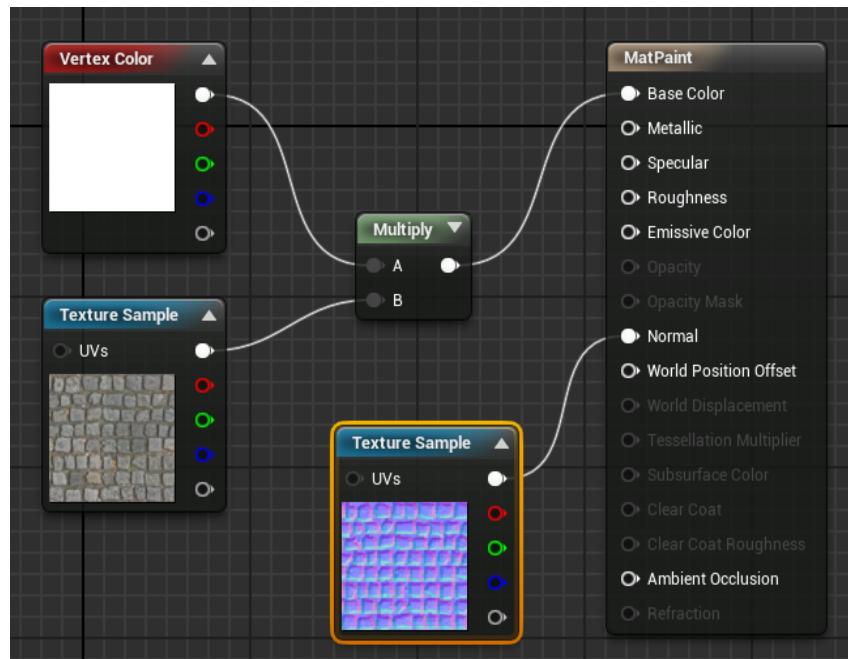


Illustration 69: Combinaison du vertex painting avec une texture

L'UTILISATION DES MASQUES

Nous allons étudier le fonctionnement des masques qui sont très utiles pour combiner des textures, mais également pour créer des masques de transparence par exemple. Nous allons en profiter pour voir de nouvelles fonctions également:

1. Créez un nouveau matériau «Mat3» et appliquez-le à l'actor «MaterialTest» et ouvrez l'éditeur de matériau avec «Mat3».
2. Dans **CB**, sélectionnez la texture «T_Tech_Dot_M» et ouvrez-la avec l'éditeur de texture.

Observez ses différents canaux un à un comme nous avons appris à le faire précédemment. L'illustration 70 montre les différents canaux de cette image.

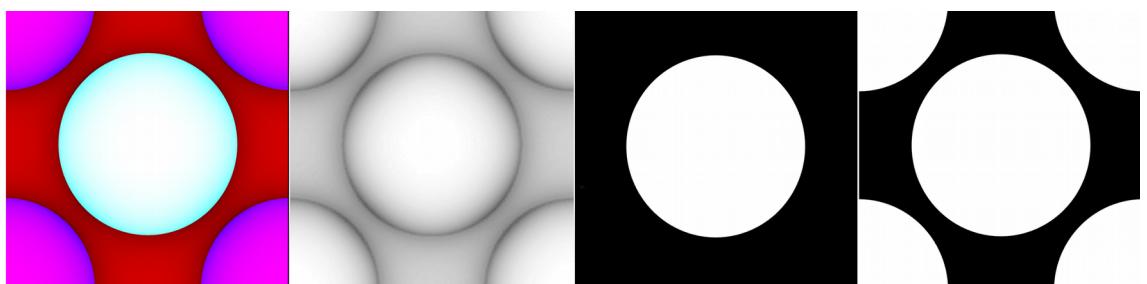


Illustration 70: Les différents canaux de la texture, de gauche à droite: les trois canaux (R+G+B), canal R, canal G, canal B

3. Ajoutez cette texture au canevas du matériau.

Pour bien comprendre comment fonctionnent les masques, nous utilisons 2 couleurs et nous allons nous servir de cette texture pour mélanger les couleurs :

4. Ajouter deux couleurs de votre choix au canevas (pour rappel, une couleur se trouve sous l'appellation «Constant3Vector» dans la palette).
5. Ajoutez un nœud de type «LinearInterpolate», dans «palette», onglet «Math».

Notez que vous pouvez aussi directement réaliser un  sur le canevas et chercher par le nom. Un nœud «Lerp» apparaît.

Cette fonction permet une interpolation linéaire entre deux valeurs à l'aide d'un coefficient. En prenant les valeurs 5 et 10 et en réalisant une interpolation linéaire de 0.5, on obtient 7.5. En faisant une interpolation linéaire de 0.8, on obtient $(10-5)*0.8 + 5 = 9$. Les interpolations linéaires sont très utilisées dans les animations. «Lerp» va nous permettre ici de mélanger les couleurs selon un canal alpha.

6. Reliez l'entrée A du «Lerp» à la première couleur, et l'entrée B à la seconde.
7. Reliez ensuite l'entrée Alpha à la sortie principale (R+G+B) de notre texture.

Que constatons nous? Rien du tout. C'est normal, nous n'avons pas relié la sortie du Lerp à l'entrée «Base Color» de «Mat3». Mais nous n'en avons pas besoin.

8.  sur le nœud «Lerp» et choisissez «Start Previewing Node».

Automatiquement, vous pouvez voir le résultat dans la fenêtre de «preview». Ce n'est pas très utile ici, mais si vous travaillez sur un matériau disposant d'une cinquantaine de nœuds, vous serez heureux de pouvoir

tester des résultats intermédiaires. Pour couper le preview, il suffira d'un nouveau  sur le nœud «Lerp» et de choisir «Stop Previewing Node».

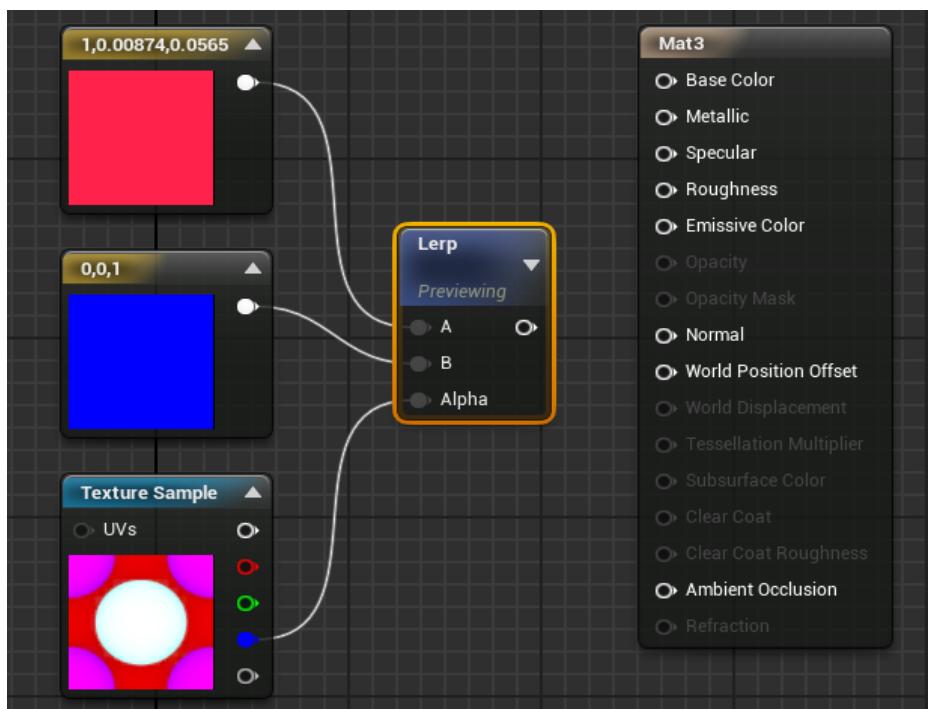


Illustration 71: Mat3 en mode "preview" sur le Lerp (sortie "Blue")

9. Testez l'opération sur les différentes bornes de sortie de notre texture.

L'illustration 72 montre le résultat obtenu en utilisant un vert et un bleu comme couleurs à mélanger.

Comme vous l'avez vu précédemment, le «Lerp» effectue réellement un mélange: il ne laisse pas forcément l'une ou l'autre couleur, mais réalise un mélange via le canal alpha. Ainsi, le bleu mélangé avec le vert est devenu un cyan dans les deux premières images de l'illustration. Les deux dernières sont plus faciles à comprendre car la combinaison est binaire: soit une couleur passe, soit l'autre, il n'y a pas de mélange à proprement parler.

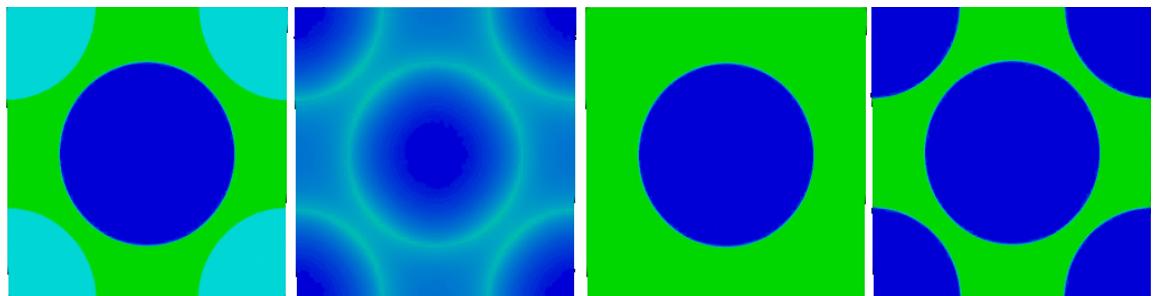


Illustration 72: Application successive des différentes sorties, de gauche à droite: les trois canaux (R+G+B), canal R, canal G, canal B

10. Enfin, si vous souhaitez utiliser «Mat3», ⚡ sur le nœud «Lerp» puis «Stop Previewing Node». Et reliez la sortie de «Lerp» à l'entrée «Base Color» de «Mat3». Puis Sauvez.

Maintenant que vous avez compris la technique des masques, examinons comment créer par exemple un mur recouvert de lierre:

1. Créez un nouveau matériau «IvyWall»
2. Ajoutez les textures «T_Brick_Clay_Old_D», «T_Brick_Clay_Old_N», «T_Brush_D» et «T_Brush_N» (à défaut de lierre, on utilise une texture qui s'en approche).
3. Reliez les éléments du graphe comme dans l'illustration 73:

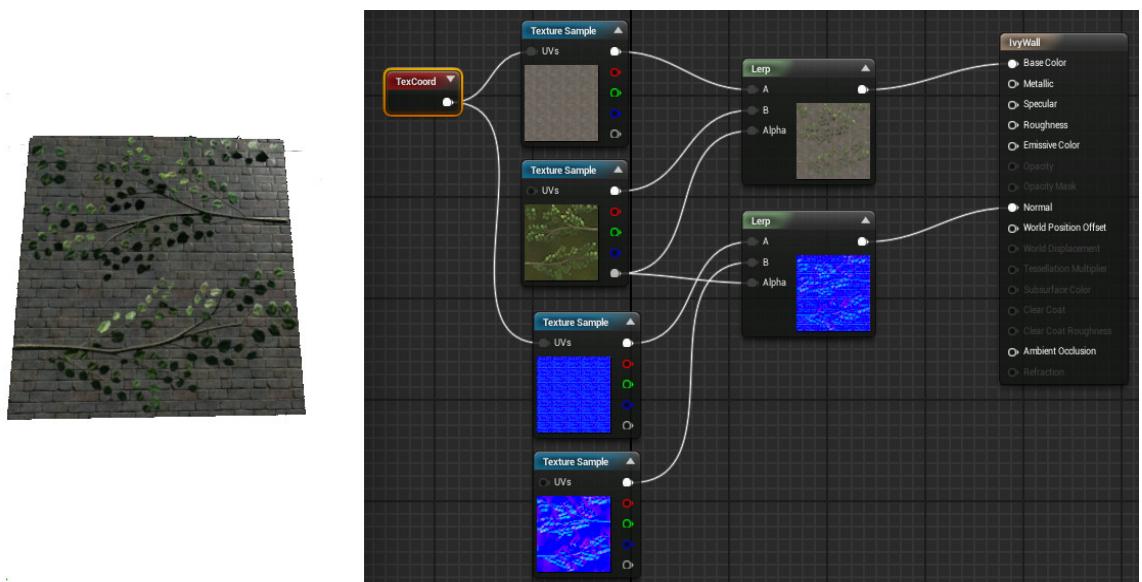


Illustration 73: Mixage entre deux textures en utilisant le canal alpha de la seconde

Cette fois-ci, nous avons utilisé le **canal alpha** de la texture du lierre pour déterminer quelle texture afficher en priorité.

VERTEX PAINTING ET MASQUES

De la même façon que nous avons «peint» un objet avec des couleurs, on peut aussi peindre avec de la texture. Nous avons vu précédemment que l'on pouvait «colorer» une texture avec cette méthode. Là, c'est différent et cela augure presque la façon dont nous allons peindre plus tard le paysage (même si la technique reste différente).

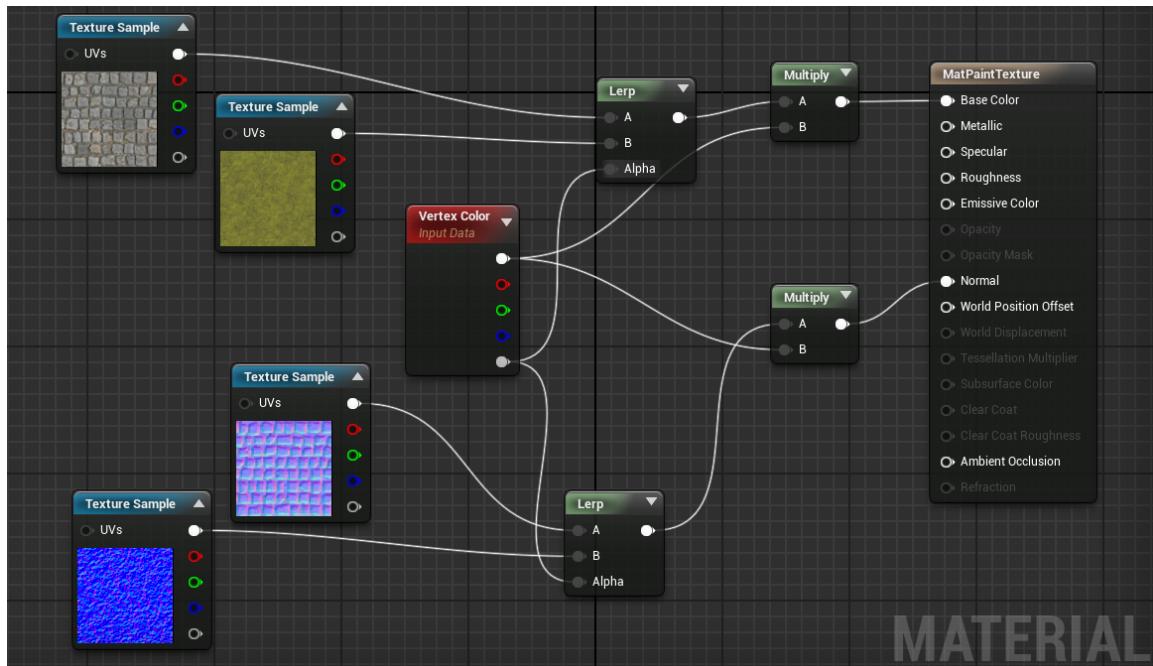


Illustration 74: Paramétrage du matériaux pour utiliser le canal alpha du vertex painting

Reprenez à partir de l'exemple précédent:

1. Dans «Materials», créez un nouveau matériau «MatPaintTexture» et ouvrez-le en édition
2. Ajoutez les éléments au graphe en suivant l'illustration 74

Choisissez 2 textures différentes, peu importe lesquelles, avec leurs normal maps associées. Remarquez qu'on utilise le canal alpha de la sortie du nœud «Vertex Color» pour doser le mélange entre les deux textures, grâce au nœud «Lerp» vu précédemment.

3. Dans **CB**, «StarterContent», «Props», sélectionnez à nouveau **«Material Sphere»** et ajoutez-le à la scène pour les tests
4. Appliquez-lui le nouveau matériau
5. Revenez à l'éditeur de niveau, sélectionnez l'actor précédemment ajouté et allez dans **MD**, volet «Paint» (ou «**Shift + F2**»)
6. Assurez-vous que «**Vertices**» est coché (et non «**Textures**»)
7. Choisissez une nouvelle couleur dans «Paint Color» - celle-ci n'a aucune importance, seule sa valeur alpha est importante. Choisissez un alpha de 1.0

L'alpha de 1 signifie que là où nous allons peindre, c'est la texture reliée à l'entrée B du LERP qui va apparaître. Si nous effaçons, c'est la texture de l'entrée A qui sera utilisée.

Si on choisit une couleur avec un alpha de 0, c'est l'inverse qui se produit.

8. On garde les mêmes valeurs que précédemment: «Radius» à 25, «Strength» à 0.4 et «FallOff» à 0.5.

9. Puis dirigez le curseur souris vers l'actor et **commencez à peindre** comme la fois précédente
(avec **Shift** enfoncé au début si vous avez choisi un **alpha** de 1).

On obtient un résultat proche de l'illustration 75 :

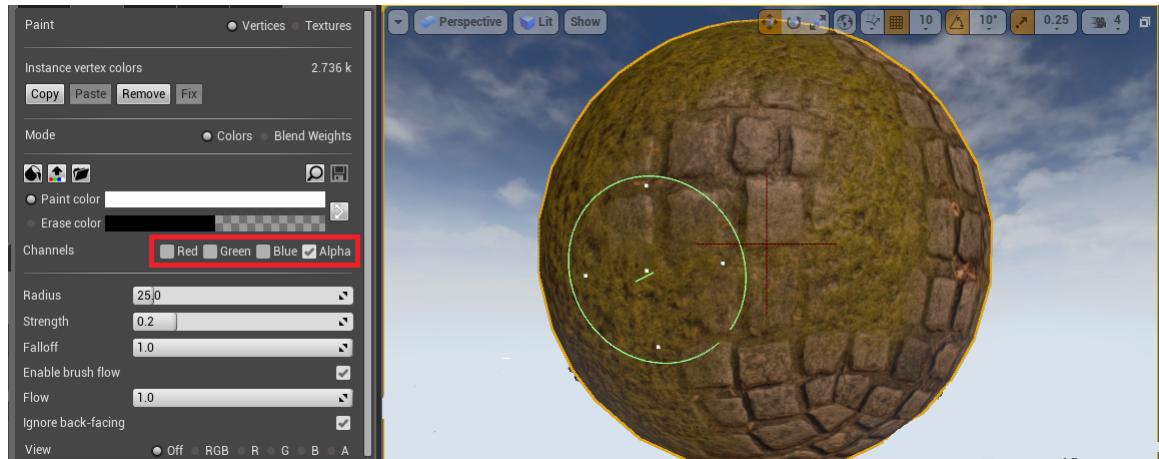


Illustration 75: Mélange de deux textures en utilisant le canal alpha du vertex painting

Grâce à cette technique, **vous pouvez ajouter facilement de la mousse sur les arbres et les rochers, un effet mouillé, ou bien ajouter des défauts, des traces de saleté ou de vieillissement** sur n'importe quel objet, sans avoir à passer par des «decals» (que nous verrons dans le tome 2).

GESTION DE LA TRANSPARENCE

Il y a deux façons de gérer la transparence: en utilisant un masque ou en rendant un matériau transparent.

- La première consiste à **utiliser une texture**, comme précédemment, pour rendre totalement transparent une partie de sa géométrie. Il n'y a aucun degré de transparence.
- La seconde consiste à rendre «progressivement» transparent le matériau en lui appliquant un **coefficent de transparence** (qu'on appelle souvent coefficient «Alpha»).

Première approche: la technique des masques

1. Créez un nouveau matériau «Mat4» et ouvrez l'éditeur de matériau.
2. Sélectionnez le nœud «Mat4» et dans **DT**, panel «Material», modifiez «Blend Mode» en «**Masked**».

L'entrée «**Opacity Mask**» de «**Mat4**» est maintenant **disponible**.

3. Toujours dans le panel «Material», cochez «**Two Sided**» pour que le matériau soit visible également «à l'intérieur» de l'objet.
4. Ajoutez la texture «T_Tech_Dot_M» et reliez sa sortie «Blue» à l'entrée «**Opacity Mask**» de «**Mat4**»
5. Ajoutez une nouvelle texture (celle de votre choix) et appliquez sa sortie à l'entrée «**Base Color**» de «**Mat4**».

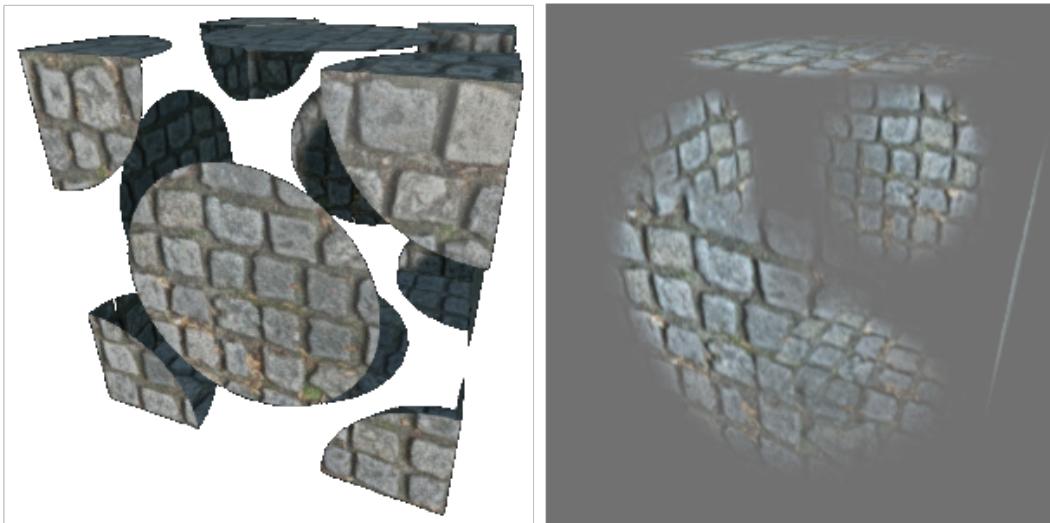


Illustration 76: Transparence par masque à gauche, transparence alpha à droite

Qu'observe t-on? L'image de gauche de l'illustration 76 représente l'application du matériau sur un Cube dans le preview. Rappelez-vous à quoi ressemble la sortie «Blue» de la texture «T_Tech_Dot_M», c'est la première image en partant de la droite de l'illustration 70. Au final, seules les parties en blanc de la texture seront affichées. Si nous n'avions pas coché «Two Sided» nous n'aurions pas vu la partie cachée de l'objet, c'est à dire la partie intérieure. Hors, la transparence rend l'intérieur visible. Afficher le matériau sur la face externe et interne de l'objet permet de rendre l'affichage plus réaliste.

Que serait-il arrivé si nous avions utilisé un gris plutôt qu'un noir ou un blanc? Rien du tout, il n'y a pas de degrés de transparence en utilisant cette entrée. Au dessus d'une certaine intensité de couleur, la texture devient visible, au dessous de cette intensité, elle est invisible.

Seconde approche: la transparence alpha

1. Sous **CB**, ⌘ sur «Mat4» et sélectionnez «Duplicate» (ou **Ctrl + W**). Vous savez copier-coller des matériaux maintenant! Renommez-le «Mat5» puis ouvrez le matériau dans l'éditeur.
2. Sélectionnez le nœud «Mat5» et dans **DT**, panel «Material» modifiez «Blend Mode» en «Translucent». L'entrée «Opacity» de «Mat5» est maintenant disponible.
3. Supprimez la texture «T_Tech_Dot_M» et ajoutez la texture «T_Dust_Particle».

Ce n'est pas la meilleure texture pour illustrer cette démonstration, mais il n'y en a pas de meilleure dans le «StarterContent». Nous allons y remédiez.

4. Ajoutez un nœud «Texture Coordinate» à l'entrée «UVs» de la nouvelle texture. Appliquez un Tiling de 0.5x0.5 pour n'avoir qu'une partie de la texture.
5. Reliez à sa sortie principale un nœud «Multiply» pour augmenter l'intensité lumineuse de la texture. Dans **DT**, entrez 10 dans le champ «Const B». Pour voir ce que cela donne, vous pouvez faire un preview sur le nœud «Multiply».
6. Reliez la sortie du nœud «Multiply» à l'entrée «Opacity» de «Mat5».

Cette fois-ci, il y a plusieurs niveaux de transparence en fonction de l'intensité lumineuse des éléments composant la texture qui sert de couche alpha. Nous aurions pu également appliquer une constante (ex: 0.5) qui aurait rendu la texture semi-transparente dans sa globalité.

MATÉRIAUX ÉMISSIFS

Un matériau est capable d'**émettre de la lumière**. Ainsi, il vous sera possible de simuler une rampe lumineuse, un **néon** ou tout autre effet de type «**Tron**». Nous allons reprendre notre scène et créer un simple néon:

1. Tout comme vous avez créé précédemment un bouton de lumière, créez un néon: redimensionnez en «X:4», «Y:0.5» et «Z:0.1». Puis, nommez-le «Neon» et placez-le au plafond.
2. Créez un nouveau matériau «**TubeNeon**» et appliquez-le à l'objet «Neon».
3. Ouvrez le matériau en édition.
4. Ajoutez un nœud «**Constant3Vector**» et choisissez une couleur bleue.
5. En partant de la sortie principale du nœud précédent, ajoutez un nœud «**Multiply**». Dans **DT**, passez «**Const_B**» à 30. Puis reliez la sortie du nœud à l'entrée «**Emissive Color**» de «**TubeNeon**». Dans le preview, vous avez déjà une idée de la façon dont la lumière se propage.
6. Sauvez, revenez à l'éditeur de niveau et lancez un «**Build**» pour recalculer les lumières, puis **Testez**.

S'il vous est possible d'être ébloui par la lumière et de voir qu'elle semble se propager, vous ne voyez aucun impact sur les objets environnants. C'est normal, il nous faut procéder à une modification pour cela!

7. Sélectionnez l'objet «**TubeNeon**», puis dans **DT**, panel «**Transform**», cliquez sur «**Static**».
8. Puis, dans le panel «**Lighting**», cochez «**Use Emissive for Static Lighting**» et passez «**Diffuse Boost**» à 5.
9. Lancez de nouveau un «**Build**» pour recalculer les lumières, puis testez à nouveau.

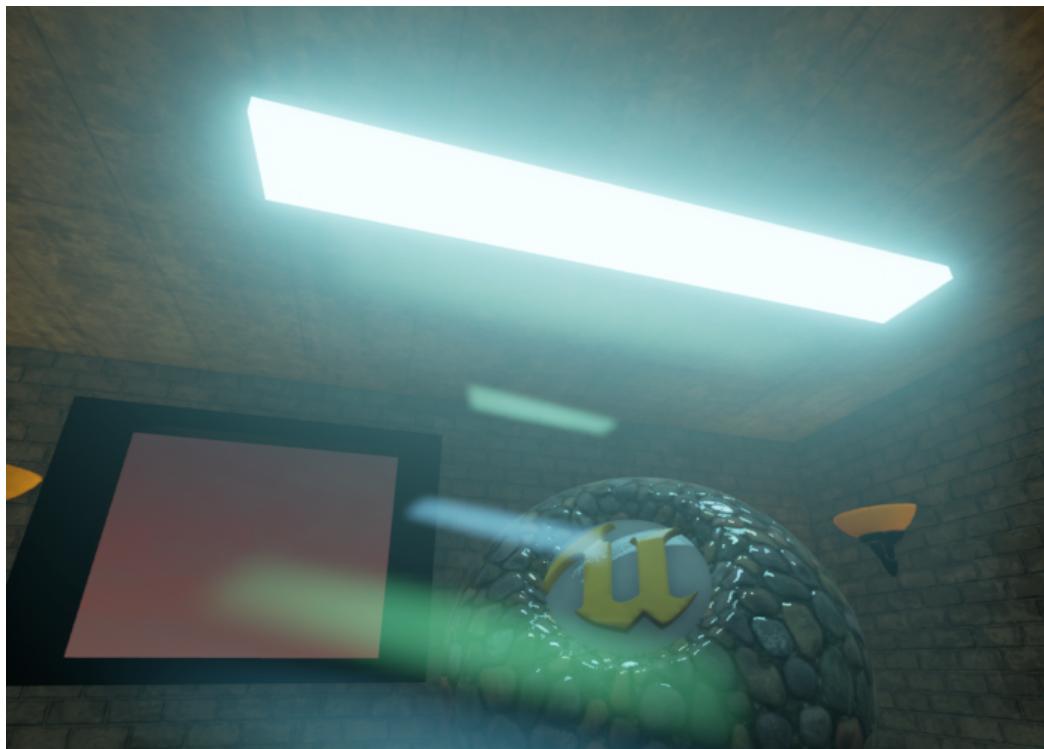


Illustration 77: Crédit à l'origine de la capture d'écran: <https://www.unrealengine.com/tutorials/materials-emissive-materials>

Cette fois-ci, il y a bien une couleur bleue qui se mêle aux objets. Notre néon a donc bien éclairé son environnement comme le montre l'illustration 77

Pourquoi avoir été obligé de passer en «**Static**» ?

«Emissive Color» n'agit pas comme une **lumière dynamique**. En réalité, s'il est possible d'éclairer les objets environnants, il s'agit d'un pré-calcul qui passe par les **lightmaps**. L'effet est réussi, mais la conséquence est qu'il devient non seulement impossible de déplacer la source de lumière, mais également d'éclairer des objets en mouvement. Il n'est pas non plus possible d'éteindre cette lumière directement (il faut disposer d'une seconde lightmap pour effectuer cette opération). Pour faire le test, prenez un objet qui se trouve à proximité du néon et passez-le en mobilité «dynamique». Vous verrez que le néon n'a plus aucun impact sur ce dernier.

MATÉRIAUX ANIMÉS

Il est possible de déplacer les coordonnées UV d'une texture afin de la faire **défiler de droite à gauche ou de haut en bas**. C'est une technique qui était utilisée pour simuler un courant d'eau de façon assez simpliste.

Voici comment procéder:

1. Créez un nouveau matériau «**MatAnim**» et ouvrez l'éditeur de matériau.
2. Ajoutez n'importe quelle texture comme entrée «Base Color» de «**MatAnim**».
3. Puis, à l'entrée «**UVs**» de cette texture, ajoutez un nœud «**Panner**».
4. Sélectionnez «**Panner**» et dans **DT**, panel «**Material Expression Panner**», modifiez «**Speed X**» en 0.1
5. Enfin à l'entrée «**Coordinate**» de Panner, ajouter un nœud «**Texture Coordinate**»

Que constatez-vous? La texture se déplace lentement de la droite vers la gauche donnant l'impression de glisser sur l'objet. Il est possible de réaliser le même effet de haut en bas en modifiant «**Speed Y**». Une valeur négative inverse le sens de la course.

Notez qu'il est possible d'utiliser également la fonction «**Rotator**» pour effectuer une rotation sur la texture.

MATÉRIAUX DYNAMIQUES

Il y a de très nombreuses fonctions qui peuvent être appliquées aux matériaux.

Examinons l'une d'elles: **ActorPositionWS**. Appliqué à l'entrée «Base Color» d'un Matériau, avec un nœud «**Divide**» de 3000 par exemple, cela nous donne le résultat de l'illustration 78. Il s'agit d'un modèle de pyramide du répertoire «shape» du «Starter Content». En fonction de la position de l'actor, la couleur de base change. Mais ActorPosition peut être utilisé de bien d'autres manières.

Nous pouvons notamment faire appel aux fonctions suivantes:

- **CameraPositionWS**: En fonction de la position de la caméra, la couleur change (rien à voir avec la distance de l'actor à la caméra)
- **ObjectOrientation**: En fonction de l'orientation de l'objet
- **ObjectRadius**: En fonction de la dimension de l'objet
- **PixelNormalWS**: S'applique à chaque point du matériau en fonction de la normale à l'objet (appliqué sur une sphère, donne un résultat multicolore)
- **WorldPosition**: S'applique à chaque point du matériau en fonction de la position du point dans l'espace (objet multicolore également)

Ce ne sont pas les seules fonctions. Elles ne servent pas seulement à faire varier la couleur. Nous avons utilisé «bêtement» ces fonctions pour avoir une idée de leur utilisation, mais ce sont des valeurs qui sont renvoyées par ces dernières et qui peuvent être traitées par le programme contenu dans le matériau.

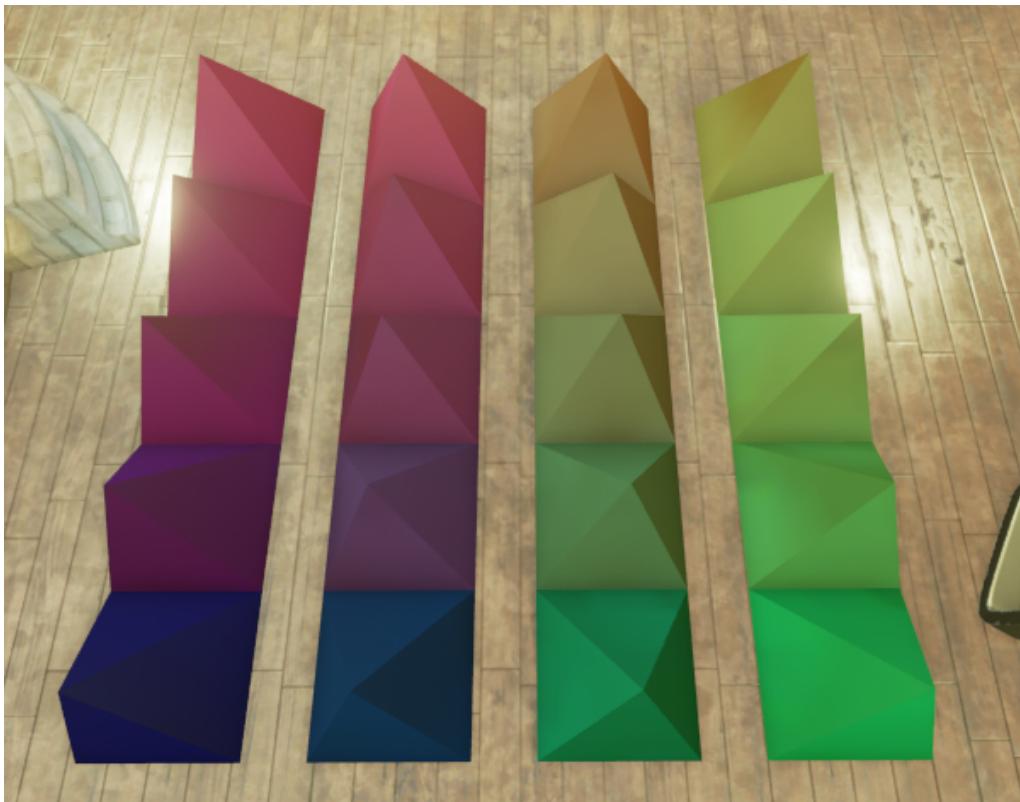


Illustration 78: Changement dynamique de la couleur du matériau en fonction de l'emplacement de l'acteur

QUELQUES MOTS SUR LES AUTRES MATERIAUX

Pour se rendre compte de l'étendue des possibilités offertes par UE4 en terme de matériaux, nous vous proposons une petite exploration:

1. Ouvrez le projet «ContentExamples».

Il est disponible en téléchargement dans la rubrique «Learn» de l'interface de lancement.

2. Ouvrez le level «Material_Nodes» et lancez en cliquant sur «Play»
3. Déplacez-vous en observant sur votre gauche les différents matériaux. Vous avez appris à concevoir la plupart des matériaux présentés.
4. Rendez-vous à la section «1.10 World Offset».

Le matériau est capable de **déformer l'objet de façon dynamique**. L'illustration 79 donne un aperçu de l'utilisation de cette fonction. Pour plus d'informations, nous vous conseillons d'étudier les exemples fournis dans ce level de présentation.

5. Continuons notre visite et allons à la section «1.11 World Displacement».

Vous vous rappelez du **normal mapping**: il permet de donner l'impression que la texture a un certain volume en jouant sur la façon dont la lumière interagit avec cette dernière. En aucun cas la surface de l'objet n'est modifiée. Dans le cas présent, en utilisant **World Displacement**, on modifie réellement la surface de l'objet! C'est pour cette raison que nous l'utilisons en activant la fonction «**Tessellation**» (disponible avec **DirectX 11/+**) qui permet de subdiviser les faces de l'objet pour lui ajouter un maillage plus fin qui pourra être déformé.

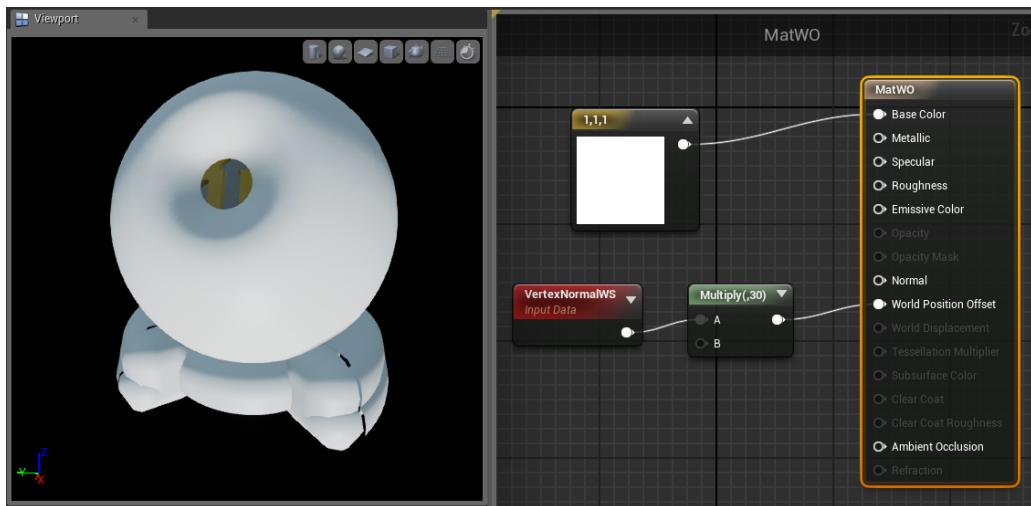


Illustration 79: Utilisation de l'entrée World Offset du Matériau

L'illustration 80 donne un exemple d'utilisation de cette fonction. Nous avons utilisé les textures «T_Cobblestone_Peeble» qui possèdent une version adaptée (extension «M»). Dans cet exemple, nous gérions manuellement le niveau de Tessellation en décochant «**Adaptive Tessellation**» et en ajoutant une constante liée à l'entrée «**Tessellation Multiplier**». Vous pouvez jouer sur le multiplicateur (ici 35) pour rendre l'effet plus ou moins important.

Ici, nous avons désactivé le mode automatique de la tessellation. Toutefois, dans une utilisation normale, nous ne vous le conseillons pas. En effet, UE4 est suffisamment optimisé pour déterminer quand et quel multiplicateur utiliser en fonction de la distance de la caméra à l'objet. C'est une partie de la gestion du LOD (Level of Detail).

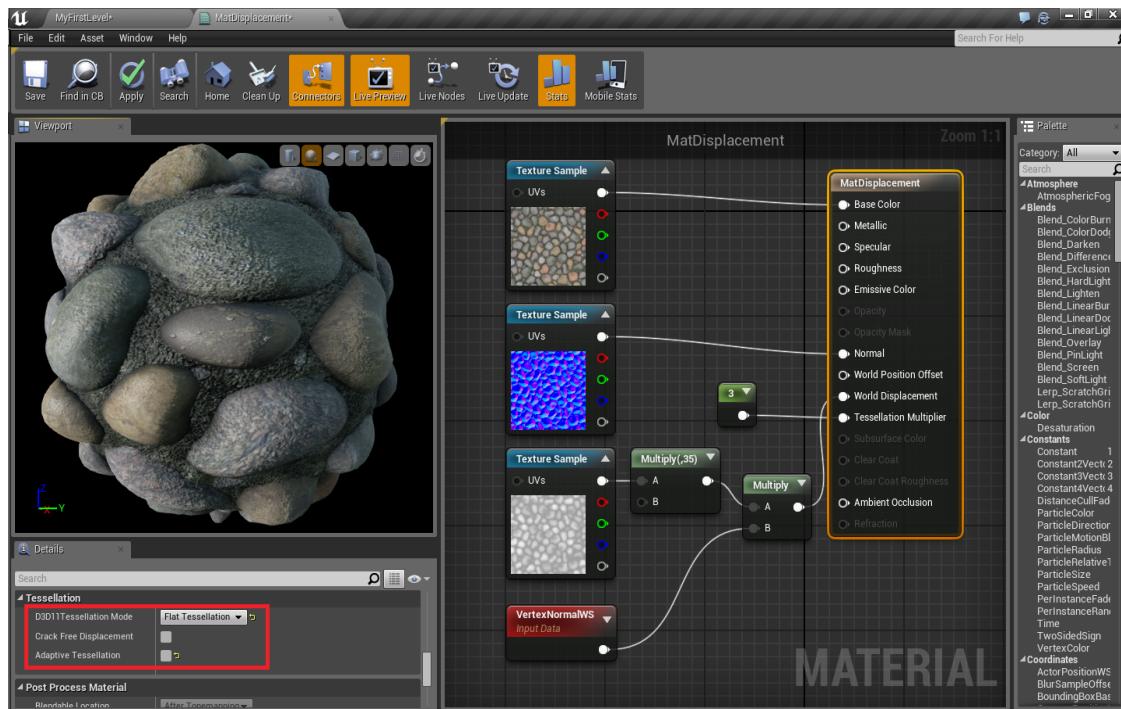


Illustration 80: Matériau avec World Displacement



Le **LOD (Level of Detail)** est une technique qui définit le **niveau de détail** d'un objet suivant la distance qui le sépare de la caméra. Plus un objet est éloigné, et moins l'objet a besoin de détails. Cette technique permet d'**optimiser la gestion de grands espaces** en allégeant ainsi le rendu de la carte graphique. C'est donc une technique qui, dans les moteurs modernes, utilise le niveau de **tessellation** des objets pour augmenter ou diminuer le niveau de détail. Dans les moteurs plus anciens, on stockait parfois plusieurs niveaux de détails de l'objet sous forme pré-calculée. Mais plusieurs autres méthodes existent, comme remplacer les objets très éloignés par des «**billboards**», c'est à dire des images (sprites) faisant toujours face à la caméra.

6. Continuons notre visite et allons à la section «**1.13 Subsurface Color**».

Pour utiliser le subsurface, il suffit d'utiliser l'entrée «Subsurface Color» d'un matériaux en ayant sélectionné préalablement «Subsurface» dans «Lighting Mode» du panel «Material» de **DT**.



Le «**subsurface scattering** » (**SSS**) est une technique utilisée pour faire «sortir» la **couleur intérieure** d'un objet soumis à une lumière. Quand vous déplacez la flamme d'une bougie derrière votre main, une couleur «rougeâtre» semble jaillir de votre main. Votre peau est en quelque sorte semi-transparente. Mais quelque soit la couleur extérieure de votre peau, votre sang est rouge (sinon, consultez un médecin de toute urgence). Vous pouvez obtenir le même résultat en utilisant cette technique.

7. Enfin, terminons notre visite par la section «**1.14 Refraction**».

Si vous souhaitez créer un matériau de type «verre», il ne suffit pas de le rendre transparent. Imaginons que vous souhaitez appliquer ce matériau sur un vase. Si ce vase possède une certaine épaisseur, tous les objets passant derrière ce dernier doivent paraître déformés.

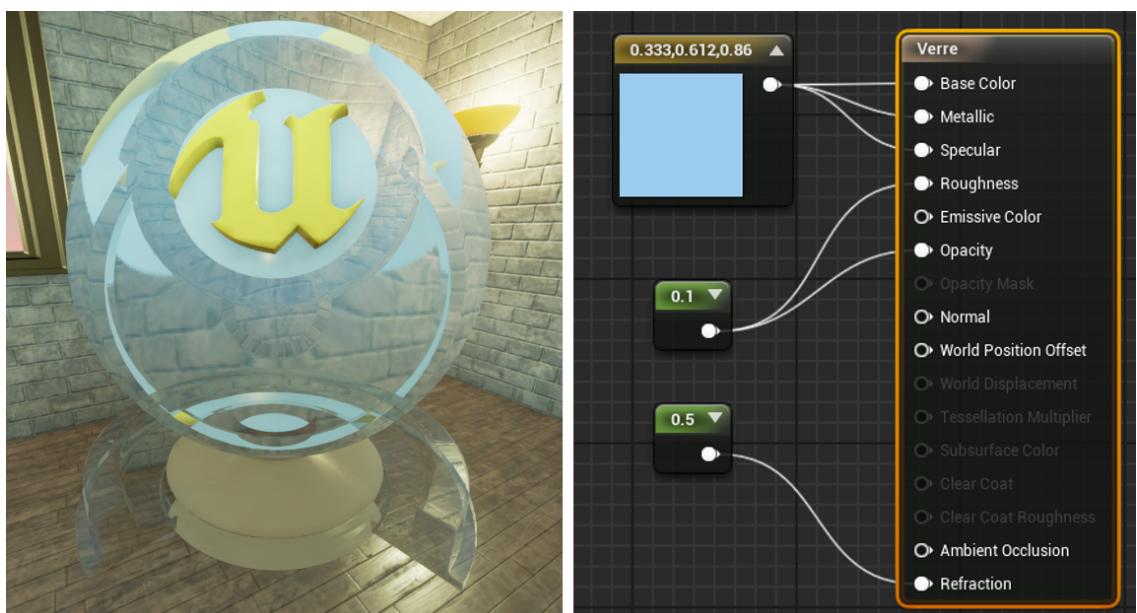


Illustration 81: Modification de l'indice de réfraction d'un matériau

Cette déformation (effet loupe) peut être réalisée en modifiant le coefficient de réfraction de la surface. A partir du moment où vous travaillez sur un matériau transparent, l'entrée «**Refraction**» du matériau est utilisable.

L'illustration 81 nous montre un exemple de matériau de type «Verre» appliqué sur le modèle de démonstration des matériaux d'UE4.

Voilà, nous avons vu le paramétrage de base des matériaux, mais il reste de très nombreuses possibilités. Vous pourriez étudier les matériaux livrés avec le «Starter Content», mais nous ne vous le conseillons pas forcément. Il s'agit de matériaux très travaillés, difficiles à décrypter.

FONCTION DE MATÉRIAUX

La palette de fonctions disponibles dans l'éditeur de matériaux n'est pas limitée aux fonctions internes d'UE4. Vous pouvez créer vos propres fonctions.

Dans l'exemple suivant, nous allons créer une fonction «**Wind**» qui simule l'effet du vent, puis nous modifierons le matériau «IvyWall» que nous avons créé avant pour utiliser cette fonction:

1. Placez-vous dans votre répertoire «Material» et cliquez sur «Create», puis allez dans «Materials & Textures» et enfin, ⚡ sur «**Material Function**». Renommez-le en «**WindEffect**», puis ouvrez-le en mode édition

C'est bien dans l'éditeur de matériaux que s'ouvre la nouvelle fonction. Au lieu d'avoir le nœud du matériau au centre du graphe, nous avons juste le nœud «output» de sortie de la fonction.

2. Ajoutez les divers éléments au graphe comme dans l'illustration 82

Pour la fonction «panner», prenez «Speed X: 5» et «Speed Y: 1» par exemple.

«Input power» est de type «**FunctionInput**». Une fois déposée sur le graphe, vous pouvez modifier son type de données en allant dans **DT**, «Input Type» et en sélectionnant «Function Input Vector2».

«**FunctionInput**» permet d'ajouter des paramètres d'entrée à la fonction. Ainsi, nous utilisons cette variable pour gérer la puissance de l'oscillation. Vous notez qu'il existe plusieurs types de variables possibles. Ici, nous souhaitons juste passer 2 valeurs, une pour chaque axe.

Le vecteur (10,10) est ici passé en exemple pour que la fenêtre de preview puisse donner un aperçu, mais il n'est absolument pas essentiel pour que la fonction soit opérationnelle.

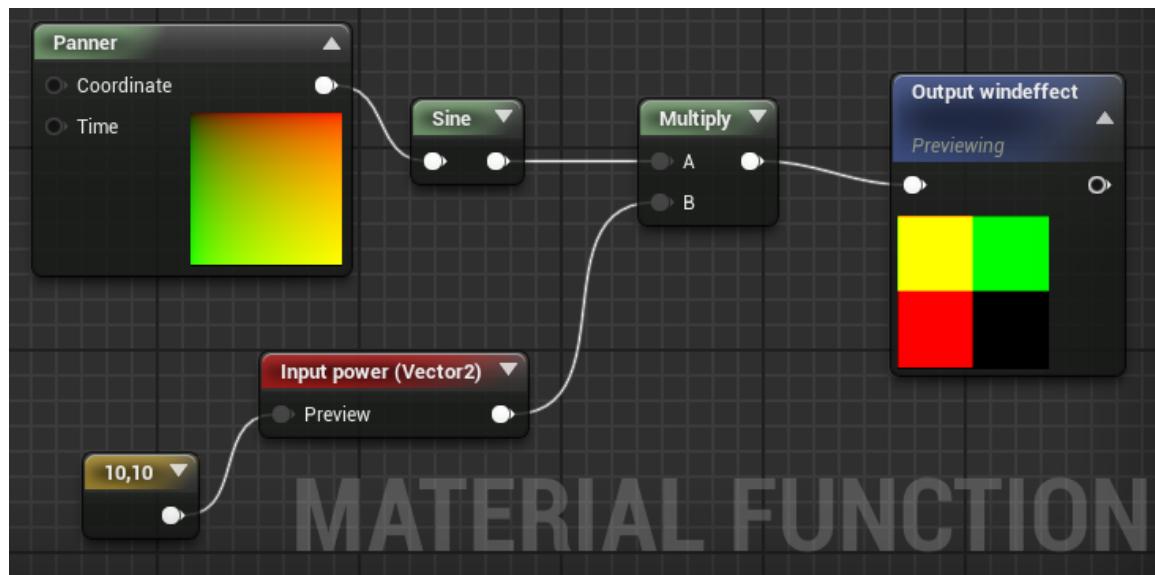


Illustration 82: Fonction matériau permettant de simuler un effet de vent

3. Cliquez sur le canevas (pas sur un nœud) pour que **DT** affiche les détails de la fonction comme sur l'illustration 83

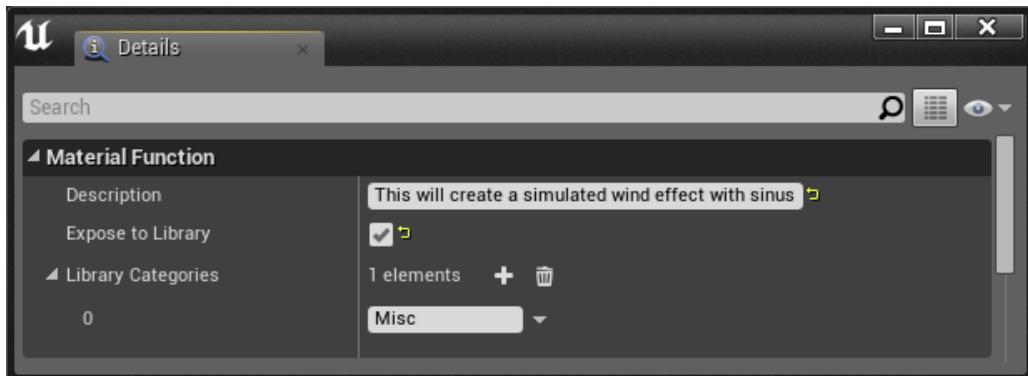


Illustration 83: Détail de la fonction

4. Dans la description, entrez une phrase décrivant l'objet de la fonction
 5. Cochez «**Expose to Library**» pour que la fonction soit disponible dans la «palette»
 6. Développez «Library Categories» si ce n'est pas encore fait et sélectionnez une catégorie ou laissez «Misc».

Vous avez remarqué que les fonction de la «palette» sont organisées par catégories. Vous pouvez ici sélectionner une catégorie connue, ou créer votre propre catégorie.

7. Sauvez et revenez à l'éditeur de niveau

Nous allons maintenant créer un matériau pour le drapeau et utiliser la fonction «**WindEffect**»:

8. Créez un nouveau matériau «**Flag**» et passez en mode édition
 9. Ajoutez les éléments au graphe comme dans l'illustration 84

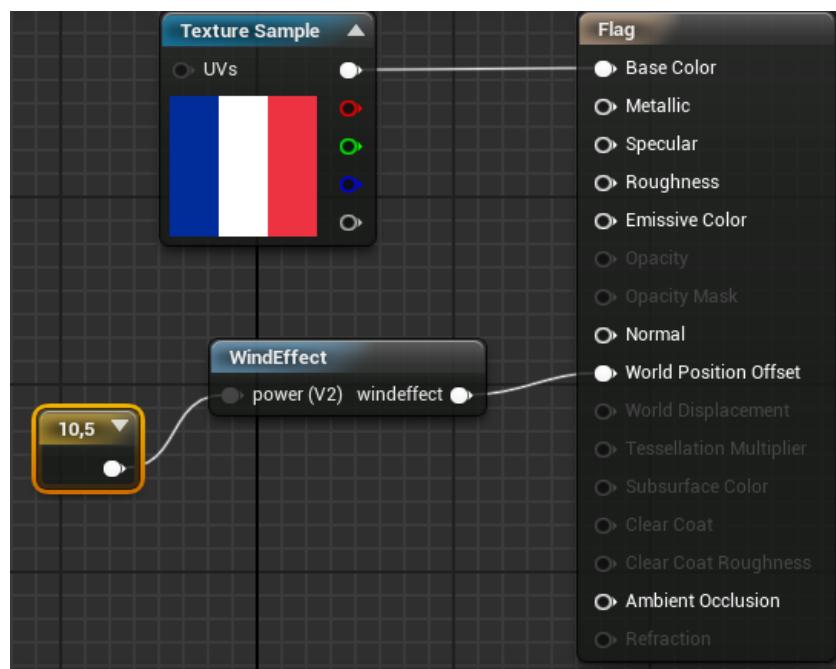


Illustration 84: Exemple d'utilisation de la fonction créée juste avant

(10,5) est un «Constant2Vector» qui permet de passer les paramètres 10 et 5 à WindEffect. Notre fonction n'agit que sur «World Position Offset», c'est à dire qu'il opère une déformation de l'objet comme vu précédemment.

- 10. Rendez le matériau double face en cochant la case «double side» du nœud «flag».
- 11. Sauvez et revenez à l'éditeur de niveau

Maintenant, il ne reste plus qu'à tester le matériau sur un plan. Vous disposez d'un tel asset dans le répertoire «Shapes» du «Starter Content».

Nous avons utilisé cette fonction sur un drapeau, mais vous pouvez de la même façon ajouter du vent dans les feuilles d'un arbre ou sur un brin d'herbe.

INSTANCE DE MATÉRIAU

Jusqu'à présent nous n'avons créé que des matériaux non paramétrables. Il fallait les appliquer directement sur un objet et éventuellement les modifier pour obtenir des changements.

Il y a une autre approche: **créer un modèle de matériau avec quelques paramètres modifiables en entrée et appliquer des instances de ce modèle à des objets**.

Voici comment procéder:

- 1. Créez un nouveau matériau «**MatModel**» et ouvrez l'éditeur de matériau.
- 2. Ajoutez la texture «T_Brick_Clay_Beveled_D»
- 3. Dans la «palette», panel «Parameters», sélectionnez «**VectorParameter**» et ajoutez le nœud sur le canevas. Renommez en «basecolor». Choisissez une couleur «bleue».
- 4. Reliez les sorties des nœuds «Texture Sample» et «color» à un nœud «Multiply».
- 5. Reliez ensuite la sortie «Multiply» à l'entrée «Base Color» de «**MatModel**».

Le modèle est prêt. Chaque instance du modèle peut prendre en compte un nouveau paramètre, «basecolor».

Pour l'utiliser:

- 6. Revenez dans l'éditeur de niveau et dans **CB**,  sur «**MatModel**» puis sélectionnez «Create Material Instance». Nommez-le «**MatInstance1**».
- 7. Ouvrez «**MatInstance1**» avec l'éditeur de matériaux et vous obtenez une nouvelle fenêtre, comme dans l'illustration 85.
- 8. Dans **DT**, on retrouve notre paramètre «basecolor». Il suffit de cocher la case pour pouvoir modifier la couleur d'origine. Dans l'illustration, nous avons choisi un jaune.
- 9. Sauvez, puis appliquez l'instance «**MatInstance1**» à un objet.

Nous voyons tout de suite qu'il est possible par cette méthode de créer assez rapidement une déclinaison de plusieurs matériaux. Mais il y a aussi un autre avantage:

- 10. Revenez sur «**MatModel**» et remplacez la texture «T_Brick_Clay_Beveled_D» par une autre.
- 11. Sauvez et revenez à l'éditeur de niveau.

Que constatez-vous? L'objet sur lequel nous avons appliqué «**MatInstance1**» a bien pris en compte les modifications que nous venons de réaliser. **Les matériaux instanciés bénéficient de toutes les modifications apportées aux matériaux qui leur ont servi de modèle. Le lien est bien dynamique.**

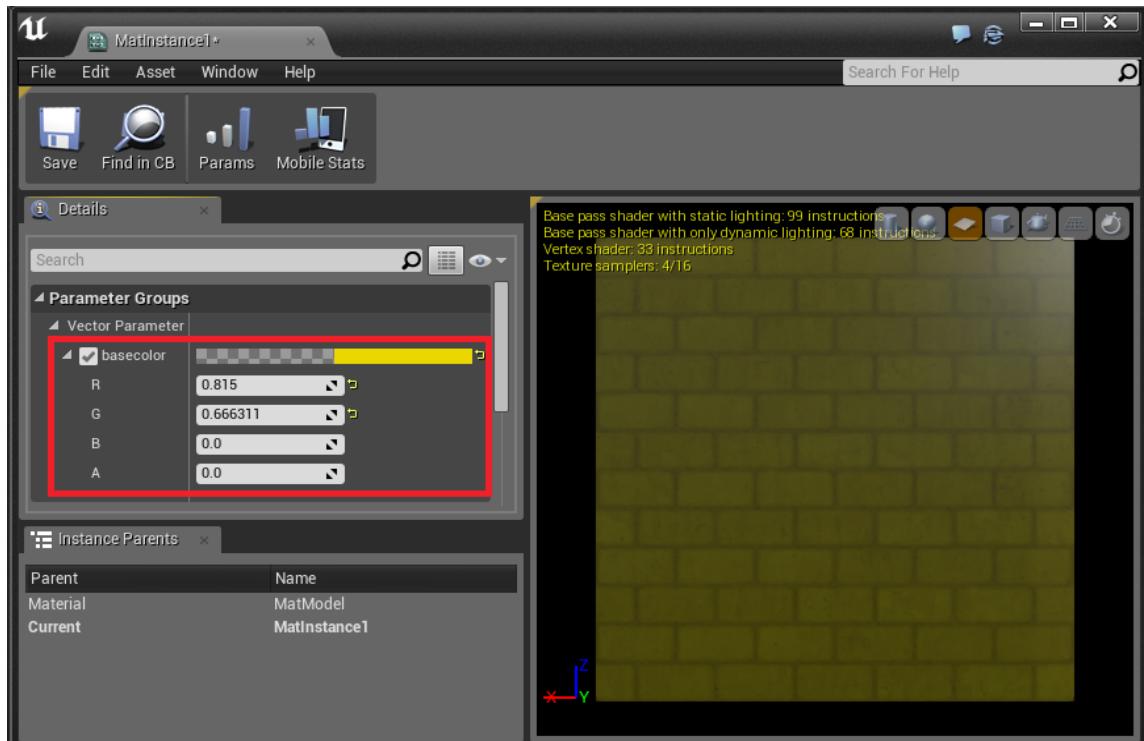


Illustration 85: Modification d'une instance de matériau

MISE EN PLACE DU PAYSAGE



4

UE4 est livré avec un outil exceptionnel permettant de créer des paysages très réalistes. Il est bien entendu possible de faire appel à des outils tiers, générant ainsi une **heightmap** ré-exploitable sous UE4, ou tout simplement en important le paysage comme un modèle 3D (mais on risque ainsi de se couper des optimisations du moteur permettant de gérer les grands espaces).



Une **Heightmap** est image représentant une carte de «hauteur» (topographie) utilisée pour générer un terrain en relief. Chaque pixel est utilisé pour déformer une grille 3D et lui donner de la hauteur. En général, elles sont en niveau de gris et plus un pixel est proche du blanc, plus le point correspondant possède une hauteur élevée. Ainsi, il suffit de la générer aléatoirement (bruit, **diagramme de Voronoï**) ou de la peindre, pour générer un terrain réaliste.

CRÉATION D'UN LANDSCAPE

1. Créez un nouveau matériau «**MatLand**»
2. Dans **MD**, cliquez sur le volet «**Landscape**» (illustration 86). L'outil «**manage**» est celui qui s'active par défaut quand aucun landscape n'est sélectionné.
3. Sélectionnez le matériau «**MatLand**» dans «**Material**»,
4. Puis dans «**Section Size**» passez la valeur à «**31x31 Quads**».

Si votre ordinateur est modeste, vous pouvez utiliser une section plus petite et redimensionner: le maillage du terrain sera moins fin, mais il est possible d'obtenir de très bons résultats tout de même.

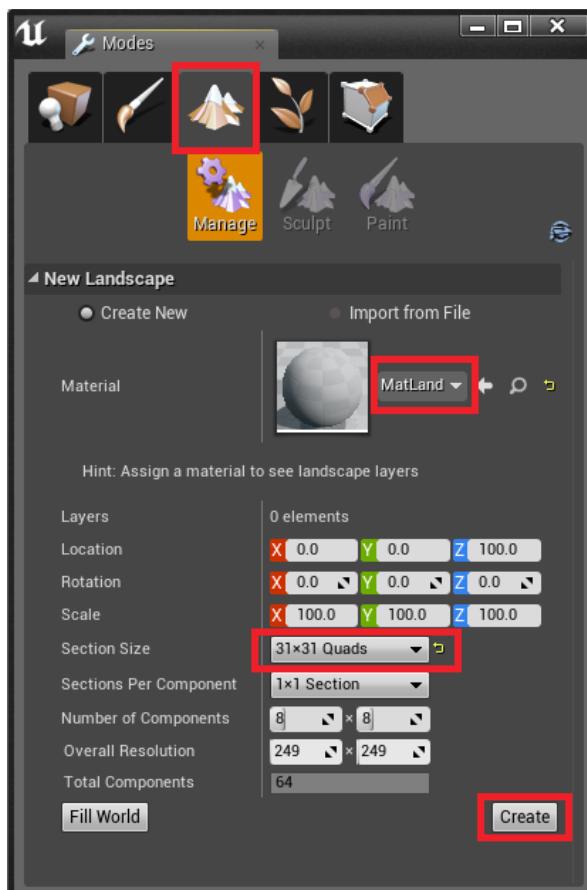


Illustration 86: Création d'un Landscape

5. Et enfin, cliquez sur «Create».

Deux nouveaux actors apparaissent dans **WO**:

- le **landscape** lui-même,
- un autre nommé **«LandscapeGizmoActiveActor»**. Ce dernier n'est présent que dans l'éditeur et il ne sert qu'à la sélection et la modification du landscape.

Quelques mots sur la création du landscape:

- **«Layers»**: notre matériau ne comporte pas de «couche» (layer), c'est à dire plusieurs choix de texture. Mais nous allons y remédier un peu plus loin et vous en expliquer l'utilité.
- **«Location», «Rotation», «Scale»**: il est possible de réaliser les différentes transformations ici, mais cela sera encore possible par la suite.
- Le landscape est organisé en composants (**«Components»**): ici, il y a 8x8 composants. Chaque composant est divisé en sections: ici une seule section. Et chaque section est composée d'un certain nombre d'éléments rectangulaires (Quads).
- **«Section Size»**: Définit la taille d'une section en nombre de Quads. C'est utilisé principalement par le système d'optimisation basé sur le LOD et sur le Culling. Plus les sections sont petites, et plus le LOD est agressif. Cela signifie que le CPU est un peu plus réquisitionné, mais que le GPU est, lui, allégé. Pour créer de grands paysages, il est souhaitable de ne pas augmenter la taille des sections, mais de jouer plutôt sur le redimensionnement du paysage, sinon le CPU est trop sollicité.
- **«Sections Per Component»**: Par défaut, on n'applique qu'un seul niveau de LOD par composant. Mais il est possible de diviser encore le composant en 4 en appliquant **«2x2 Sections»**.
- **«Number of Components»**: Nombre de composants. Modifier cette valeur permet d'agrandir le terrain.
- **«Overall Resolution»**: paramètre non lié au LOD, mais à la résolution du terrain. C'est le nombre de vertices (points 3D) utilisés pour générer le landscape. Une modification de la résolution entraîne des modifications dans les sections précédentes.
- **«Fill the world»**: permet d'agrandir le terrain à sa valeur maximale

Overall size (vertices)	Quads / section	Sections / component	Component size	Total Components
8129x8129	127	4 (2x2)	254x254	1024 (32x32)
4033x4033	63	4 (2x2)	126x126	1024 (32x32)
2017x2017	63	4 (2x2)	126x126	256 (16x16)
1009x1009	63	4 (2x2)	126x126	64 (8x8)
1009x1009	63	1	63x63	256 (16x16)
505x505	63	4 (2x2)	126x126	16 (4x4)
505x505	63	1	63x63	64 (8x8)
253x253	63	4 (2x2)	126x126	4 (2x2)
253x253	63	1	63x63	16 (4x4)
127x127	63	4 (2x2)	126x126	1
127x127	63	1	63x63	4 (2x2)

Illustration 87: Tailles recommandées par UE4 pour la génération de terrains



Le «**CPU**» est le processeur central de l'ordinateur. Notez que les ordinateurs modernes disposent de plusieurs CPU, ayant eux-même plusieurs coeurs. Le CPU est sur la carte mère de l'ordinateur et il est considéré comme le cerveau de la machine. Les tâches graphiques sont toutefois confiées à un autre processeur, le **GPU**. Si le GPU est trop sollicité, le nombre de FPS (images par seconde) va chuter. Toutefois, si le CPU est trop sollicité, c'est le jeu complet qui va ralentir fortement. Il y a souvent un équilibre à trouver.

Le «**Culling**» est un procédé consistant à masquer certains objets ou certaines parties d'un objet pour des raisons d'optimisation. Par exemple, si un mur se trouve entre des objets et la caméra, alors il devient inutile de traiter ces objets. Il y a plusieurs formes de culling. Le «**Backface Culling**» consiste à masquer les faces d'un objet qui sont cachées par les autres d'un même objet. Si vous observez une sphère par exemple, la moitié des faces, celles qui se situent derrière l'objet, n'a aucune raison d'être traitée.

Il est possible d'ajouter un autre **heightmap** créé à partir d'un programme externe en utilisant la fonction «**Import From File**».

Apportons quelques modifications à notre paysage:

- 6. Sélectionnez «Landscape» et restez dans la section «**Manage**» du volet «Landscape» de **MD**
- 7. Cliquez sur «**Selection Tool**» et utilisez les outils «Add» et «Delete» pour modifier le landscape tel que sur l'illustration 88.

La forme n'a pas besoin d'être respectée à la lettre, l'objectif est de vous faire manipuler ces outils.

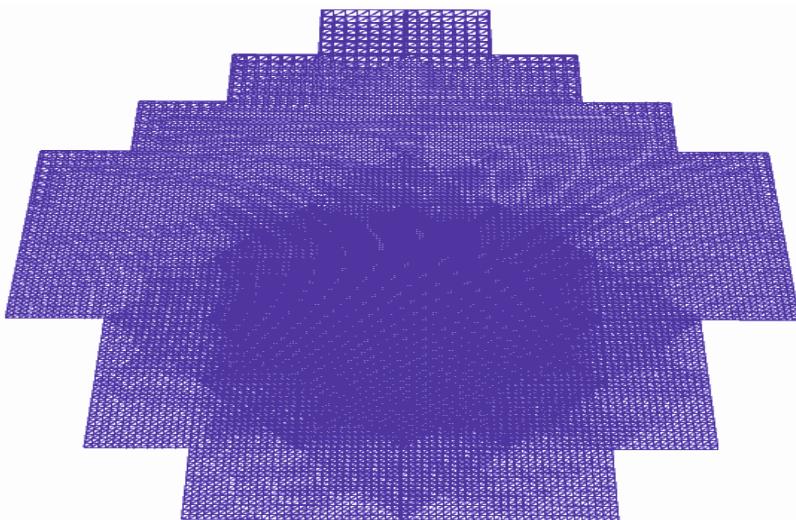


Illustration 88: Ajout et suppression de composants

«**Brush Size**» peut-être utilisée pour agrandir le nombre de composants que l'on peut ajouter ou supprimer en même temps (ex: 2 pour «2x2 composants»).

Notez la présence de **3 autres outils**:

- «**Move To Level**»: permet de déplacer une partie du paysage dans le level persistent. Nous reviendrons un peu plus tard sur le «level streaming».
- «**Change Component Size**»: pour modifier les caractéristiques de notre paysage (celles que nous avons paramétrées à la création).
- «**Edit Splines**»: pour utiliser des courbes, nous utiliserons cet outil un peu plus loin également.

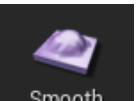
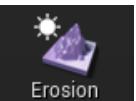
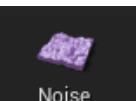
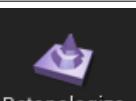
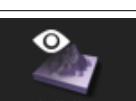
Et pour finir:

- 8. Déplacez le paysage au complet selon l'axe Z pour arriver au niveau des marches de l'escalier.

MODELER LE PAYSAGES

Les différents outils et brosses

Nous arrivons à la partie la plus amusante, celle qui consiste à sculpter le terrain comme s'il était fait d'argile. Dans **MD**, volet «**Landscape**», section «**Sculpt**», plusieurs outils sont disponibles pour cela:

	Outil « Sculpt »: permet de surélever le terrain en utilisant un clic gauche ou d'abaisser le terrain en utilisant « Shift + clic gauche ». L'icône est en surbrillance quand l'outil est actif.
	Outil « Smooth »: permet de lisser le terrain en homogénéisant les surfaces trop ciselées. Cela induit un «adoucissement» du terrain.
	Outil « Flatten »: permet de créer des zones de plat. Il suffit de partir de la zone de hauteur choisie et de balayer progressivement autour pour amener les parties voisines à la même hauteur.
	Outil « Ramp »: pour créer un chemin «plat» entre deux points du paysage, en suivant une ligne droite. Pour créer des chemins incurvés, il faut utiliser un autre outil: l'éditeur de «splines» que nous verrons un peu plus loin.
	Outil « Erosion »: pour simuler une érosion thermique en ajustant la hauteur du terrain, transférant la matière des niveaux hauts vers le bas. L'outil permet de creuser facilement des couloirs. Plus il y a une différence de hauteur, plus l'érosion est importante. Cet outil applique également un effet de bruit, au besoin, pour donner un aspect aléatoire plus naturel. « Threshold » permet de modifier la différence de hauteur minimum nécessaire pour que les effets de l'érosion ne s'appliquent. Une valeur plus petite se traduit par une érosion plus importante.
	Outil « Hydro Erosion »: Pour simuler une érosion pluviale. Un filtre de bruit est utilisé pour déterminer où la pluie initiale est distribuée. Puis la simulation est calculée pour déterminer le débit initial de l'eau de cette pluie ainsi que la dissolution, le transfert de l'eau, et l'évaporation. Le résultat de ce calcul fournit la valeur réelle utilisée pour abaisser le heightmap. Pour creuser des chutes d'eau: utilisez l'outil «Erosion» pour créer les canaux principaux, et «Hydro Erosion» pour lisser l'ensemble et donner un visuel plus réaliste.
	Outil « Noise »: pour ajouter du «bruit», c'est à dire des variations aléatoires de hauteur et rendre la surface plus réaliste.
	Outil « Retopologize »: ajuste automatiquement les sommets du maillage du Paysage avec un X / Y compensé pour améliorer la densité de sommets sur les falaises, ce qui réduit l'étirement de la texture. A n'utiliser que si vous avez des étirements pas très jolis de texture car cela a tendance à ralentir l'affichage du terrain.
	Outil « Visibilité »: pour créer des trous dans le paysage. Il ne s'agit pas de creuser, mais de permettre l'utilisation du second matériau spécifique aux «holes» qui créent une zone transparente à l'endroit de la grotte. En effet, nous travaillons sur une heightmap, il n'est donc pas possible de créer des grottes horizontales dans le paysage. Il vous faudra les modéliser de façon externe et les ajouter comme des actors dans le terrain. Avec l'outil «visibility», vous pourrez «gommer» l'entrée de la grotte sur le terrain.

Vous allez tenter de **reproduire dans les grandes lignes** le paysage de l'illustration 89 en n'utilisant que l'outil **sculpt**:

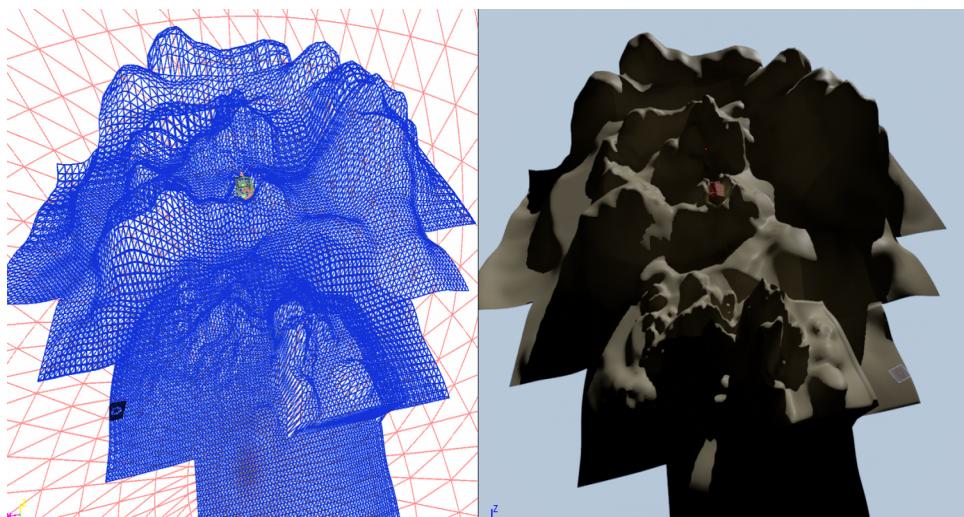


Illustration 89: Exemple de modélisation du terrain en utilisant l'outil "Sculpt"

Passons à la pratique:

1. Passez en mode «**sculpt**» et sélectionner l'outil «**sculpt**».
2. Dans le panel «**Brush Settings**», passez «**Brush Size**» à «**1024**» et gardez un «**falloff**» de «**0.5**».

Une brosse importante permet de modifier le terrain dans de larges proportions. Pour commencer à modeler le terrain, utilisez un brosse relativement large pour sculpter dans les grandes lignes.

La brosse est symbolisée par un premier cercle extérieur. Le second cercle intérieur correspond au «Falloff». Le «Falloff» correspond à l'atténuation de l'effet de la brosse. Plus il est important, plus l'effet est adouci.

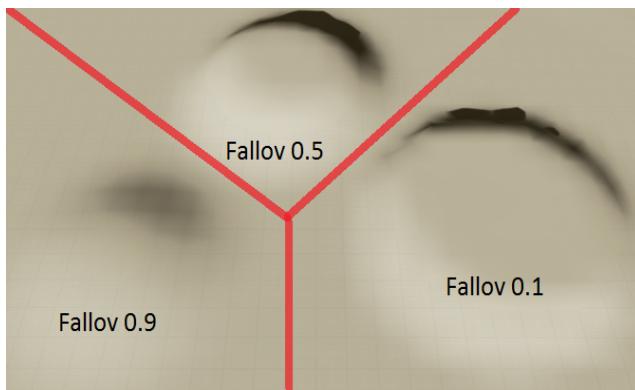


Illustration 90: Les différents effets du falloff sur une brosse sculpt

3. Commencez à balayer le paysage en maintenant le bouton gauche de la souris enfoncé pour éléver des zones et «**Shift**+clic gauche» pour en baisser d'autres.

Si l'effet n'est pas important, ou si au contraire vous souhaitez que le modelage soit **plus énergique**, modifiez le paramètre «**Tool Strength**». Plus il est important, moins vous aurez besoin de passer la brosse pour créer l'effet.

N'hésitez pas à revenir en arrière de temps en temps (undo: **Ctrl+Z** et redo: **Ctrl+Y**).

La brosse par défaut est ronde («**circle brush**»). Mais vous pouvez utiliser d'**autres brosses** comme:

- «**Pattern**»: permet d'utiliser une texture comme motif. L'effet sera plus ou moins prononcé selon l'intensité des pixels de la texture. Ainsi, il est possible de marquer une partie du paysage en de lui donner le relief d'une image. On peut aussi ajouter des détails par ce biais.
 - «**Texture**»: sélection de la texture utilisée comme motif
 - «**Channel**»: choix du canal R,G,B ou Alpha de l'image. On n'utilise pas la couleur même, mais le degré d'intensité.
 - «**Texture scale**»: facteur de redimensionnement de la texture: 0.5 pour utiliser 4 fois la texture ou 2 pour ne prendre que le centre.
 - «**Texture rotation**»: pour faire une rotation de l'image
 - «**Texture Pan (UV)**»: pour déplacer «droite/gauche» et/ou «haut/bas» la texture sur la brosse.
- «**Alpha**»: fonctionne comme «Pattern» mais oriente la texture dans le sens de la peinture.
- «**Component**»: utilise la totalité du composant comme une brosse. Ainsi, combiné avec le sculpt par exemple, c'est la totalité de la surface du composant qui est concernée. «Brush Size» permet alors d'utiliser plusieurs composants simultanément.

Ensuite, nous allons **travailler les détails**:

4. changez la brosse de taille: passez à 512, ajoutez des détails, puis à 256, etc. N'hésitez pas à changer la forme de la brosse, ou encore celle du falloff (illustration 91) pour obtenir l'effet voulu.

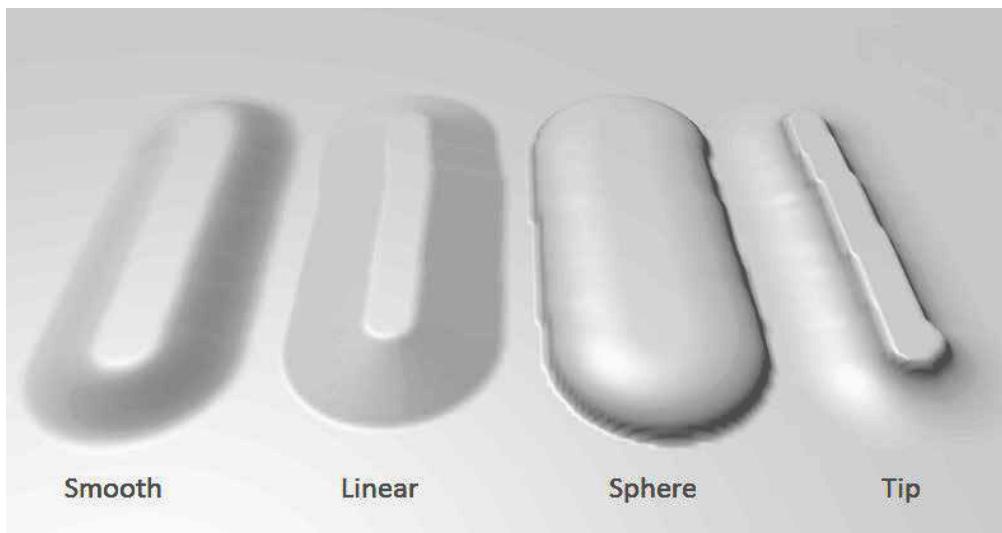


Illustration 91: Différentes formes de falloff

La première phase est réalisée. Nous allons maintenant ajouter quelques **plates-formes** sur lesquelles personnages et véhicules pourront plus facilement se déplacer:

5. Utilisez les outils «**Erosion**» et «**Hydro Erosion**» pour ajouter quelques couloirs sur le flanc des collines.

Ces deux outils sont relativement complexes et possèdent de nombreuses options. C'est en testant différentes valeurs et en les utilisant régulièrement que vous réussirez à vous les approprier: un long discours ne servirait qu'à semer en vous la confusion.

Vous n'êtes pas non plus obligé d'appliquer trop de détails maintenant: lorsque nous aurons «peint» le paysage, il vous sera possible d'ajouter des détails en reprenant un peu la modélisation.

Plates-formes et rampes d'accès

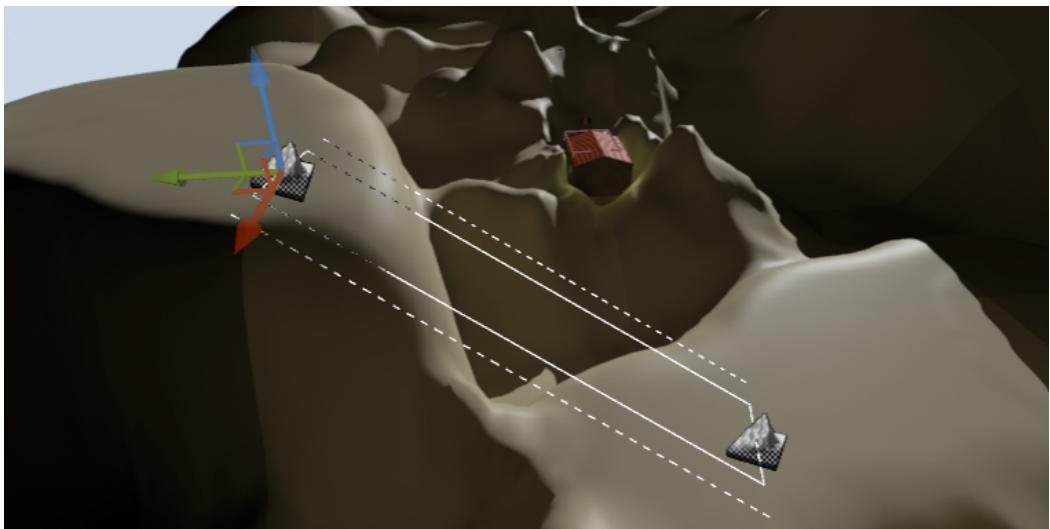


Illustration 92: L'outil "Ramp" en cours d'utilisation

Pour ajouter une **rampe d'accès**:

6. Utilisez l'outil «Flatten» pour réaliser 2 plates-formes comme sur l'illustration 92.

Peu importe où vous placez ces deux plates-formes, l'important c'est qu'il y ait une bonne différence de hauteur entre les deux.

7. Utilisez l'outil «Ramp» pour tracer une route entre les deux plates-formes. Pour cela, cliquez sur la première plate-forme (une première icône apparaît), puis cliquez sur la seconde (une second icône). Dans «Ramp Width», choisissez «1024» pour avoir un chemin suffisamment large.
8. En cliquant sur chaque icône, vous pouvez encore déplacer le point de départ et d'arrivée.
9. Enfin, appuyez sur «Entrée» pour valider: la route va se former comme sur l'illustration 93.

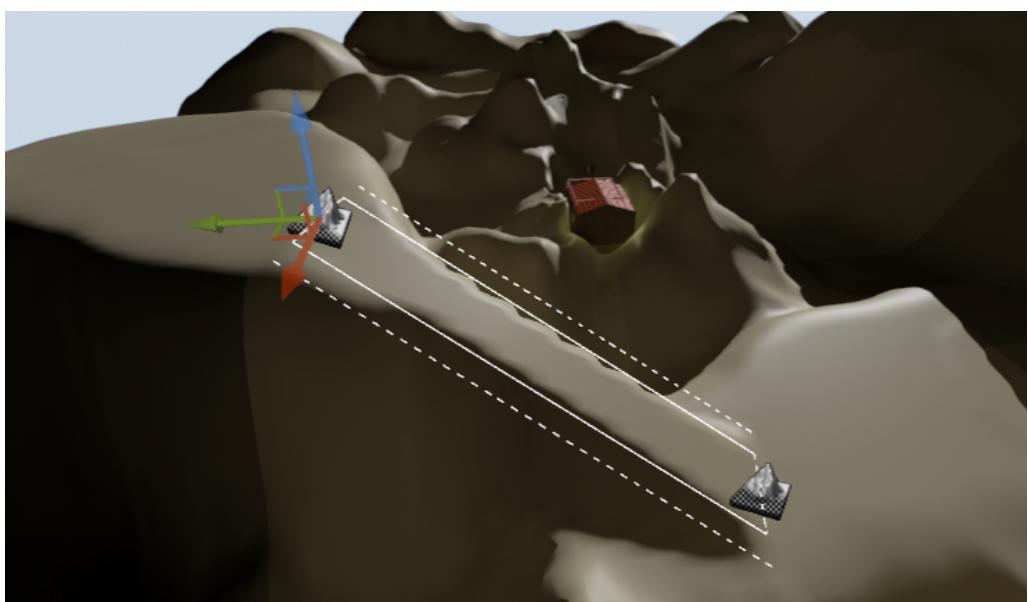


Illustration 93: Formation du chemin après validation de l'outil "ramp"

Tracer une route

Nous avons tous les outils nécessaires pour réaliser le tracé d'une route. Mais imaginez que vous ayez besoin d'agrandir une montagne, ou de déplacer un canyon... faut-il recommencer tout le travail sur les chemins? Si avez créé le chemin «à la mimine», la réponse est malheureusement oui!

Heureusement, un outil est fourni par UE4 pour tracer automatiquement ces chemins en utilisant des courbes de Bézier, ainsi que des points d'étape. C'est très pratique pour créer un circuit de course de voitures dans la montagne, mais pas seulement! Tout chemin peut-être réalisé en utilisant cette technique. D'ailleurs, c'est ce que nous allons faire pour notre jeu.

Voici comment procéder:

10. Allez dans le volet «Manage» et sélectionnez l'outil «Edit Splines».

11. Prenez une vue aérienne de l'ensemble du terrain et «**Ctrl**+clic gauche» sur un point du terrain comme dans l'illustration 94 (cercle bleu pour le premier point).

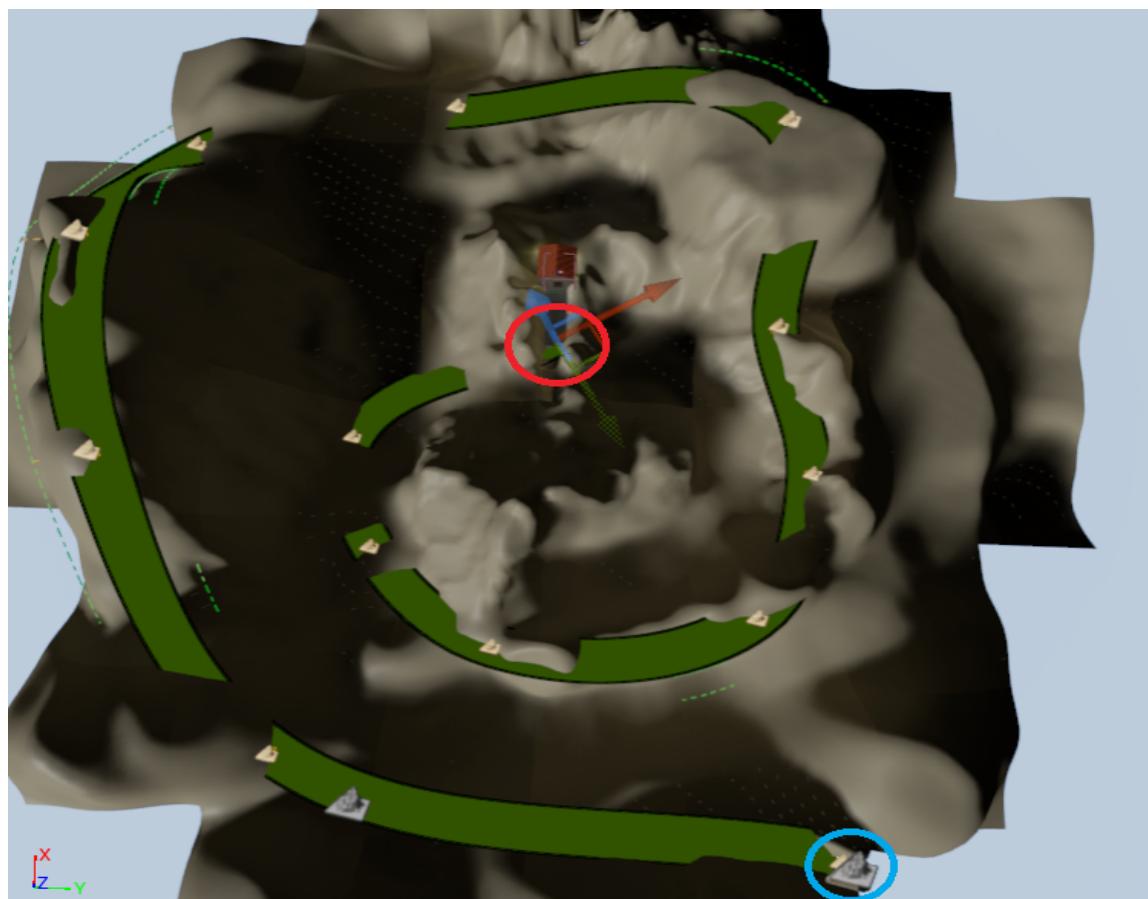


Illustration 94: Édition des "splines" avant application

12. Dans **DT**, panel «Landscape Spline Control Point», paramétrez comme sur l'illustration 95.

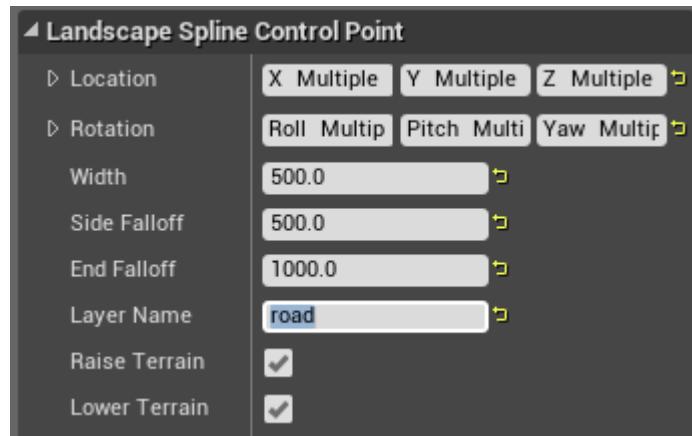


Illustration 95: Panel Details, volet de paramétrage de la courbe

13. Ensuite, ajoutez une à une différentes bornes («**Ctrl**+clic gauche») pour aller du point de départ (cercle bleu) à l'arrivée près des escaliers (cercle rouge).

Notez que **chaque borne est modifiable**: vous pouvez agrandir le chemin en cours de route, déplacer la borne pour mieux coller au terrain, etc.

Quand tout est OK:

14. Dans **MD**, cliquez sur «**All splines**»: permet de surélever ou d'abaisser le terrain pour qu'il se conforme au nouveau chemin.

Le résultat est visible à l'illustration 96 :

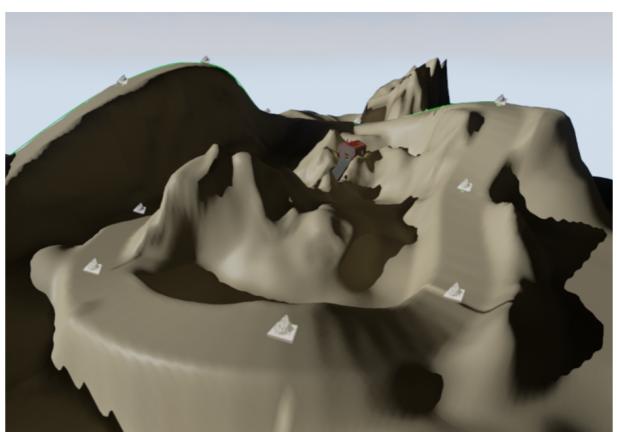
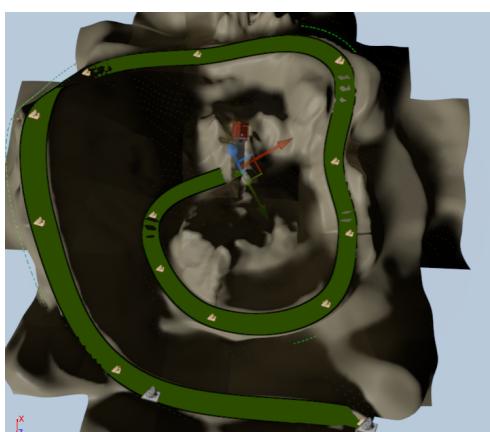


Illustration 96: Résultat de l'application de l'outil "splines"

Copier-coller des portions de paysage:

Il y a un outil du panel «Sculpt» dont nous n'avons pas encore parlé. Il s'agit des outils «**selection**» et «**Copy/paste**» de «**region tools**».

Pour les utiliser, **c'est très simple**:

15. Sélectionnez l'outil «**Selection**», puis avec la brosse, sélectionnez la zone à copier comme si vous utilisiez un pinceau.

16. Une fois que la zone vous semble bonne, appuyez sur **Ctrl + C** pour copier.

17. Une boîte (le **gizmo**) apparaît. On peut la redimensionner selon les besoins et la déplacer, mais il

faudra refaire un **Ctrl+C** pour copier le modèle.

Il est toujours possible de manipuler le modèle en mémoire en effectuant des transformations comme la rotation et le redimensionnement.

18. Déplacez le gizmo vers la zone que vous souhaitez modifier en collant cette nouvelle portion de paysage, puis **Ctrl+V** pour coller.

L'opération est relativement simple, mais il y a des paramètres que vous pouvez modifier.

Entre autres, il y a «**Paste mode**» qui permet de faire un filtre sur «uniquement les bosses» («**raise**») ou «uniquement les creux» («**lower**»). On peut aussi modifier «**Tool strength**» pour avoir, soit une copie parfaite (strength à 1), soit un modèle qui s'en approche (strength < 1).

Voilà, nous avons vu les principales fonctions. Nous pourrons toujours revenir plus tard sur la modélisation pour affiner les résultats. Maintenant, nous allons pouvoir «peindre» le paysage !

PEINDRE LE PAYSAGE

Gestion des layers

Nous pourrions juste modifier le matériau et lui donner une texture correspondant à de la roche par exemple. Mais nous pouvons faire beaucoup mieux: gérer plusieurs types de terrain (roche, sable, herbe, neige, route, etc.) et combiner le tout dans un matériau.

Ensuite, nous pouvons utiliser l'outil «Paint» de «Landscape» pour appliquer tel ou tel matériau au terrain.

Il y a aussi une autre méthode, plus automatique, qui consiste à définir une texture différente en fonction de la hauteur du terrain. Nous vous indiquerons un peu plus loin comment procéder. Pour le moment, nous allons préparer notre matériau afin qu'il soit «multi-couches» ou «multi-layer»:

1. Ouvrez le matériau «**MatLand**» dans l'éditeur de matériaux
2. Ajoutez un noeud «**LandscapeLayerBlend**», puis allez dans **DT**
3. Ajoutez un premier layer, renommez-le «**Ground**» (sol), «**Blend Type**»: «**LB Weight Blend**».

Ici, on choisit d'utiliser le mode «paint», mais si nous avions préféré utiliser la hauteur du terrain pour déterminer le layer à utiliser, nous aurions pris «**Blend Type**»: «**LB Height Blend**».

4. Ajouter un layer «**Grass**» pour l'herbe, «**Sand**» pour le sable, «**Road**» pour la route et «**Snow**» pour la neige

Encore une fois, si votre ordinateur est modeste, contentez-vous de 2 ou 3 layers. La composition du sol n'est pas un élément essentiel car nous allons ajouter des assets et de la végétation.

5. Relier la sortie du noeud à l'entrée «**Base Color**» du matériau
6. Copiez-collez le noeud «**Layer Blend**» et reliez sa sortie à l'entrée «**Normal**» du matériau

Nous donnons ici un exemple avec l'entrée «**Normal**», mais la technique peut être utilisée pour chaque entrée du matériau, dès qu'il faut appliquer une différence entre les layers.

7. Ajoutez les textures diffuses et normales correspondantes et reliez-les aux noeuds «**Layer Blend**» comme sur l'illustration 97.

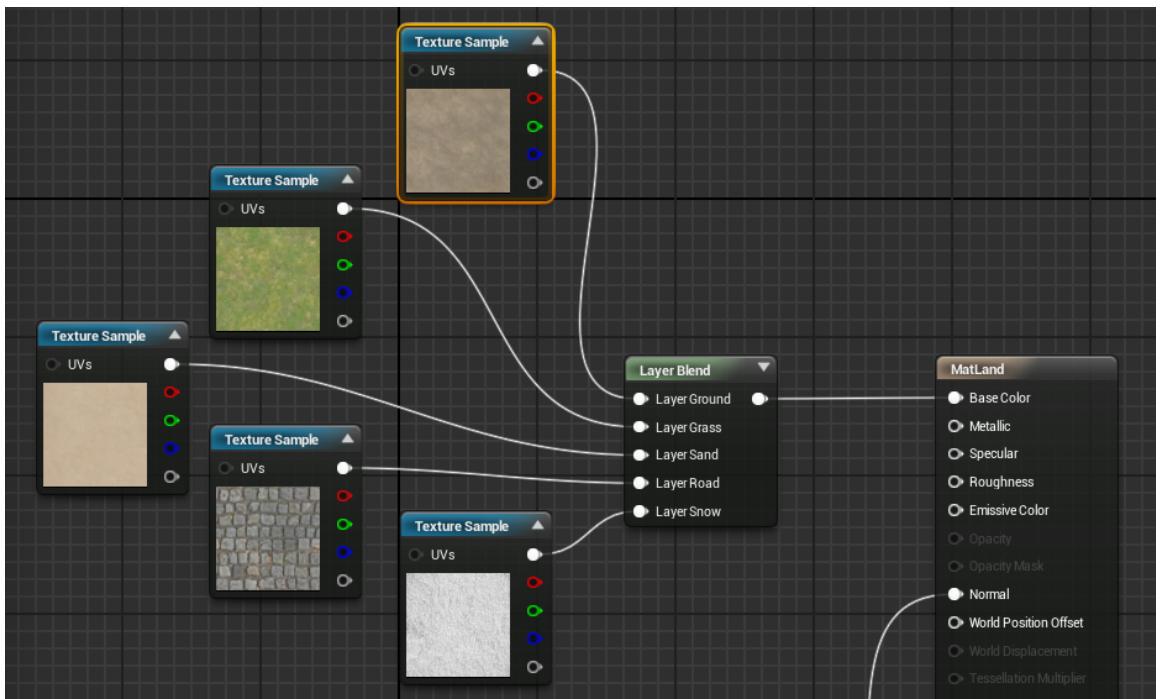


Illustration 97: Edition du matériau: paramétrage des layers pour les textures diffuses

8. Sauvez et revenez sous l'éditeur de niveau
9. Sélectionnez l'actor «landscape» et allez dans **MD**, volet «Landscape», section «Paint».

Dans le panel «Target Layers», vous retrouvez les différents layers créés sous l'éditeur de matériaux.

10. Cliquez sur «+» à droite de «Ground» comme sur l'illustration 98 pour créer une nouvelle map qui va contenir les informations du layer, et sélectionnez «Weight-Blended Layer (normal)».



Illustration 98: Ajout d'une nouvelle map d'info du layer

11. Une nouvelle fenêtre s'ouvre: laissez l'éditeur enregistrer l'asset où il le souhaite, ou créez un répertoire «landscape», puis cliquez sur «OK».
12. Sous «Ground», sélectionnez le nouveau fichier «Ground_LayerInfo».
13. Répétez l'opération pour les autres layers

Nous pouvons peindre le paysage:

14. Cliquez sur «ground» dans «Target Layers» et choisissez une brosse assez volumineuse (ex: «Brush size» à 1024).
15. Appliquez la brosse sur l'ensemble du paysage (passez l'affichage en mode «Unlit» si vous avez

- plusieurs zones trop sombres)
16. Prenez une brosse moins importante (ex:512) et peignez toute la **route** («target layer»: «Road») dessinée avec la courbe spline.
 17. Peignez le sommet des collines avec la **neige** («target layer»: «snow»)
 18. Appliquez du **sable** («target layer»: «sand») au bord des routes et un peu où vous le souhaitez
 19. Et enfin, créez une ou deux prairies avec l'**herbe** («target layer»: «grass»)

C'est un processus qui peut être très long avec une carte de cette taille. Le résultat n'est pas vite apparent: la peinture est faite grossièrement.

Par la suite, il faudra «littéralement» se promener dans le paysage, ajouter des détails, peindre par ci, par là si on veut obtenir un résultat à la hauteur de nos espérances.

Mais avant cela, que diriez-vous si on faisait sortir la roche du sol de façon automatique?

Un matériau qui s'adapte à l'inclinaison du terrain

La fonction que nous allons créer prend en paramètres deux textures (des «Texture Object» plus précisément, et pas des «Texture Samples») et choisi automatiquement laquelle afficher en **fonction de l'inclinaison de la surface du terrain**.

1. Dans «Materials», créez une nouvelle «**Material Function**» et passez en mode édition
2. Recopiez les différents éléments de l'illustration 99.

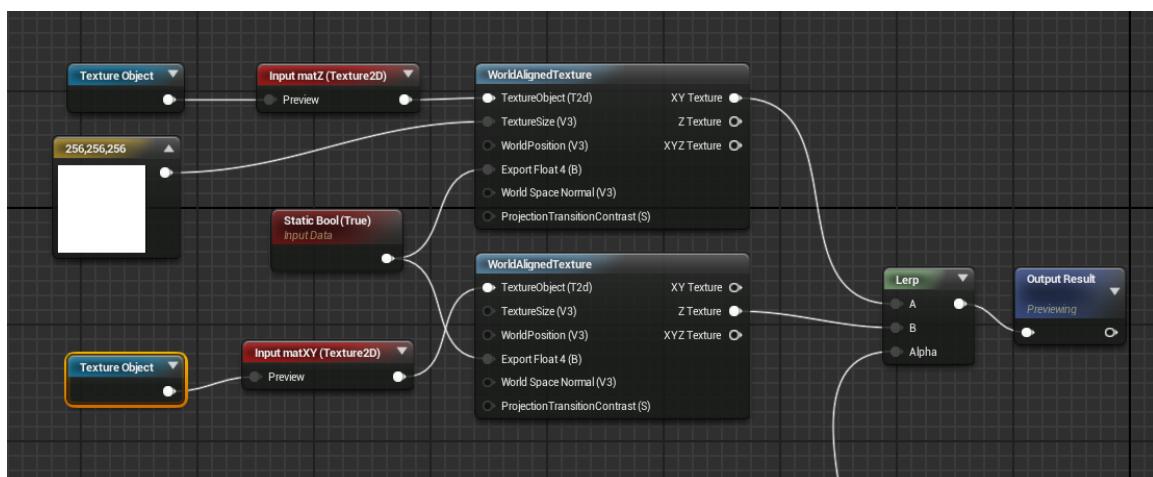


Illustration 99: Fonction matériau "MatMultiZ"

La fonction **Lerp** (interpolation linéaire) va choisir entre 2 textures:

- son entrée A est liée à la texture que nous allons utiliser pour recouvrir la partie verticale d'un objet,
- et l'entrée B est liée à la texture recouvrant la partie horizontale.

Entre les deux, un dégradé géré par le Lerp. Les paramètres d'entrée (les «inputs») sont matZ et matXY, les deux textures («Texture Object» et non «Texture Sample»).

Les deux «Texture Object» sont là uniquement pour l'aperçu, on peut les omettre sans modifier le fonctionnement de l'ensemble.

«**WorldAlignTexture**» permet de plaquer une texture en la répétant comme un motif sur la surface d'un objet, indépendamment de la taille ou de la rotation de cet objet. Cette fonction vous permet de spécifier la direction dans laquelle la texture sera projetée, ainsi que sa dimension.

Pour «**MatZ**», nous ajoutons une précision sur la taille de la texture (256x256) afin d'obtenir un redimensionnement de celle-ci, mais nous aurions pu le passer en paramètres.

3. Reliez l'entrée «alpha» du Lerp précédent au graphe de l'illustration 100.

Notez que le nœud «**Tangent To World**» s'obtient en utilisant «Vector Transform». La fonction «**Mask**» est un «component mask», c'est dans **DT** qu'on choisit le canal «B».

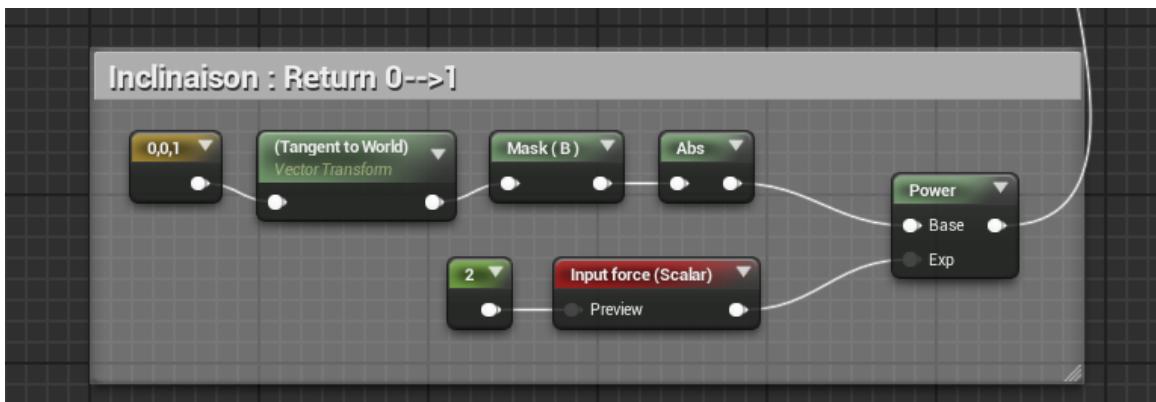


Illustration 100: Fonction alpha du Lerp qui renvoie entre 0 et 1 selon l'orientation de la surface

Après avoir examiné cette fonction, vous pourriez vous demander où se trouve la valeur du vecteur normal à la face de l'objet permettant de déterminer dans quelle mesure elle est horizontale ou verticale?

«**Tangent To World**» convertit un vecteur 3D d'un système coordonné de référence à l'autre. C'est dans **DT** qu'on détermine le système d'entrée (Tangent) et de sortie (World). Par défaut, tous les calculs de shaders dans un matériau sont faits dans l'espace tangent. Les vecteurs caméras, les vecteurs lumières, etc... doivent tous être transformés en espace tangent avant d'être utilisés dans un matériau.

C'est donc dans notre vecteur (0,0,1) que se trouve la réponse! Nous n'utilisons que la coordonnée Z de ce vecteur, mais c'est en effectuant une transformation spatiale de «tangent» en «world» que nous obtenons notre valeur.

«**Mask**» agit comme un filtre sur la coordonnée Z après transformation de l'espace.

«**Abs**» est une fonction renvoyant la valeur absolue de son entrée, quelle que soit la valeur en entrée, elle sera positive en sortie.

«**Input force**» est un paramètre qu'on ajoute pour donner plus ou moins de force au choix de la texture – Plus la valeur est élevée, plus le passage d'une texture à l'autre sera brutale (à 1, un dégradé très progressif de l'une à l'autre s'opère en fonction de l'inclinaison de la surface). C'est au travers de la fonction «**Power**» (puissance) que cette vitesse de transition s'exprime (Exemple: Power (7,4) = $7 \times 7 \times 7 \times 7$), c'est à dire, une progression exponentielle et non une progression linéaire.

4. Cochez la case «**Expose To Lib**» pour pouvoir utiliser la fonction à partir de la palette, puis sauvez.

Enfin, nous allons **utiliser cette fonction** dans notre matériau:

5. Modifiez votre matériau «**MatLand**» pour prendre en compte votre nouvelle fonction, comme sur l'illustration 101.

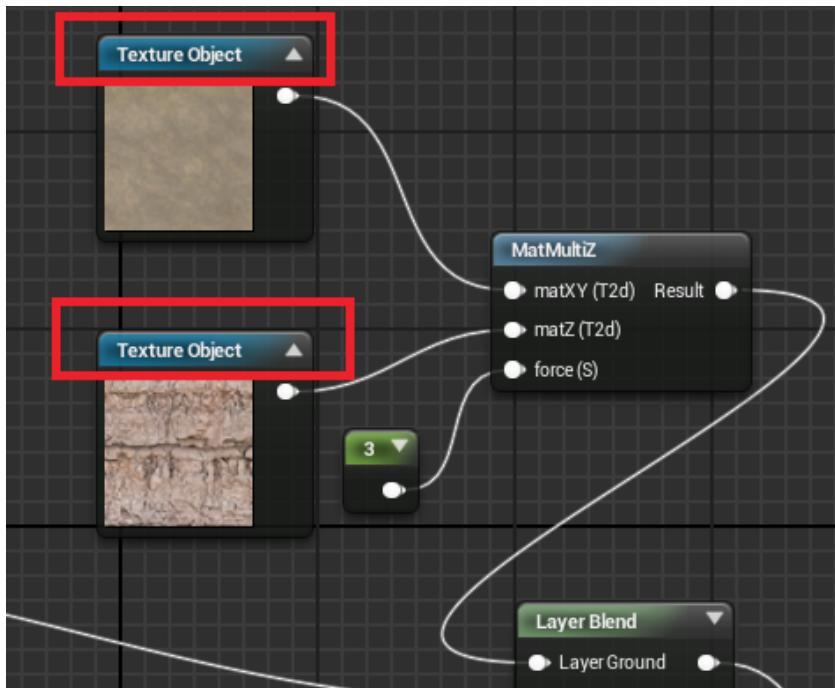


Illustration 101: Exemple d'utilisation de MatMultiZ

Pour rappel, les deux textures en entrée sont des «Texture Object». Pour obtenir ces dernières, utilisez les «Texture Sample» que nous connaissons et sur la texture, puis «Convert to Texture Object».

La texture que nous avons utilisée ci-dessus a été récupérée dans le projet «VehicleGame» que vous trouverez en téléchargement gratuit dans «Showcases and Complete Projects» du «MarketPlace».

Vous ne pouvez malheureusement **pas mélanger les deux approches**, c'est à dire n'utiliser que MatMultiZ pour l'un des layers, et non pour les autres. Dès que vous avez plusieurs layers par composant, l'affichage ne fonctionne pas.

Donc soit vous utilisez la fonction pour chaque layer, soit vous la remplacez par un équivalent comme sur l'illustration 102:



Illustration 102: Éléments à insérer pour utiliser MatMultiZ dans un matériau multi-layers

AJOUT D'UN PLAN D'EAU

1. Dans **CB**, «StarterContent», «Shapes», sélectionnez «**Shape_Plane**», renommez-le «Lake» et ajoutez-le à la scène
2. Redimensionnez-le pour qu'il soit de la taille du paysage (en XY).
3. Déplacez «Lake» pour recouvrir une partie de votre décor, mais en laissant la route apparente.
4. Dans «Starter Content», répertoire «Materials», vous trouverez un matériau du nom de «**M_Water_Lake**». Appliquez-le à «Lake»

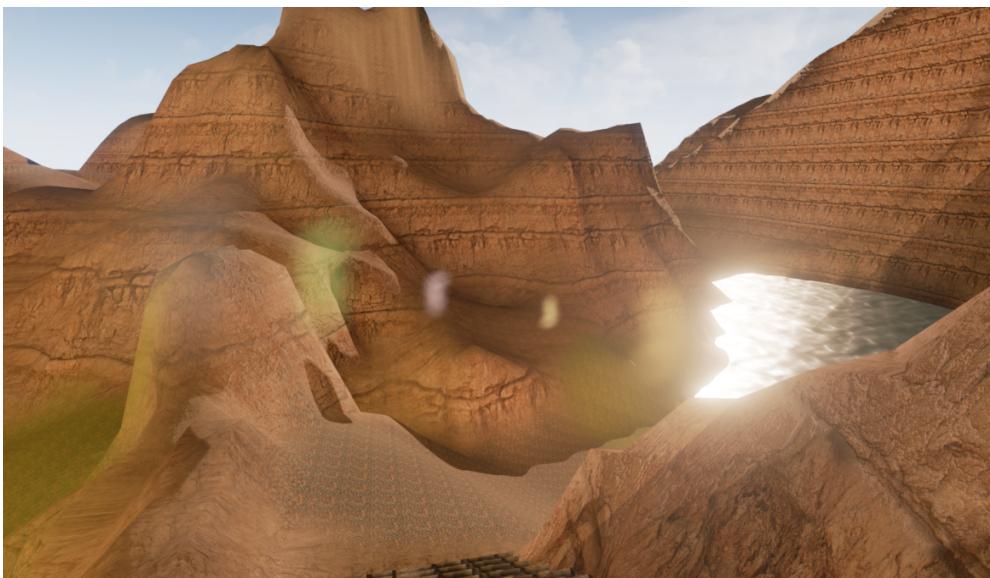


Illustration 103: Vue de la maison sur le paysage que nous venons de modéliser (plan d'eau à droite)

Nous avons déjà un premier aperçu de ce que sera notre paysage (voir illustration 103).

C'est un bon début: l'ajout de la végétation devrait parfaire l'ensemble.

Maintenant que vous connaissez la technique, somme toute très basique, vous pouvez trouver des plans d'eau déjà prêts sur le «**Market Place**». Certains sont même gratuits (section «Learn») comme vous pouvez le constater sur l'illustration 104.

Illustration 104: lot de "Water Planes" disponible gratuitement sur le Market Place

Il suffit de **télécharger** le pack et de **l'ajouter au projet courant**.

Le pack est intéressant car il contient 3 exemples de maps:

- «**LakeWater_Example**»: plan d'eau relativement calme
- «**OceanWater_Example**»: plan d'eau type «océan» avec de l'**écume!** (illustration 105)
- «**TranslucentWater_Example**»: plan d'eau en animation, transparent.

L'étude de ces différents matériaux est très intéressante, notamment pour la formation de l'écume.



Illustration 105: Plan d'eau type océan avec de l'écume

MISE EN PLACE DE LA VÉGÉTATION ET DU DÉCOR

Création d'un matériau «herbe»

Au fur et à mesure que nous avançons dans la création du paysage, vous avez probablement compris l'importance du travail sur les matériaux. C'est la raison pour laquelle nous avons préféré étudier ce domaine avant d'attaquer cette partie.

Les connaissances que vous avez pu acquérir vont servir ici pour créer un objet de type végétation que nous allons pouvoir utiliser avec l'outil «**foliage**».

Commençons par **créer notre matériau**:

Nous avons besoin d'une texture un peu particulière, représentant une touffe d'herbe. Pour cela, cherchez sur **Google images** avec les mots clés «**grass sprite**». Il est important que la texture sélectionnée possède une **couche alpha de transparence** comme dans l'illustration 106.

1. Téléchargez la texture et placez-la dans votre répertoire «Textures». Nommez la «**grass**»

Sous l'éditeur de texture, cochez «**Alpha**» dans «Color Channels» accessible via le bouton «View». Juste en dessous, dans les «Viewport Options», définissez un «**Background**» de type «**Checker**» pour mieux visualiser la transparence, comme dans l'illustration 106.

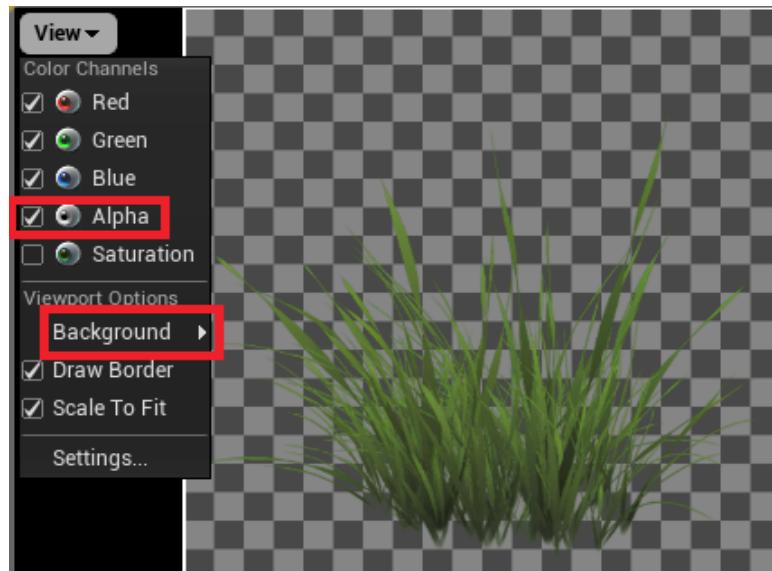


Illustration 106: Exemple de texture de type "herbe" avec un fond alpha

2. Créez un nouveau matériau «**MatGrass**» et dans **DT**, panel «Material», modifiez le «**Blend Mode**» en «**Masked**» pour prendre en compte la transparence par le canal alpha de la texture.
3. Cochez «**Two Sided**» pour que le matériau soit visible dans tous les sens s'il est utilisé par un plan.
4. Décochez «**Tangent Space Normal**» pour utiliser un système plus simple pour l'entrée Normal (juste un vecteur).
5. Importez la nouvelle texture «grass» dans le graphe
6. Ajoutez les différents nœuds comme sur l'illustration 107 puis sauvez.

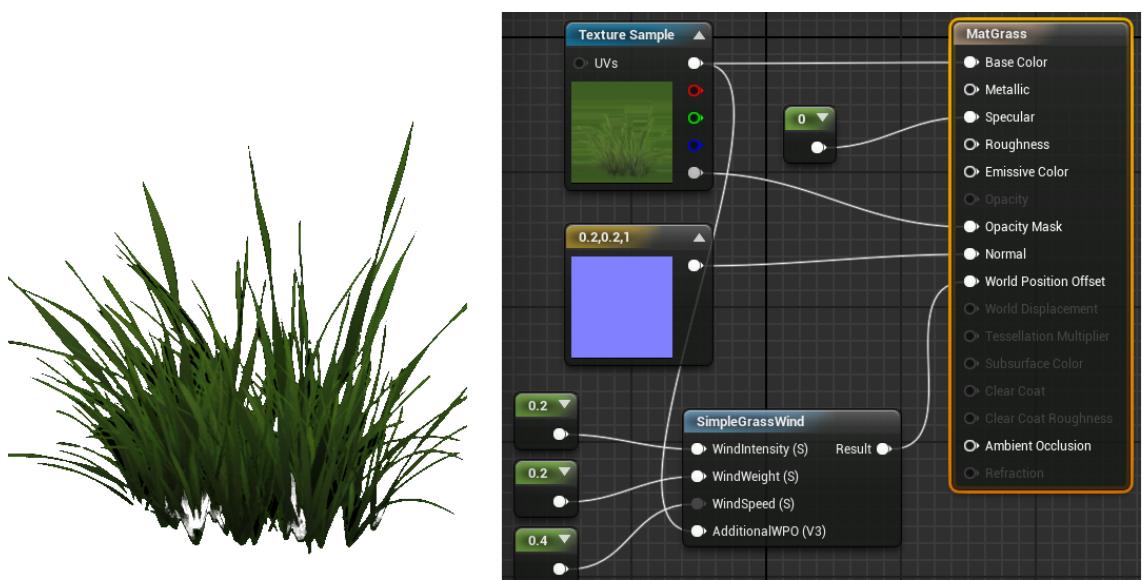


Illustration 107: Matériau MatGrass avec un petit effet vent

«**SimpleGrassWind**» est une fonction matériau permettant d'**ajouter un petit mouvement sur le matériau**. Il sert à faire bouger les feuilles d'un arbre, les brins d'herbe, les fleurs... tout ce que l'on souhaite. Il suffit de le relier à l'entrée «**World Position Offset**» du matériau. Ses différentes entrées permettent de moduler l'effet (force, vitesse, etc.). Réalisez plusieurs tests et regardez l'effet dans le preview.

Maintenant, il reste à appliquer ce matériau sur un «static mesh». Si vous savez utiliser un modeleur, vous pouvez modéliser la touffe d'herbe sous la forme de plusieurs plans enchevêtrés comme sur l'illustration 108. Sinon, vous pouvez utiliser l'asset «**grassMesh**» **téléchargeable** à cette adresse: <http://bit.ly/1EW0vku>

Nous allons voir par la suite comment fonctionne l'outil foliage. Nous allons utiliser comme exemple un autre asset, mais rien ne vous empêche de l'utiliser avec celui-ci.

L'illustration 109 donne un exemple de l'utilisation de ce nouveau matériau dans l'environnement précédemment créé. Les brins d'herbe bougent délicatement dans le vent... le bord du lac commence à être agréable!

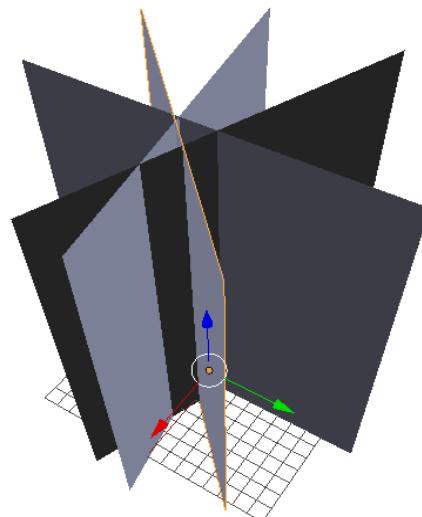


Illustration 108: Plusieurs plans assemblés pour former une touffe de végétation

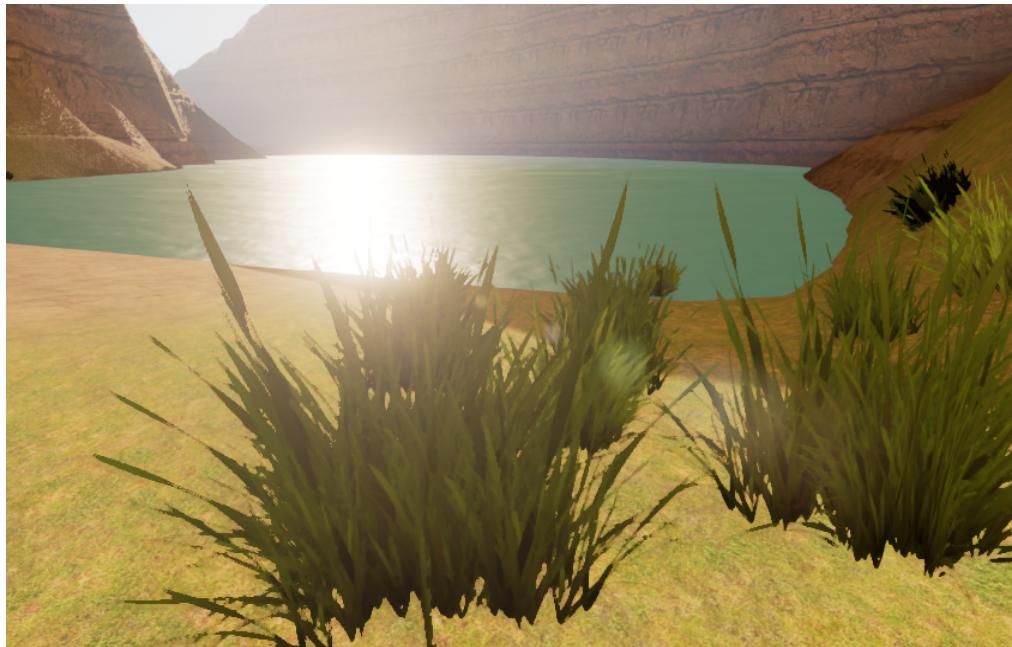


Illustration 109: Exemple d'intégration dans la paysage

Utilisation de l'outil foliage

Lorsqu'on souhaite souvent utiliser un élément de décor (fleur, rocher, lampadaire, ...) on pourrait être tenté de les insérer dans la scène sous la forme d'un «actor». Mais cela finirait par alourdir le graphe de scène (visible sous **WO**), aboutissant à terme à une **saturation** d'Unreal Engine.

L'outil «**Foliage**» consiste à utiliser un modèle d'objet graphique (staticmesh) et à le cloner un très grand nombre de fois. Ces clones, sur lesquels on peut appliquer des transformations de base (déplacement, rotation, redimensionnement), s'appellent des **instances**.

Unreal Engine intègre un grand nombre d'optimisations, qui vont du clipping au Lod, en passant par des économies de transfert entre la mémoire centrale et la mémoire du GPU. Et le **résultat est grandiose**: selon votre configuration, si vous paramétrez correctement votre paysage, vous pourrez insérer plusieurs centaines de milliers d'instances d'arbres, **voir des millions** !

Le terme «Foliage» se traduit en français par **feuillage**, ce qui peut porter à confusion, sauf si on connaît l'historique de l'outil. Au départ, on utilisait les instances pour créer un semblant de feuillage sur les arbres: il s'agissait le plus souvent de plans contenant l'image d'une feuille qu'on dupliquait un très grand nombre de fois. Maintenant, nous utilisons ce procédé pour tout objet que l'on duplique en nombre et dont nous n'avons pas besoin de programmer le comportement.

Faisons connaissance avec l'interface (illustration 110). Quand on ouvre le volet «Foliage», on découvre plusieurs sections (sous-onglet):



Illustration 110: Outil "foliage"

- «**Paint**»: pour «peindre» le paysage, mais aussi des objets ou des BSP, à l'aide de Static Meshes.
- «**Reapply**»: permet de modifier des instances déjà créées via l'outil «Paint»
- «**Select**»: pour sélectionner certains instances individuellement, pour les repositionner ou les supprimer par exemple
- «**Lasso**»: idem, mais en utilisant un lasso, c'est à dire, en utilisant la brosse pour sélectionner à la volée un certain nombre d'instances
- «**Fill**»: permet de recouvrir la surface d'un objet avec un Static Mesh.

L'outil «**Paint**» permet de «peindre» lorsqu'il est utilisé avec le clic gauche, mais on peut également effacer en utilisant «**Shift**+clic gauche».

Concernant l'outil «Paint»:

- «**Brush Size**»: taille de la brosse que nous allons utiliser pour peindre.
- «**Paint Density**»: multiplicateur qui s'opère sur la densité propre à chaque foliage. Si la densité du foliage est 100, et que «Paint Density» est à «0.3» alors la densité effective quand on applique la brosse est de 30.
- «**Erase Density**»: avec une densité de 0, l'outil efface toute la zone couverte par la brosse. Avec une densité de 0.5, seule la moitié des instances disparaissent.

- «**Filter**»: permet de déterminer le type d'objet que la brosse peut peindre. Si on a ajouté des rochers de type «**Static Mesh**» et que l'on souhaite qu'ils soient «peints» en même temps que le paysage, il est nécessaire d'activer les cases «**Landscape**» et «**Static Meshes**». «**Translucent**» permet de peindre sur les surfaces transparentes.

Enfin, glissez un «**Static Mesh**» dans la zone encadrée en rouge («**Drop Static Meshes Here**») pour ajouter une **nouvelle instance de foliages**. C'est aussi dans cette zone qu'on retrouve la liste des foliages déjà créés.

C'est en forgeant qu'on devient forgeron:

1. Allez dans **CB**, «Starter Content», «Props» et sélectionnez l'asset «**SM_Brush**»: il s'agit d'un petit arbre.
2. Allez dans le volet «**Foliage**» de **MD**, section «**Paint**» et glissez-déposez «**SM_Brush**» dans la zone prévue à cet effet.

Un nouveau panneau «**Meshes**» apparaît à l'endroit même où vous avez déposé «**SM_Brush**» comme sur l'illustration 111.

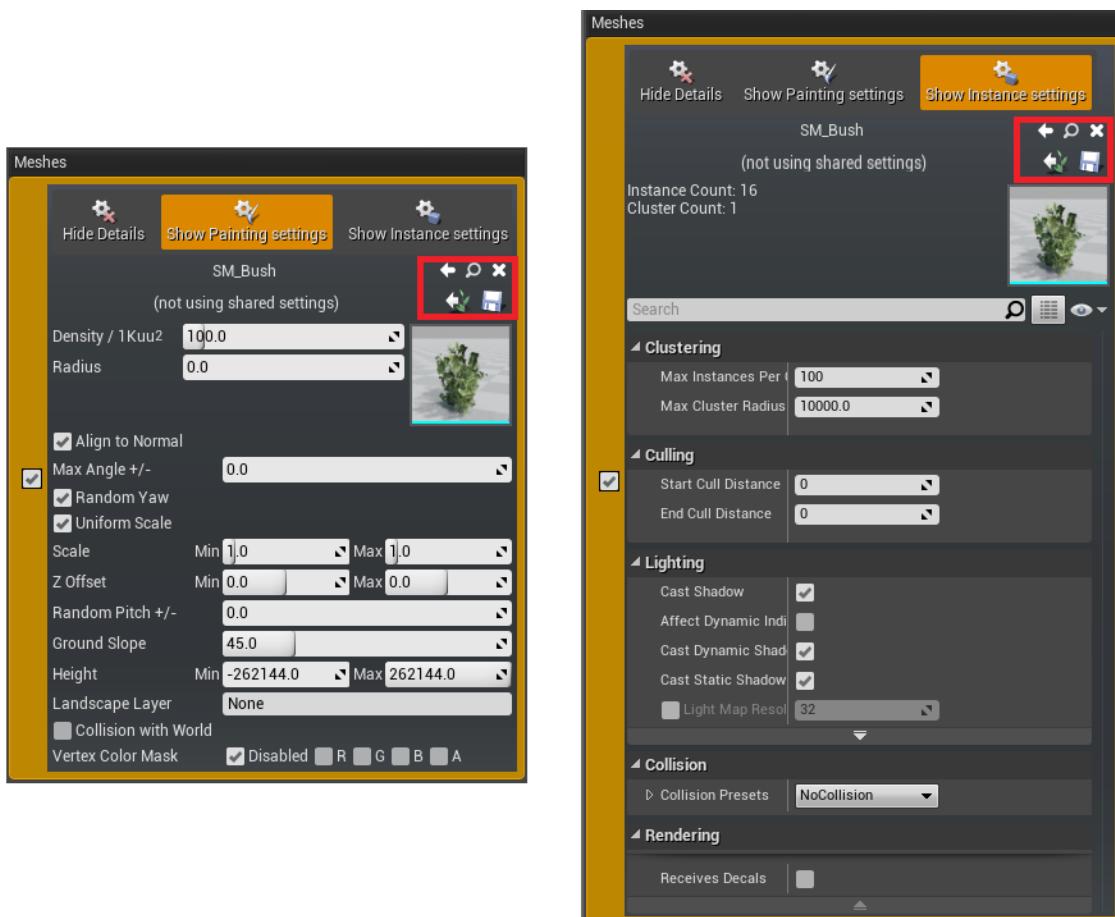


Illustration 111: Outil Foliage Paint: à gauche le volet de settings, "Painting" à gauche et "instance" à droite

Étudions de plus près ces volets:

- Le volet «**Hide Details**» permet de basculer sur une vue simplifiée, ce qui est pratique lorsqu'on dispose de nombreuses instances.

- Le volet «**Show Painting Settings**» (à gauche): il correspond au paramétrage de l'instance actuellement utilisée pour peindre.
- Le volet «**Show Instance Settings**» (à droite): il s'agit d'autres paramètres relatifs aux instances en cours de création. On retrouve, entre autres, le nombre d'éléments déjà déposés («**Instance Count**»).

Communs aux deux panneaux, on trouve le nom du «Static Mesh» actuellement utilisé, ainsi que diverses icônes (encadré rouge).

Les paramètres que nous pouvons modifier constituent une configuration qui peut être sauvegardée. En face du nom du mesh:

- «Flèche gauche»: substituer le static mesh en cours par celui sélectionné dans **CB**
- «Loupe»: rechercher et sélectionner dans **CB** le static mesh utilisé dans par l'outil foliage
- «Croix»: supprimer toutes les instances du foliage (et la configuration par la même occasion)

«**Not using shared settings**» signifie qu'on n'utilise pas une configuration qui a été préalablement sauvegardée à l'aide de l'icône «disquette». Juste à gauche de cette icône, une «flèche» permet de charger la configuration qui est actuellement sélectionnée dans **CB**

Attachons-nous maintenant à décrire les éléments du volet «**Painting Settings**»:

- «**Density / 1 Kuu²**»: Nombre d'instances max par 1000x1000 unités de terrain
- «**Radius**»: Espace minimal (rayon) entre deux instances
- «**Align to normal**»: Si case est cochée, alors les éléments sont toujours placés perpendiculairement à la surface. Sinon, ils sont verticaux.
- «**Max angle +/-**»: Si «align to normal» est coché, «max angle» va définir l'angle max auquel l'instance peut-être inclinée. Si par exemple, il est fixé à 15° et que la paroi est à 45°, alors l'instance ne sera pas perpendiculaire à la surface, mais inclinée de 15°. Si la paroi est à 10°, alors cela ne change rien.
- «**Random Yaw**»: Permet une rotation d'axe Z de l'instance, de façon aléatoire.
- «**Uniform scale**»: Si la case est cochée, alors tout redimensionnement se fait de façon uniforme sur tous les axes. En décochant cette case, on accède à un paramétrage par axe.
- «**Scale min, max**»: Permet un redimensionnement au hasard des instances, borné selon un min et un max.
 - «**Z offset min, max**»: Permet aux instances de ne pas être collées à la surface.
Valeur négative → les objets peuvent apparaître plus bas que la surface.
Valeur positive → ils paraissent voler au dessus de la surface.
Le bornage min/max permet d'ajouter une composante aléatoire qui est tirée entre min et max.
- «**Random pitch +/-**»: Autorise une rotation de l'instance jusqu'à un certain angle par rapport à l'axe Z, et ce de façon aléatoire.
- «**Ground slope**»: Permet de définir l'inclinaison maximale de la surface, inclinaison au-delà de laquelle l'instance ne sera pas créée. Par exemple, si vous souhaitez pouvoir peindre une surface verticale, il faudra passer ce paramètre à 90°.
- «**Height min, max**»: Permet d'établir une limite de hauteur, plus précisément une bande dans laquelle il est possible de peindre, mais pas au-delà. C'est une façon de définir une altitude pour telle plante, et une autre pour telle autre.
- «**Landscape Layer**»: permet de limiter l'action de la brosse à un seul layer. Si vous voulez ajouter de l'herbe et que vous avez un layer «grass», indiquez juste ce nom dans ce champ.
- «**Collision with world**»: Si la case est cochée, un test de collision avec les autres objets est réalisé et l'instance n'est créée que si elle n'entre en collision avec aucun objet. Ainsi, il n'est plus possible d'ajouter un élément sous la surface («**Z Offset**») par exemple.
- «**Vertex color mask**»: Certains objets peuvent être peints au niveau des vertex. On peut utiliser les masques de couleur au niveau des vertex pour limiter l'action de la brosse à certaines zones.

Si vous importez des heightmap de paysage à partir de logiciels externes, vous serez peut-être amené à utiliser cette fonction.

Bon, **revenons à notre tâche**, nous reviendrons sur le second volet juste après:

3. Prenez une brosse de taille suffisamment large (ex: «1024») avec un facteur de densité de «1.0» (**«Paint density»**). Filtrez pour ne conserver que le **«Landscape»**
4. Paramétrez la densité à «128» unités / Kuu²
5. Espacez les arbustes d'au moins «2» unités (**«Radius»**)
6. Le facteur de redimensionnement (**«scale»**) est entre «0.5» (la moitié) et «3»
7. On permet un enfoncement dans le terrain des arbustes allant jusqu'à «-10» unités (**«Z offset min»**)
8. On permet une inclinaison des troncs jusqu'à 15° (**«Random Pitch +/-»**)
9. Les arbustes poussent sur des surfaces inclinées jusqu'à 35° (**«Ground Slope»**)
10. Et enfin, limitez le **«Landscape Layer»** à **«grass»** pour ne faire pousser les arbustes que dans les zones «vertes», ces zones ayant elles-mêmes été définies au bord de l'eau et dans les plaines de basse altitude.
11. Enregistrez la configuration sous **«Tree_Settings»**
12. Balayez les zones correspondantes «vertes» avec la brosse en cliquant pour déposer des arbustes et **«Shift + clic gauche»** pour effacer quand vous n'êtes pas satisfait.

Commencez par déposer entre 500 et 1000 arbustes, cela devrait déjà donner de bons résultats. Le nombre d'instances peut se consulter dans le volet **«Show Instance Settings»**.

13. Allez dans le volet **«Show Instance Settings»**
14. Changez **«Start Culling Distance»** à 3000 et **«End Culling Distance»** à 6000

Pour que le **«Start Culling Distance»** fonctionne, il va falloir tout d'abord modifier le matériau.

15. Ouvrez le matériau **«M_Brush»** en édition
16. Modifiez le nœud tel que sur l'illustration 112, puis sauvez.

Maintenant, vous allez vous approcher d'une zone boisée et vous éloigner progressivement vers le ciel. Quand vous serez arrivé à 3000 unités des premiers arbustes, vous les verrez progressivement s'atténuer (**«fade out»**) jusqu'à complètement disparaître en arrivant à 6000 unités de distance.

Ce n'est pas un réglage optimal pour notre scène, mais c'est intéressant pour les tests. Vous pourrez toujours modifier ces valeurs un peu plus tard.

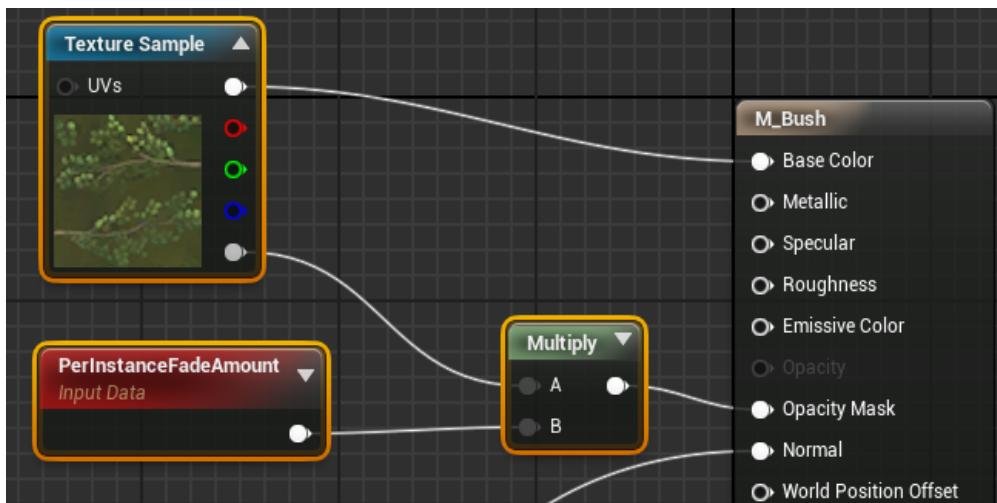


Illustration 112: Ajout de "PerInstanceFadeAmount" pour permettre au fade out du culling de fonctionner

Ces paramètres d'optimisation «**Culling Distance**» permettent de cacher les objets éloignés. Ici, il est réglé pour qu'un joueur de type «personnage» puisse se déplacer sur le terrain sans être trop gêné par la disparition de ces objets.

C'est en testant que vous pouvez décider d'augmenter ou non le culling: c'est aussi un paramètre de configuration possible pour votre jeu.

Notez qu'une valeur de 0 pour «**End Cull Distance**» désactive le système de culling du foliage courant

17.Décochez la case «Cast Dynamic Shadows»

Un autre paramètre important est le «lighting». Si vous laissez cochée «**Cast dynamic Shadows**», il y a un calcul permanent réalisé en fonction des lumières «mobiles». En effet, les ombres projetées par les lumières «statiques» sont encodées dans les lightmaps en pré-calcul – ce n'est pas un facteur de ralentissement. Mais si vous pouvez utiliser des ombres dynamiques, c'est tant mieux! Vous pourriez avoir besoin des ombres dynamiques si vous utilisez une torche, ou si vous faites varier la position du soleil et que c'est un facteur important dans le jeu.

De la même façon, si par exemple vous venez d'ajouter des brins d'herbe et que l'ombre statique projetée n'a aucune importance, vous pouvez désactiver les ombres statiques en décochant «**cast static shadows**».

18 Effectuez de nouveau un «Build» et testez.

Vous notez que les modifications que vous avez apportées à la configuration n'ont pas besoin d'être sauvegardées à nouveau. Elles sont sauvegardées automatiquement dès que vous modifiez la moindre valeur. La configuration est dite partagée («**shared**»). Vous pouvez couper cette synchronisation en cliquant sur une nouvelle icône qui remplace la «disquette» et dont la description indique «**Do not store ...**».

Un point important: aucun actor n'apparaît sous **WO**. C'est normal, **le foliage n'est pas considéré comme un actor**, il consomme donc, à ce titre, beaucoup moins de ressources. Toutefois, on ne peut pas entrer en interaction avec ces éléments, autrement que par une collision. Ainsi, pour ajouter des rochers, des maisons ou tout autre objet de décoration, vous pouvez passer par le foliage.

Les **instances** sont groupées par cluster. Quand les objets disparaissent grâce au culling, ils le font simultanément par cluster. Si le nombre de clusters n'est pas suffisant et si le radius est trop important, le fonctionnement du culling est altéré. Le nombre idéal d'instances par cluster dépend de la complexité du mesh: plus le mesh est détaillé, c'est à dire possède un nombre de polygones important, et moins le cluster doit contenir d'objets. Le cluster est aussi utilisé par le système de LOD.

Comme dans l'exemple précédent avec le matériau «herbe», n'hésitez pas à modifier le matériau «**M_Brush**» pour ajouter un effet de vent avec «**SimpleGrassWind**». Vous pouvez également ajouter quelques touffes d'herbe par ci, par là.

Nous avons fait le tour du foliage. Nous vous invitons à modifier individuellement un élément du foliage avec l'outil «**Select**», à utiliser le «**lasso**» pour sélectionner plusieurs éléments et les déplacer par exemple. L'outil «**Fill**» est assez pratique pour couvrir toute une surface de façon semi-aléatoire, sans peindre.

Sur le «**Market Place**», vous trouverez de nombreux assets que vous pouvez ajouter comme décors dans votre niveau: des cactus, des rochers, etc. Certains sont gratuits, d'autres payants. Vous pouvez également utiliser les assets contenus dans les exemples fournis par Epic Games.

19.Ajoutez un autre «Static Mesh» au foliage (exemple: «SM_Rock**» du répertoire «Props» du «StarterContent»).**

Que constatons-nous ?

Observez l'illustration 113: les deux éléments sont sélectionnés – cela signifie que si vous appliquez la brosse, vous allez «peindre» avec les deux éléments simultanément! Veillez donc à cocher et décocher les éléments en fonction de vos besoins pour ne pas avoir de surprise.



Illustration 113: Deux foliages sélectionnés et utilisables simultanément par la brosse

Pour que le personnage ne passe pas à travers les rochers, il est nécessaire d'effectuer deux modifications:

- 20.Ouvrez «SM_Rock» avec l'éditeur de mesh.
- 21.Si «**collision**» n'est pas actif, cliquez sur le bouton pour l'**activer** (illustration 114)

Vous remarquez qu'**aucun mesh de collision n'entoure le rocher**, nous allons devoir en créer un:

- 22.Menu «Collision», cliquez sur «**Auto Convex Collision**»: gardez les paramètres proposés et dans la fenêtre «Convex Decomposition», cliquez sur «**Apply**»

Un mesh de collision apparaît alors autour du mesh.

- 23.**Sauvez** et revenez à l'éditeur de niveau

Maintenant, il va falloir paramétriser le foliage de type «rocher»

- 24.Sélectionnez le foliage de type rocher et allez dans le volet «**Instance Settings**»
- 25.Sélectionnez «**Collision Presets**»: «**BlockAll**».

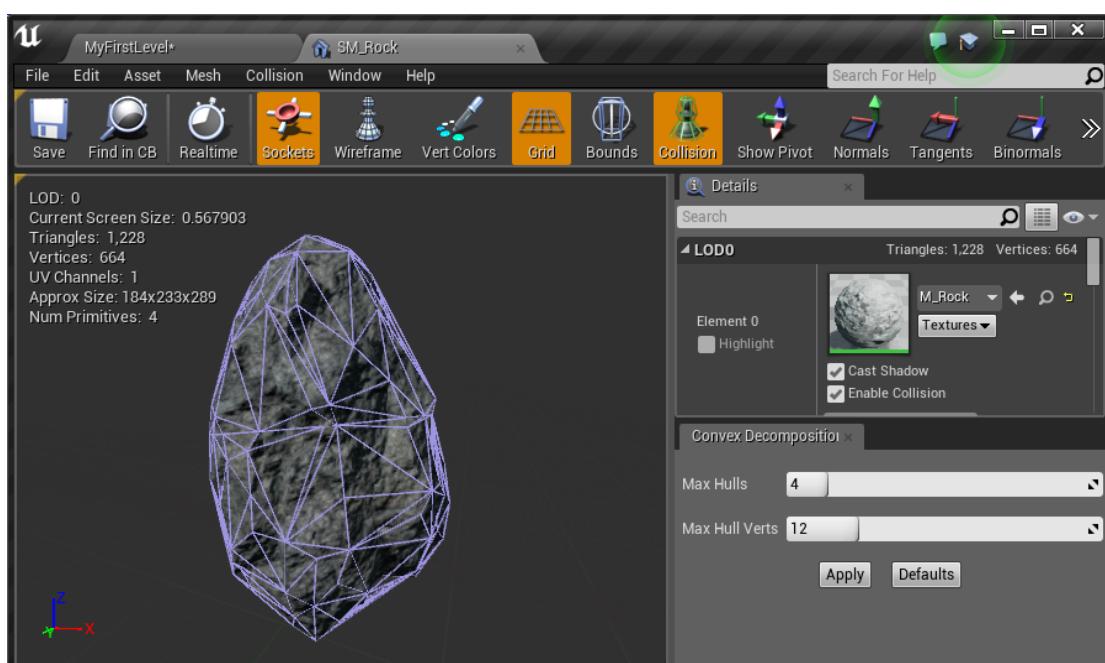


Illustration 114: Ajout d'un mesh de collision autour de SM_ROCK

En utilisant quelques assets supplémentaires tirés du pack «Landscape Mountains», voici quelques vues du paysage que nous obtenons:



Illustration 115: Vue vers le lac à partir de la maison avec le foliage

Les assets de la démo «Open World»

Une conférence intitulée “Tech & Art of Epic’s Open World Demo” a été tenue par Epic Games, ceux qui développent Unreal Engine, lors de la GDC 2015. Vous trouverez l'intégralité des diapositives présentées lors de cette conférence, ainsi que la vidéo (en anglais) à cette adresse: <http://bit.ly/17V9qov>



La plupart des participants sont restés médusés: la conférence commence par un film d'animation. On y voit un cerf volant traversant de vastes espaces de montagne, ainsi qu'un petit garçon qui poursuit tant bien que mal ce cerf volant. Les effets sont dignes d'un Pixar. Le paysage est splendide, l'animation est fluide et les prises de vue époustouflantes... Sauf qu'il ne s'agit pas d'un film d'animation au sens habituel du terme: tout est affiché en temps-réel (avec une Titan X monté sur une grosse machine, mais tout de même).

Avec cette démonstration, Epic a fait la démonstration de la toute puissance de son moteur quand il s'agit d'afficher des mondes vastes et ouverts (Open Worlds).

Quelques semaines après, coup de théâtre: vous pouvez désormais télécharger plusieurs assets de la démonstration et l'utiliser dans vos propres projets, et cela librement!

News

Learn

Marketplace

Open World Demo Collection

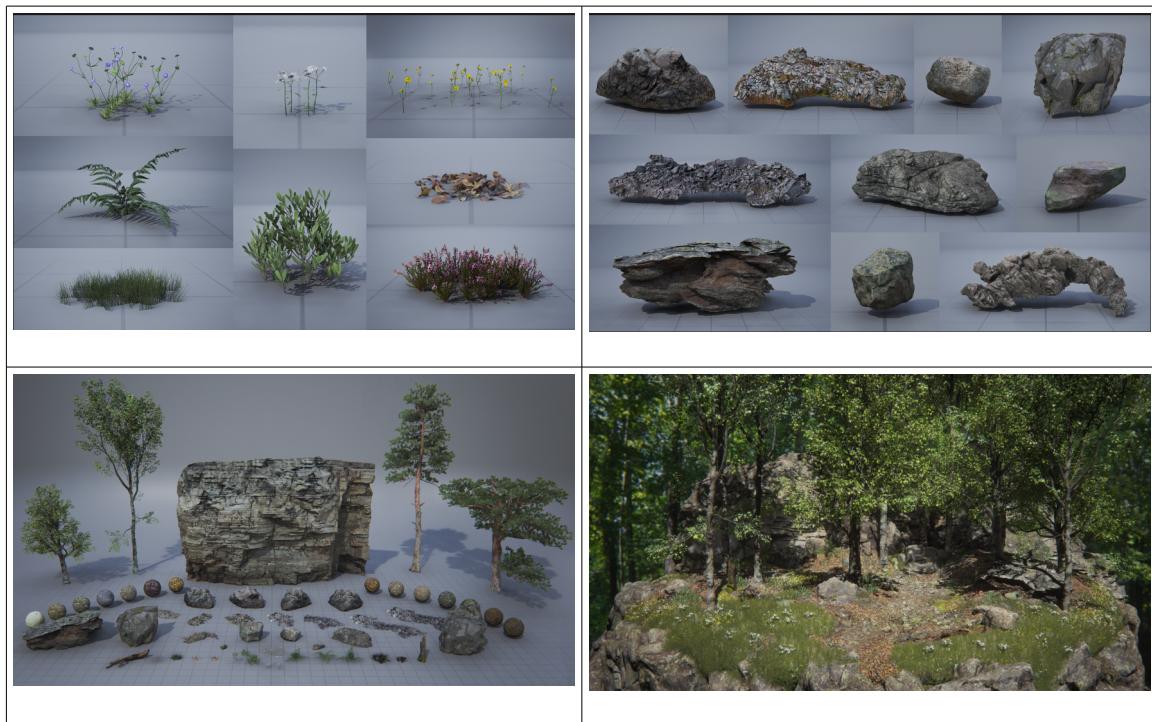
by Epic Games - April 6, 2015

Licensed for use only with UE4 based products.

A collection of realistic assets from Epic's Open World demo shown at GDC 2015. Use them to create new levels or add them to existing ones.

Add to project 4.7 ▾

Illustration 116: Assets de la démo Open World sur le Market Place



Maintenant, vous avez vraiment de quoi jouer avec l'outil «Foliage»!

Le Foliage Starter Kit

Vous trouvez également un autre pack intéressant réalisé par **MetalGameStudios** en vous rendant à cette adresse : <http://bit.ly/183FExN>



Le Foliage Starter Kit est un package spécialement conçu pour l'UE4. Il vous donne la possibilité de créer des scènes impressionnantes type «forêts / parcs / jardins» avec une grande variation due aux différents paramètres dans les instances de matériaux.

Cela signifie que vous pouvez modifier les arbres en changeant de saison: d'été à l'hiver ou l'automne en changeant juste un paramètre dans l'instance du matériau. En outre, vous pouvez modifier les effets de vent ainsi que la quantité de balancement des arbres. Vous pouvez ajouter de la neige également. Une autre caractéristique est que les plantes terrestres sont interactives: elles se déplacent quand le joueur marche sur elles.

Tous les meshes sont en **low poly** de sorte que vous pouvez les utiliser sur une très grande échelle. Des paysages entiers peuvent être remplis avec eux sans grandes pertes de performance.

Le **package** est composé de:

- 46 Meshes: Grass, Flower, Bush, Trees (with LOD's)
- 20 Textures
- 9 Master Materials: Grass/Ground Plants, Tree, Landscape, Particle Material
- 1 Particle System: Falling Leaves

Concernant son utilisation: «*Vous pouvez les utiliser dans vos projets commerciaux et non-commerciaux. Il suffit de garder à l'esprit que vous ne devez pas redistribuer le pack tel quel et ne pas l'utiliser dans d'autres moteurs qu'Unreal Engine*» (consigne de MetalGameStudios).

CYCLE JOUR/NUIT

Dans notre scène, deux objets existaient déjà à l'origine, mais nous aurions pu les ajouter à partir de la fenêtre **MD**:

- «**Light Source**»: lumière de type directionnelle
- «**BP_Sky_Shere**»: blue print permettant de gérer une atmosphère, un soleil et des nuages.

Exampons tout d'abord «**BP_Sky_Shere**» à partir de **DT**:

- «**Sun Height**»: permet de gérer la hauteur du soleil: 0 au niveau de l'horizon, 1 pour le zénith.

Pour faire tourner le soleil (droite/gauche), il faut agir sur la rotation d'axe Z de BP_Sky_Sphere.

Pour voir le ciel étoilé, utilisez -1. Dans ce cas, augmentez également la valeur de «Star Brightness».

- «**Colors determined by Sun Position**»: pour que les couleurs de l'environnement changent en fonction de la position du soleil, cochez cette case.
- «**Directional Light Actor**»: la lumière du soleil est émise grâce à une lumière directionnelle qui est dans l'illustration paramétrée comme étant «Light Source» (il faut d'abord cocher «Atmosphere Sun Light» dans ses propres détails). Le fait de lier les deux permet de réaliser un calcul automatique de «Sun Height» en fonction de la rotation de «Light Source».
- «**Cloud Opacity**»: permet de rendre les nuages plus ou moins transparents. Une valeur de «1.5» donne un effet «plus nuageux», alors qu'en réalité ils sont juste un peu plus apparents.
- «**Cloud Speed**»: vitesse de déplacement des nuages dans le ciel (pour les figer, utilisez 0).

Sans «**Light Source**», la scène est dans le noir complet (hormis la présence d'autres lampes):

Si vous ne comptez pas bouger la source de lumière pendant le jeu, optez pour «**Static**» pour qu'Unreal Engine pré-calcule un maximum de choses et optimise la gestion des ombres et autres éléments. Mais dans ce cas, il n'y a aucune projection sur les objets dynamiques tels que votre personnage. «**Stationary**» est le meilleur compromis dans ce cas.

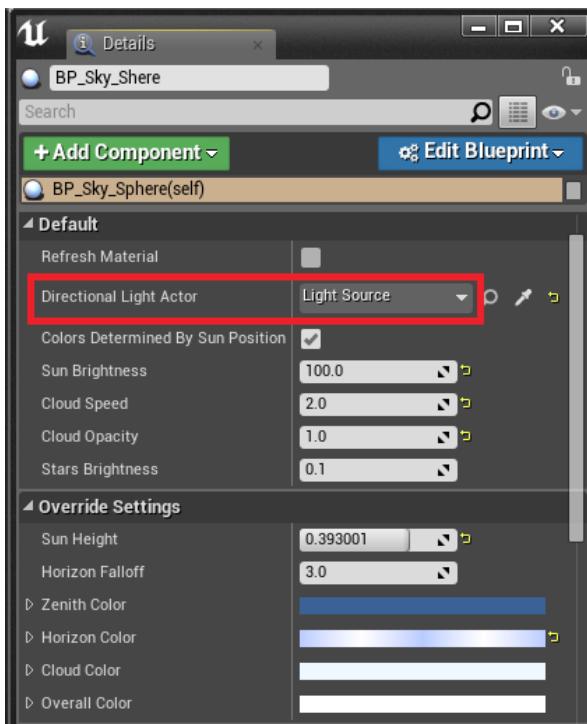


Illustration 117: Fenêtre "Details" de l'actor BP_Sky_Sphere

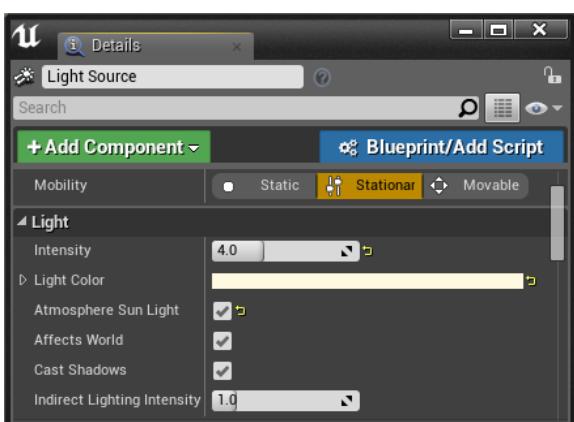


Illustration 118: Paramétrage de "Light Source"

Pour les autres paramètres:

- «**Intensity**»: intensité totale de la lumière (0 pour aucune).
- «**Light Color**»: permet de modifier la couleur de la lumière, un léger «jaune orangé» donne un coté beaucoup plus chaud à la scène.
- «**Affects World**»: si décoché, la lumière n'a aucun impact sur l'environnement.
- «**Cast Shadows**» peut-être désactivé pour ne pas afficher les ombres (un paramétrage plus précis peut être réalisé entre les ombres statiques et dynamiques avec «**Cast Static Shadows**» et «**Cast Dynamic Shadows**»). On peut tenter de forcer les ombrages dynamiques en mode «static», ça ne donne aucun résultat! **Qui peut le plus peut le moins**, c'est uniquement dans ce sens que ce paramétrage est possible.
- «**Indirect Lighting Intensity**»: permet de modifier la contribution indirecte de la lumière (quand elle ne touche pas directement un objet, mais indirectement en «rebondissant» sur d'autres). Permet par exemple d'obtenir des ombres moins franches.
- «**Min Roughness**»: Rugosité minimale, utilisé pour adoucir les reflets spéculaires (pour que ça ne brille pas trop)
- «**Shadow Bias**»: Accentue plus ou moins les ombres
- «**Shadow Filter Sharpen**»: Le filtre sharpen rend l'ombre plus «dure», moins floue en quelque sorte.
- «**Cast Translucent Shadows**»: Permet de créer ou non des ombres dynamiques à partir d'objets transparents.
- «**Affect Dynamic Indirect Lighting**»: pour choisir si elle participe ou non aux calculs dynamiques de lumière indirecte.
- «**Affect Translucent Lighting**»: pour choisir si elle affecte ou non la transparence des objets.

Quand aux autres paramètres, nous vous invitons à les découvrir par vous même car ils sont relativement difficiles à expliquer. Nous reverrons probablement plus en détails ces éléments lors d'un cas pratique dans les tomes suivants. Il y a beaucoup à dire sur les lumières, mais cela dépasse du cadre d'une introduction au sujet.

Mise en place d'un cycle jour/Nuit:

Pour cela, rien de plus simple: il suffit d'agir sur la rotation de Light Source et d'utiliser une fonction déjà disponible:

1. Dans **WO**, sélectionnez l'actor «Light Source», puis dans **DT**, cliquez sur «Movable»
2. Toujours dans **DT**, cliquez sur «Blueprint/Add Script»
3. Sélectionnez le répertoire «Blueprint» sous «Content» et renommez-le en «BP_Light_Source»
4. Ouvrez ce Blueprint en édition et allez dans l'«EventGraph»
5. Ajoutez une variable «SunSpeed» de type «Float» et de valeur 1.0. Cochez «Editable»
6. Ajoutez une variable «SkySphere» de type «Actor». Cochez «Editable» également.
7. Ajoutez les éléments de l'illustration 119
8. Revenez dans l'éditeur de niveau et sélectionnez l'actor «BP_Light_Source» dans **WO**
9. Dans **DT**, panel «Default», sélectionnez «Sky Sphere»: «BP_Sky_Sphere»
10. Dans **WO**, sélectionnez «BP_Sky_Sphere»
11. Dans **DT**, panel «Default», sélectionnez «Directional Light Actor» : «BP_Light_Source».

Nous ne pouvons pas utiliser une référence directe à l'objet «BP_Sky_Sphere» qui est dans la scène courante, puisque nous sommes dans un composant Blueprint et non dans l'«EventGraph» du niveau. C'est pourquoi nous passons par une variable «SkySphere» que l'on référence directement au niveau du Blueprint instancié, c'est à dire l'Actor.

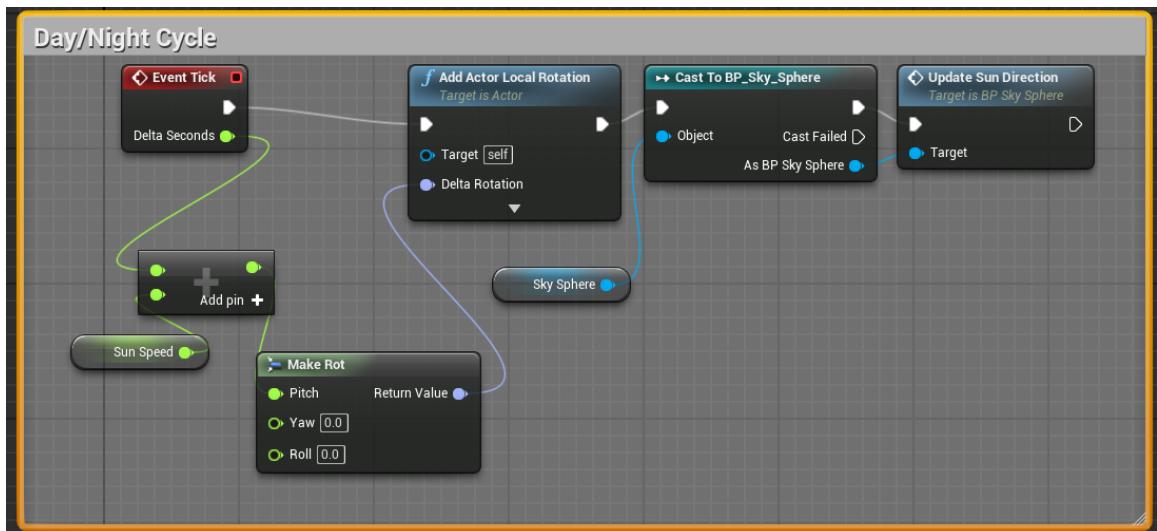


Illustration 119: Blueprint permettant une gestion du cycle Jour Nuit

Nous bénéficions ici de la fonction «Update Sun Direction» qui fait partie du Blueprint «BP_Sky_Sphere». C'est cette fonction qui déplace le soleil dans le ciel. Double-cliquez sur cette fonction pour observer son programme.

La valeur de «SunSpeed» est intentionnellement élevée pour se rendre compte du résultat. A vous de moduler cette valeur en fonction de vos besoins.

AJOUT D'UN EFFET DE BROUILLARD

Les effets de brouillard peuvent être utilisés pour ajouter une ambiance particulière comme donner l'apparence d'un cadre inquiétant. Unreal Engine peut gérer **2 types de brouillard**: le brouillard **atmosphérique** et le brouillard type **Exponential Height** basé sur la hauteur. Les brouillards de type volumétrique que l'on pouvait utiliser sous UE3 n'ont pas encore été portés vers la version 4 du moteur. Mais on peut toujours réaliser ce genre d'effet avec le système de particules que nous étudierons dans le tome 2.

Brouillard atmosphérique

Le **brouillard atmosphérique** donne une approximation de la diffusion de la lumière à travers une atmosphère planétaire. Cela peut donner à vos niveaux un look beaucoup plus réaliste. Les objets les plus lointains deviennent invisibles (en prenant la couleur du brouillard), ajoutant à l'illusion de profondeur.

Le brouillard atmosphérique est très adapté pour rendre l'effet que nous avons quand nous regardons des montagnes au loin et que ces dernières s'opacifient en prenant une couleur bleuté.

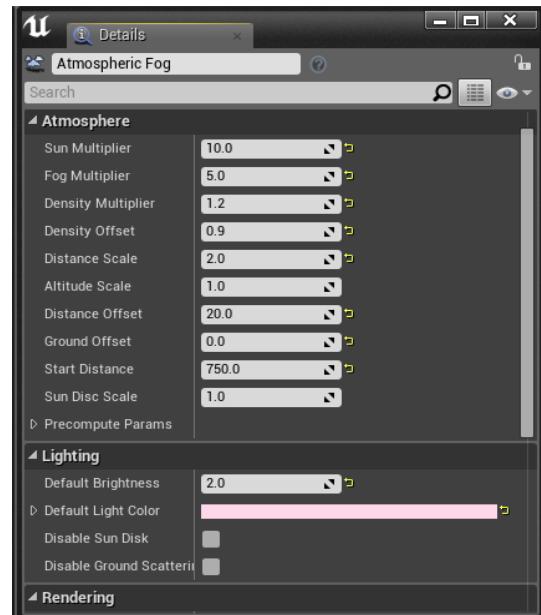


Illustration 120: Exemple de paramétrage

Voici quelques **paramètres** utiles:

- «**Sun Multiplier**»: pour multiplier l'effet de la lumière sur le ciel et le brouillard
- «**Fog Multiplier**»: n'affecte que la couleur du brouillard, pas la lumière.
- «**Density Multiplier**»: modifie la densité du brouillard
- «**Density Offset**»: contrôle la transparence du brouillard (-1 à 1)
- «**Distance Scale**»: pour modifier l'application du brouillard en fonction de la distance à la caméra.
- «**Altitude Scale**»: pour modifier l'application du brouillard en fonction de la hauteur. Ainsi, on peut jouer avec «**Distance Scale**» pour modifier l'action du brouillard sur le ciel par exemple.
- «**Distance Offset**»: pour utiliser sur de grandes distances, s'exprime en km.
- «**Ground offset**»: à quel niveau (Z) est situé le sol
- «**Start Distance**»: distance à partir de laquelle le brouillard commence à faire effet
- «**Default Brightness**»: intensité de lumière utilisée par défaut quand il n'y a aucune autre lumière dans le niveau (en lumens).
- «**Default Light Color**»: couleur utilisée pour l'effet de brouillard
- «**Visible**»: active ou désactive l'effet.
- «**Hidden in Game**»: permet de ne pas être utilisé dans le jeu si coché.

Pour faire un essai, utilisez le paramétrage de l'illustration 120, vous obtenez un résultat proche de celui-ci (couleur **Default Light Color** : RGB = {1,0.7,0.8}):



Illustration 121: Paysage avec le brouillard atmosphérique paramétré comme ci-dessus

Brouillard type Exponential Height

Le brouillard type **Exponential Height** est plus simple à paramétrier, nous vous laissons le soin de faire les tests vous-même. Il est utilisé pour réaliser un simple brouillard, dense.

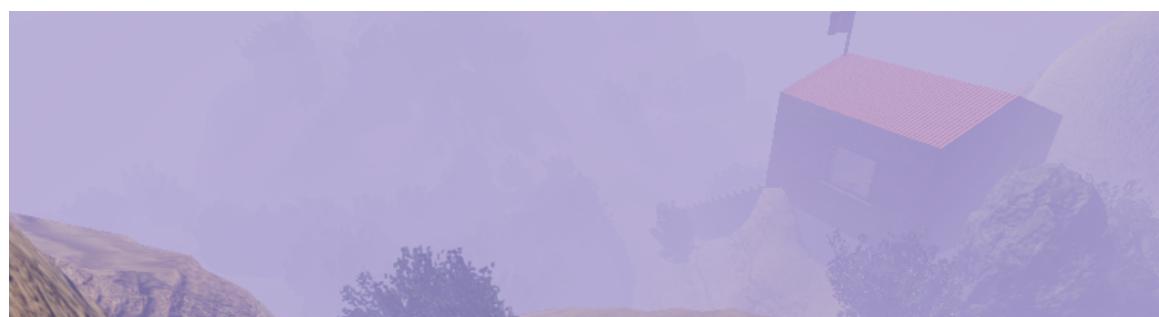


Illustration 122: Brouillard de type Exponential Height

MISE EN PLACE DU JOUEUR

Vous deviez vous demander quand nous pourrions courir dans ces plaines et découvrir le paysage à un autre niveau? Ce n'est pas par hasard si nous avons préféré traiter de ce sujet seulement à la fin. Car, s'il est aisément de mettre en place un personnage grâce aux **templates** d'UE4, cela demande certaines compétences dès qu'on souhaite aller au-delà de la simple course à pieds dans les plaines.

Importation du template «Third Person»

Avec les anciennes versions, nous aurions été obligé de passer par la création d'un nouveau projet et la migration des éléments relatifs au template, mais désormais, nous pouvons le faire directement:

1. Dans **CB**, cliquez sur «Add New», puis sur «**Add Feature or Content Pack**»
2. Sélectionnez le Template «**Third Person**» et cliquez sur «**Add to Project**» (comme sur l'illustration 123)

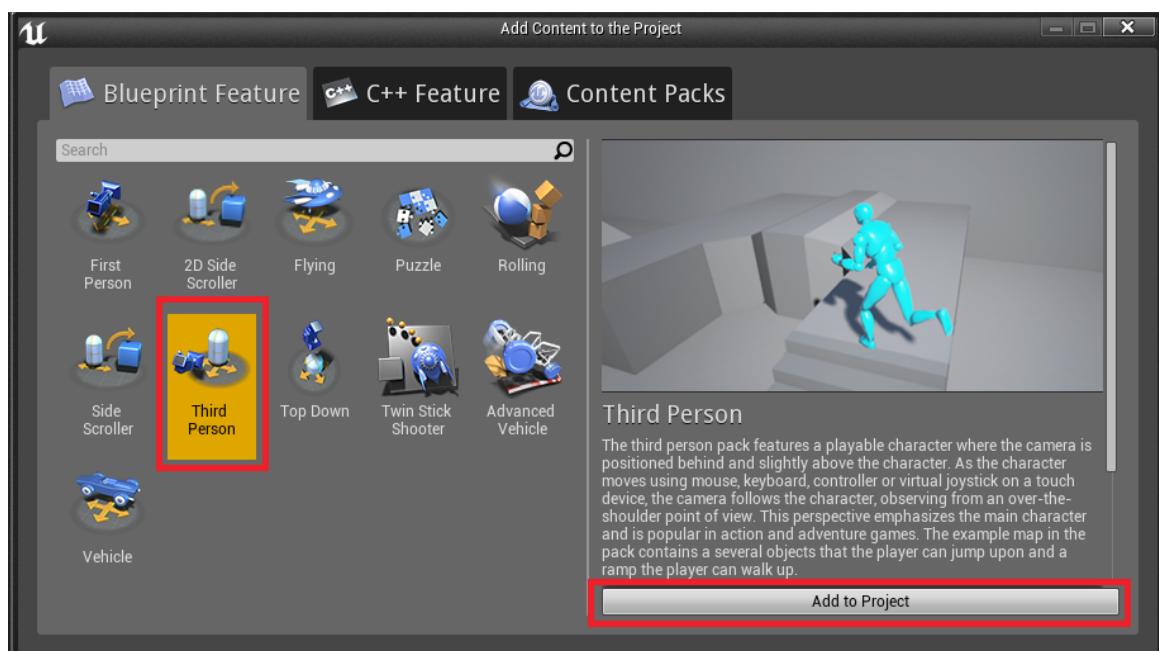


Illustration 123: Ajout du template "Third Person"

Le personnage est composé d'un static mesh «**ThirdPersonSkelMesh**» (répertoire «Character») auquel est associé un squelette «**ThirdPersonSkeleton**» de déformation pour les animations.

Et enfin, il dispose de ses propres «Blueprints»:

- un pour le personnage: «**ThirdPersonCharacter**» (répertoire «Blueprint»)
- un autre pour l'animation: «**ThirdPersonAnimBlueprint**» (répertoire «Character»)

Plusieurs animations ont été importées:

- «**Idle**» (le petit mouvement que fait le personnage car il ne fait rien d'autre),
- «**Jump_**» (saut),
- «**Run**» (course)
- et «**Walk**» (marche).

Tout ce qu'il faut à la base pour un personnage. Le personnage est accompagné également de **sa propre caméra**.

Nous aurions pu tout aussi bien prendre un personnage de type jeu de tir en vue subjective en utilisant le template «First Person». Tout comme nous aurions pu prendre le template «vehicle» pour contrôler une voiture. L'utilisation des templates permet de gagner un temps précieux pour le maquettage des projets.

3. Ouvrez le projet principal «TestProject»
4. Dans **CB**, répertoire «Blueprints», ouvrez le fichier «BP_MyGameMode» en édition
5. Dans le panel «Classes», rubrique «Default Pawn Class», sélectionnez «ThirdPersonCharacter»

A la place de l'actor «Player Start», c'est le Blueprint «ThirdPersonCharacter» qui est utilisé pour le «spawn», c'est à dire, la génération du joueur.

6. «Compilez», puis «Sauvez»

A partir de maintenant, quand on lance le jeu, le personnage apparaît, mais nous serons incapables de le contrôler.

Contrôler le personnage:

7. Ouvrez les «Project Settings» et rendez-vous dans la section «Engine», puis cliquez sur «Input»
8. Ajoutez les différentes entrées comme sur l'illustration 124.

Let's Play!

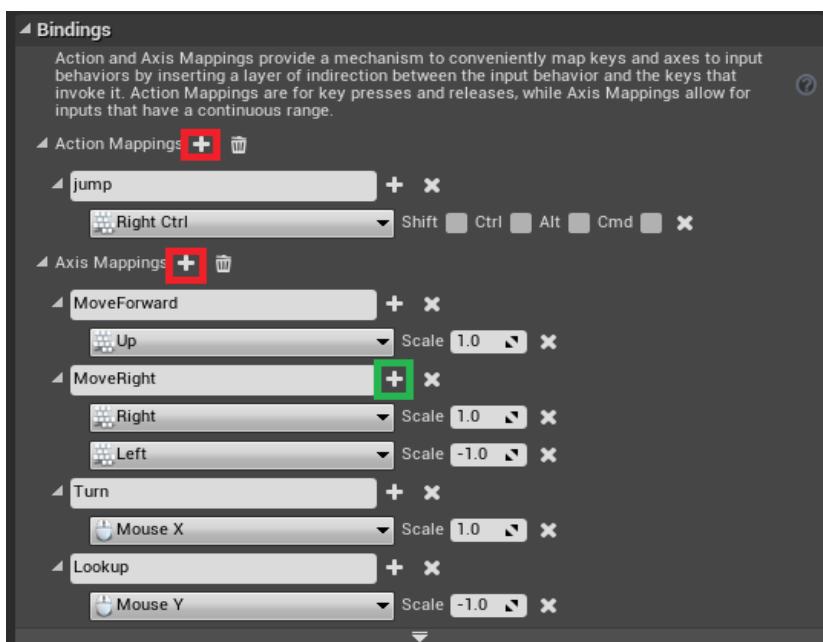


Illustration 124: Paramétrage des touches (inputs) et de la souris

Pour ajouter une «action mappings» ou un «axis mappings», cliquez sur le bouton «+» (encadré rouge) pour chaque nouvelle entrée.

Il est obligatoire de respecter les mêmes noms d'action comme «MoveForward» (avancer) car ils sont utilisés dans le Blueprint du personnage, nous y reviendrons.

«MoveRight» possède 2 entrées, il faut donc utiliser le bouton «+» de l'encadré vert pour la seconde touche.

Le mouvement de la caméra est réalisé grâce aux actions «Turn» et «Lookup». Encore une fois, c'est parce que le Blueprint a été programmé pour cela.

Normalement, avec un contrôleur standard («PlayerController»), la souris est capturée et il suffit ensuite de bouger cette dernière pour obtenir un mouvement de caméra. Mais avec notre contrôleur actuel qui laisse le curseur apparent et disponible pour des actions, il faut cliquer tout en bougeant le souris pour que cela fonctionne.

Vous pouvez vous déplacer dans le décor, sauter pour grimper le long de parois trop abruptes... Le paysage que vous avez créé de toutes pièces prend alors une consistance qu'il n'avait pas avant: les crevasses semblent réellement infranchissables et le monde semble bien plus vaste à explorer que vu du dessus.

Vous avez déjà de quoi vous amuser en programmant de nombreuses interactions avec le décor: de nouveaux bâtiments fermés avec des codes secrets ou des cartes magnétiques, des plates-formes qui se dérobent à votre passage, une boule géante en pierre qui dévale vers vous comme dans Indiana Jones...

Oui, il y a encore tellement de domaines que vous avez envie d'explorer: vous souhaiteriez créer votre propre personnage, ajouter des ennemis ou des oiseaux qui volent dans le ciel. Nous verrons tout cela dans le tome 2.

Notre jeu, pour être complet, doit posséder un menu, permettre le chargement et la sauvegarde de parties, gérer un inventaire, passer d'une carte à une autre. Il faut travailler la sonorisation pour une immersion plus importante, ajouter des musiques d'ambiance, des cinématiques («cute scenes»)... vous souhaitez peut-être exporter votre jeu vers les mobile ou sous forme web ? Et pourquoi ne pas ajouter un mode multijoueur? Certains d'entre vous souhaitent probablement travailler en C++ et créer des classes réutilisables. Il y a de nombreux domaines que nous vous proposons d'explorer dans les tomes suivants.

A Suivre ...

Les **sources complètes** du projet développé dans ce tome sont disponibles à cette adresse: <http://bit.ly/1Je5L1P>



ANNEXES



5

Remerciements

Tout d'abord j'aimerais remercier **Tim Sweeney** et toute son équipe pour avoir créé un aussi merveilleux outil et l'avoir partagé avec le monde.

Merci aux youtubers Meletou, 4music2junkie0, Chad, Falaranah, MetalGameStudio, Muhammad, RealDave, SatheeshPv, Teampro, Tesladev, Weaver Games, zoombapup et tous les autres que je ne pouvais citer ici, mais qui ont contribué, briques après briques à m'apporter un éclaircissement ou deux sur certains aspects de l'éditeur ou sur la conception de jeu.

Merci à Gege et Michel et à «mémère la bouffe» pour leur soutien moral et logistique alors que mon matériel tombait en panne.

Merci à Jean-michel Martin de Santero de TLK Games pour m'avoir permis, alors que je n'étais qu'étudiant, de sortir mes premiers jeux vidéo, mais surtout, pour m'avoir donné l'envie d'entreprendre.

Merci à tous mes amis et contacts qui suivent régulièrement mon blog et qui m'encouragent dans ce domaine vaste et passionnant de la création de jeux.

Et enfin, plus particulièrement, un grand merci à ma compagne pour son travail de relecture, pour ses idées et son soutien inconditionnel.

Table des index

Ambient occlusion.....	64	Megascans.....	61
Base Color.....	63, 65, 69,-78, 84, 96	Metallic.....	63, 65, 67
Branch.....	51, 57	Miroir.....	6, 63-69
breakpoint.....	6, 43	Normal mapping.....	21, 63, 79
Brouillard.....	7, 25, 115-116	Opacity.....	63, 75, 76, 113
Classe Blueprint.....	41, 53, 54	Open World.....	5, 7, 111
Clear Coat.....	64	Panner.....	78, 82
Collision.....	5, 17, 47, 52, 107, 109, 110	Project Settings.....	16, 26, 118
Construction Script.....	36	Quixel.....	61
Culling.....	88, 89, 108, 109	Ramp.....	90, 93
Debug.....	6, 41, 44	Refraction.....	64, 81, 82
Diffuse map.....	66	Rotator.....	55, 78
DirectX.....	9, 79	Roughness.....	63-67, 114
Editable.....	51, 55, 114	Shadow.....	109, 114
Editeur de niveau ..	6, 13, 14, 23, 36, 40, 45-52, 60, 65, 70, 74, 77, 83, 84, 97, 110, 114	Specular.....	63
Emissive Color.....	63, 77, 78	Splines.....	89-90, 94-95
Enums.....	55	Structure.....	4, 27, 28, 55, 56
FlipFlop.....	58	Subsurface color.....	4, 64, 81
Foliage.....	5, 7, 16, 102, 104-107, 109, 110, 112	Subsurface scattering.....	81
ForEachLoop.....	56, 57	Tableau.....	56, 57
ForLoop.....	58	Tangent Space Normal.....	103
Gate.....	5, 9, 58	Tangent To World.....	99
Heightmap.....	87, 89, 90, 108	Template.....	5, 7, 14, 25, 117, 125
HSV.....	64	Tesselation.....	63, 64, 79-81
Landscape.....	7, 16, 87-90, 94, 96, 97, 106-108, 111, 112	Texture Coordinate.....	69, 76, 78
Layer5, 7, 17, 25, 29, 45, 50-52, 88-91, 96-100, 107-108, 118		Texture Sample.....	66-67, 98, 100
Lerp.....	72-74, 98-99	Transparence ..	32, 71, 75-76, 102-106, 114-116
Level Blueprint.....	16, 45, 53, 54	UDK.....	9, 13
Light.....	15, 21, 24-25, 31, 36, 40-46, 55, 77-81, 109, 113-116	UV-Mapping.....	32
Lightmap.....	31, 78, 109	Vector.....	55, 64, 72, 77, 82, 84, 99
LOD.....	80-81, 88, 105, 109, 112	vertex painting.....	6, 70, 74
Material Function.....	82, 98	Volume Trigger.....	35
Matinee.....	4, 6, 11, 17, 47-52	WhileLoop.....	58
		World Displacement.....	4, 63, 79
		World Position Offset.....	63, 84, 104

Dans la même collection

The image shows the cover of the second volume of the Unreal Engine developer's notes. The title "Les cahiers du développeur Unreal Engine" is written vertically on the left side of the cover. At the top, it says "Grégory Gosselin De Bénicourt". Below that is a screenshot from a game showing a character in a blue jacket and black shorts holding a gun, with several other characters and a wireframe model in the background. The subtitle "Personnages, Intelligence Artificielle & Particules" is centered above a grid of smaller images. One image shows a character on a cube, another shows a character on a terrain, and a third shows a behavior tree diagram. The text "Versions 4.7+" is at the bottom left, and "Tome 2" is at the bottom right. The publisher logo "Editions Graziel" is also present.

Dans ce second tome, nous nous concentrerons sur l'animation des personnages et la programmation de leur intelligence, tout en continuant le jeu commencé avec le premier tome.

Au sommaire: le Template «Third Person», l'éditeur de mesh et d'animations, les différents types d'animation (morphing, skeletal bone, Blend Spaces, etc.), les Blueprints d'animation, la création de personnages avec MakeHuman, la simulation d'objets souples (Apex/PhysX), la gestion des projectiles, la création de dommages, la destruction progressive d'objets, la gestion basique d'inventaire et le ramassage d'objets. Puis nous verrons comment mettre en place des Personnages Non Joueurs (PNJ), et programmer leur intelligence: Faire des rondes sur le terrain, détecter les joueurs, partir à leur poursuite et attaquer, le tout au travers des Behaviour Trees. Enfin, nous étudierons les systèmes de particules et créerons plusieurs effets, ainsi que des «decals»



Prix éditeur: 14,90€ TTC disponible sur www.graziel.com

Du même auteur

Longtemps réservée à un cercle de programmeurs passionnés, la 3D peine encore à s'inviter au sein des petits studios de jeux indépendants, et pour cause: le ticket d'entrée est relativement élevé. **Blender** (logiciel **gratuit** et **Open Source**) permet de modéliser, d'animer, de faire un rendu 3D et surtout d'être utilisé comme moteur de jeu. C'est l'outil parfait pour le **débutant**, mais également pour celui qui veut créer un jeu de **niveau commercial**. Blender est également un très bon complément à des outils comme Unity, CryEngine, Unreal Engine et autres moteurs commerciaux.

Grâce aux ressources librement à la disposition de tous sur Internet, nul besoin aujourd'hui d'être **graphiste** ou **programmeur** pour créer un jeu.

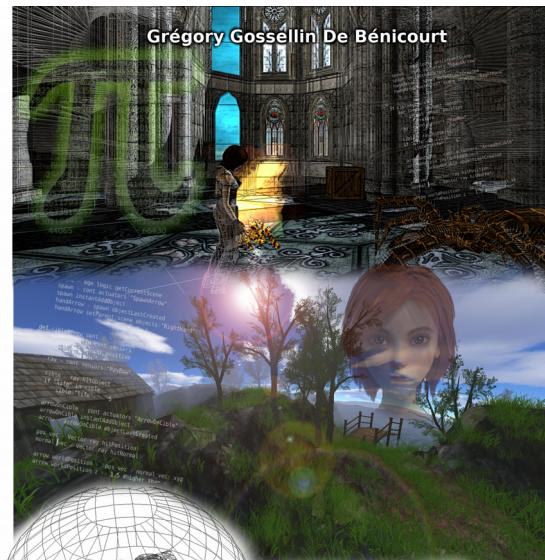
Vous n'avez pas envie de lire 400 pages de théorie sur la modélisation, les animations et la programmation python ? Tant mieux, **ce livre est fait** pour vous. Nous vous proposons de rentrer directement dans le vif du sujet avec **plusieurs projets de jeux** : un jeu de **plate-formes** (comme Super mario), un **First-person Shooter** (comme Doom, Far Cry ou Half-Life), un **Third-person Shooter** (comme Tomb Raider, GTA ou Watch Dogs), un bac à sable (comme **Minecraft**), ainsi qu'une **course de voitures** et un **simulateur d'avion**. Au delà de ces projets, une **centaine de recettes** vous permettront d'attaquer n'importe quel type de jeu.

Si vous n'êtes pas encore un mordu, le virus de la 3D va s'insinuer doucement en vous. Il est bien **plus amusant de concevoir** un jeu, que de jouer au tout dernier blockbuster. Vous allez devenir l'architecte d'un monde nouveau qui obéira à toutes les règles que vous fixerez. En fin de compte, quelle **meilleure utilisation de l'ordinateur** peut-on entrevoir que celle permettant de créer son propre univers et de le voir vivre et se développer sous ses yeux ? Les seules limites étant celles de notre imagination...

Passionné de jeux vidéo et de 3D, l'auteur a voulu rendre **hommage** à tous ces jeux qui ont révolutionné le domaine.

A propos de la collection

«Rendre les **techniques utilisées par les professionnels** accessibles à tous». Les livres de cette collection sont organisés de la façon suivante : des **projets**, des **recettes** sur lesquelles s'appuient ces derniers, et un ensemble de **fiches** pour les débutants et ceux souhaitant parfaire leurs connaissances. «Nombre de livres techniques sont abandonnés aujourd'hui dès les premiers chapitres car l'acquisition du savoir théorique demande un investissement personnel trop important». En entrant directement dans le **vif du sujet**, nous souhaitons **faciliter l'apprentissage des techniques**.



Prix éditeur: 29,90€ TTC disponible sur www.graziel.com

Vos notes

Présentation de la collection

Unreal Engine est probablement l'un des moteurs de jeu les plus aboutis de notre époque. Il est accompagné d'un éditeur à la fois **simple et puissant**, qui le rend accessible à un public non-développeur: artistes, architectes, ingénieurs, étudiants. A l'origine orienté vers les **grosses productions de jeux vidéo**, il s'est doté dernièrement d'un **nouveau système de licence** très attractif: il est désormais **gratuit** à l'utilisation, seuls 5% de royalties sont demandées sur les jeux et applications développées avec le moteur.

Les projets que l'on peut développer dépassent de loin le secteur du jeu: présentations interactives temps-réel, cinématiques, ... là où il y a une histoire à raconter, un produit ou un concept à présenter, un appartement témoin à faire visiter sur Internet, voir un concert «Next Gen» dans un univers virtuel en utilisant un casque de réalité virtuelle ! Avec Unreal Engine, on peut **exporter sa création** vers Windows, Linux/Steam OS, OSX, Android, iOS, XBOX One, Playstation 4, Oculus Rift... et même directement vers le web en HML5/WebGL.

Unreal Engine possède une **documentation abondante**, voir impressionnante. De très nombreux sujets y sont traités, mais cette pléthore d'informations rend aussi la découverte de l'outil très difficile, tant la couverture fonctionnelle du produit semble interminable. Et quand on se penche sur un thème précis, on se retrouve inévitablement en manque d'informations, ce qui peut paraître paradoxalement.

Ce que nous vous proposons est une **approche parallèle** à cela: un ensemble de **plusieurs cahiers** qui vous guideront dans la création de tel ou tel type de jeu, sans avoir la prétention de se substituer au manuel officiel. Au delà de la création des jeux, nous explorerons en détail tel ou tel aspect de l'outil, en faisant un point complet sur la question, comme «la création de cinématiques» (les «cute scenes») ou «la gestion des effets spéciaux». Chaque cahier peut être lu individuellement, et **selon vos besoins**.

Contenu de ce cahier

Dans ce premier tome, nous allons vous aider à faire vos **premiers pas sous l'éditeur** en créant votre premier jeu sous Unreal Engine.

Au sommaire: les fonctions de **modélisation** de l'éditeur, la création de **composants**, la **programmation Blueprints**, l'éditeur de cinématique **Matinee**, la création de **matériaux** (les textures, les normal maps, la transparence et les masques, les matériaux émissifs, les matériaux animés et dynamiques, les déformations via le World Displacement, les Subsurface colors, la réfraction, les fonctions de matériaux, les instances, etc.), la **modélisation de paysages** et l'outil «**foliage**».

Pour terminer, nous intégrerons un **personnage** au paysage – avatar que nous pourrons contrôler pour explorer le paysage que nous venons de créer et explorer le bâtiment que nous avons construit.

Prix Editeur: 14,90€ TTC
ISBN 979-1-09-384602-0

