Microsoft & MAGIC MAKERS

présentent

ImagineCup Junior

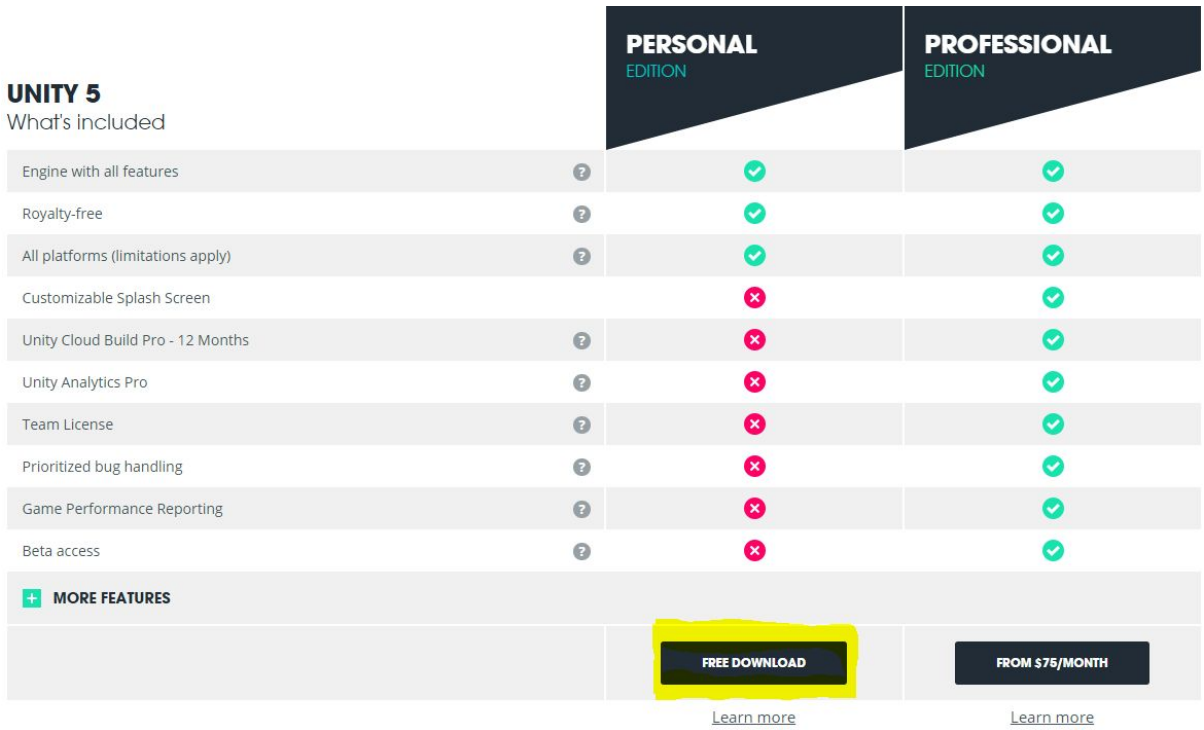
Tutoriel Complet Unity - Défi 1

0 - Découverte du terrain de jeu

0.1 - Téléchargement d'Unity3D

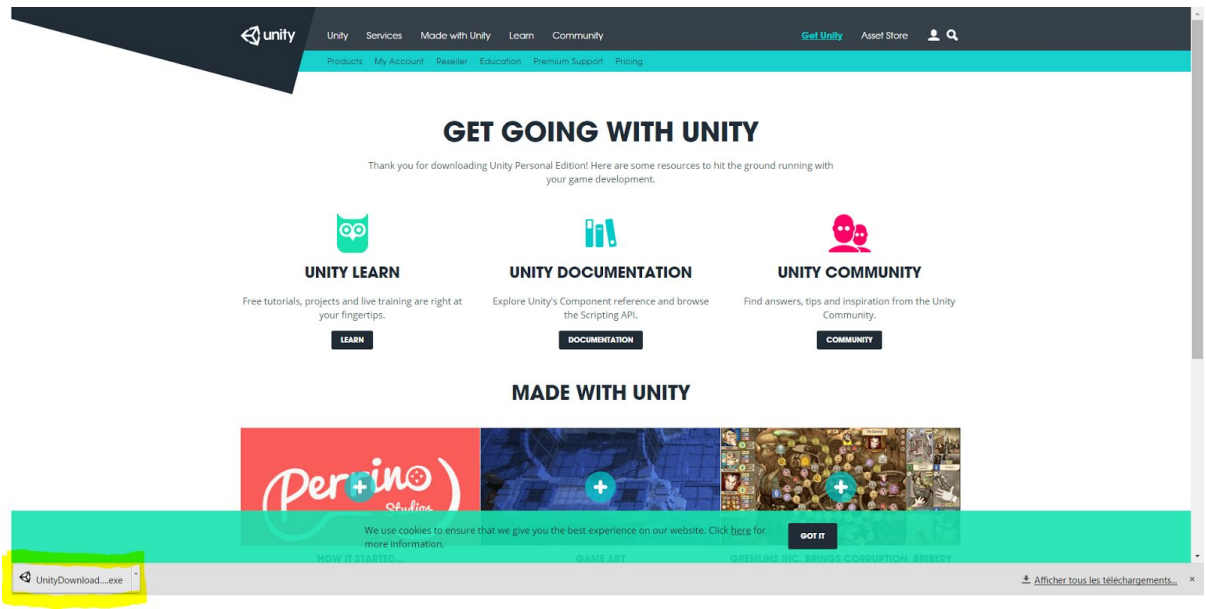
1. Aller sur le site d'Unity pour télécharger le logiciel via le lien suivant : https://unity3d.com/get-unity

1. Télécharger l'édition PERSONAL qui est totalement gratuite, en cliquant sur le bouton FREE DOWNLOAD"

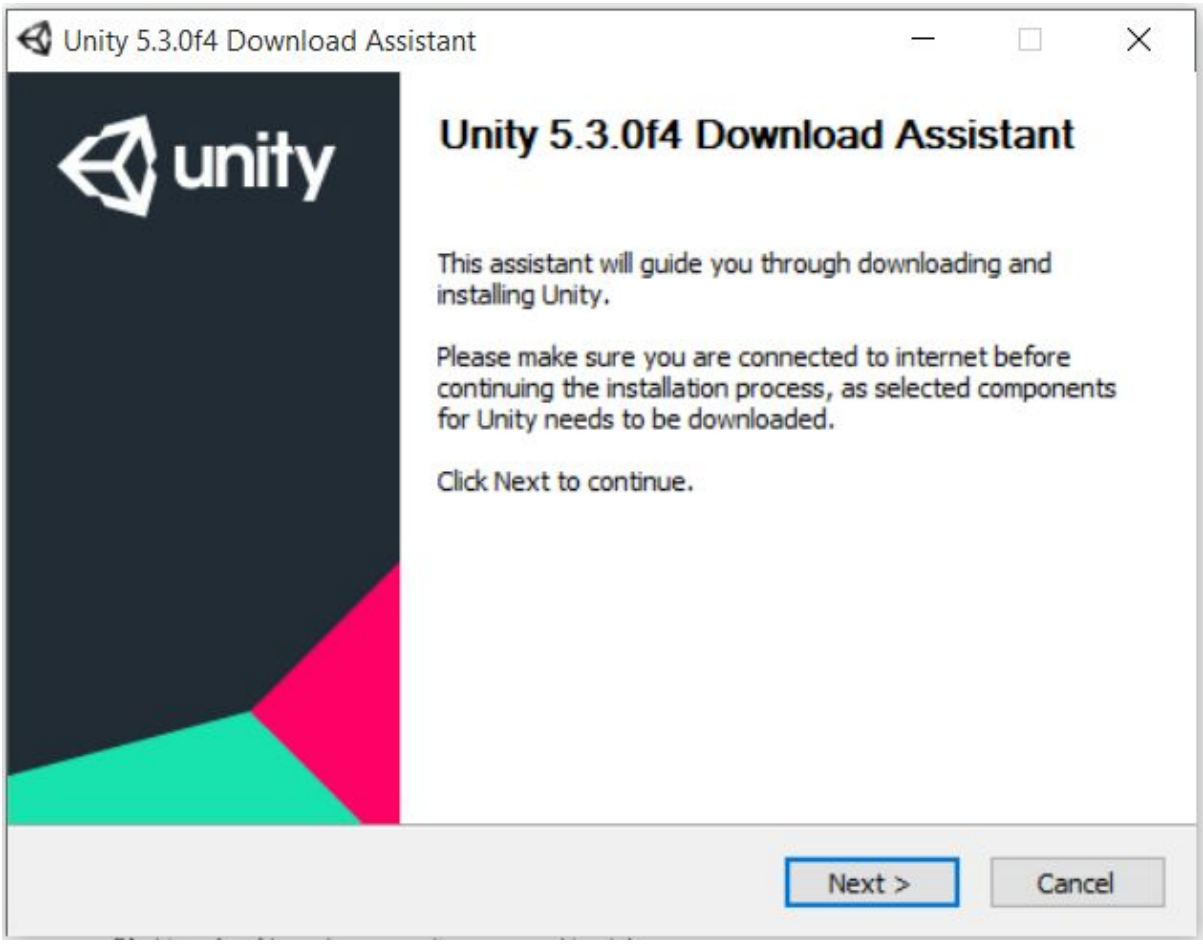


1. Cliquer sur le bouton DOWNLOAD INSTALLER.

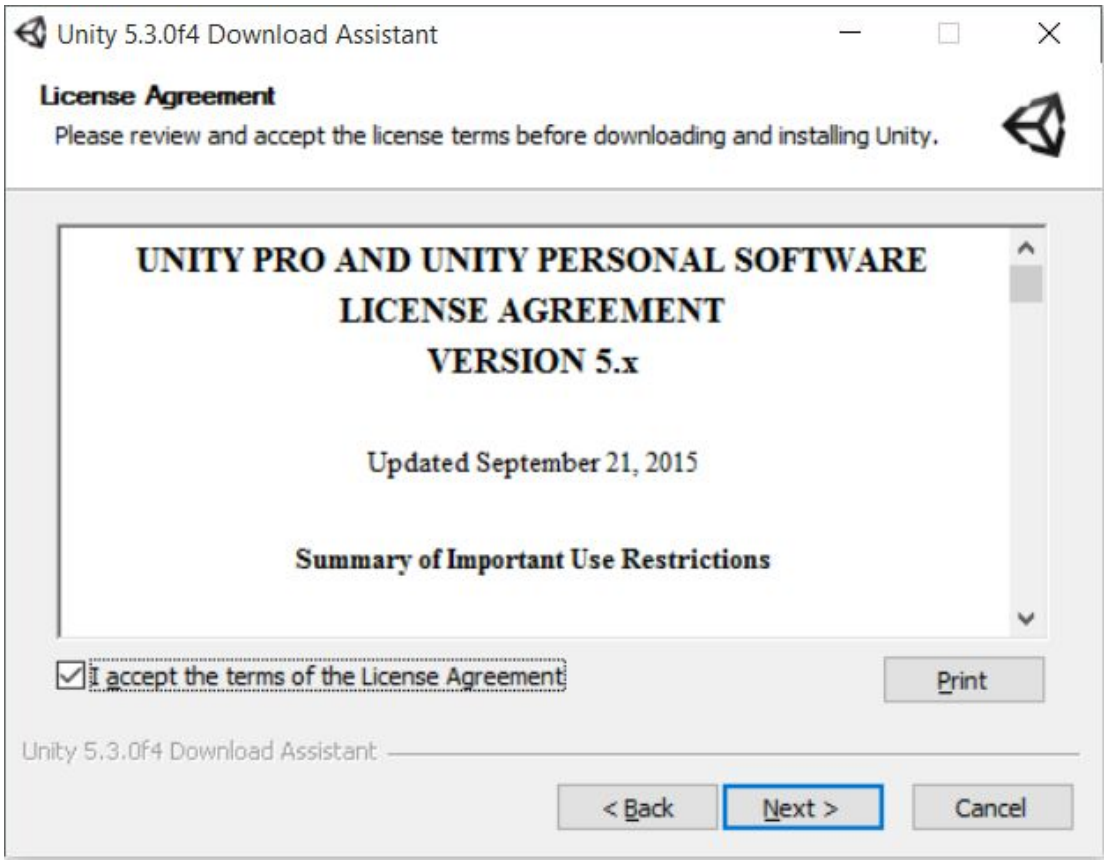
1. Une fois téléchargé, cliquer sur UnityDownloadAssistant.exe en bas du navigateur.



1. Une fenêtre s'ouvre, cliquer sur Next >

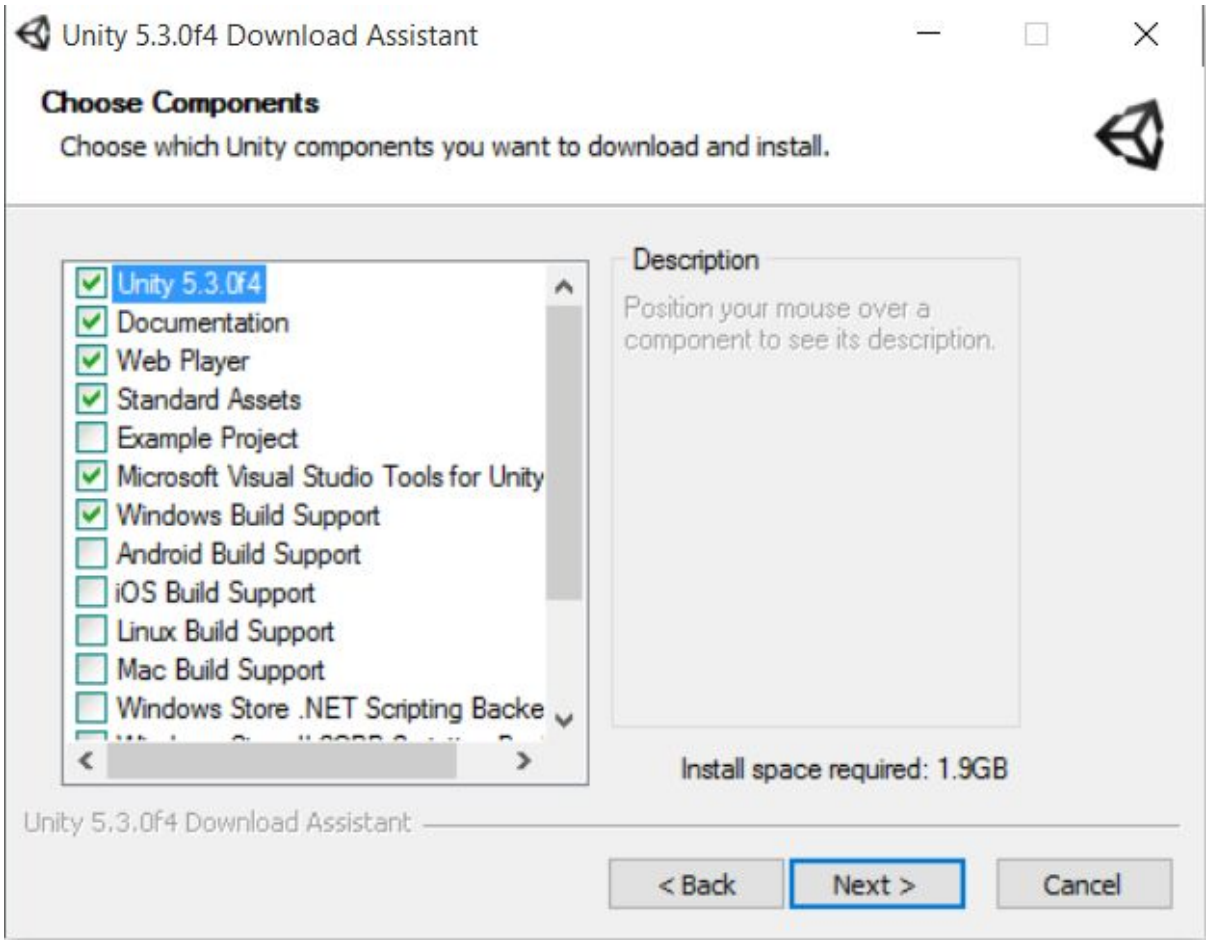


1. Cocher la case “I accept the terms of the License Agreement” puis cliquer sur Next >

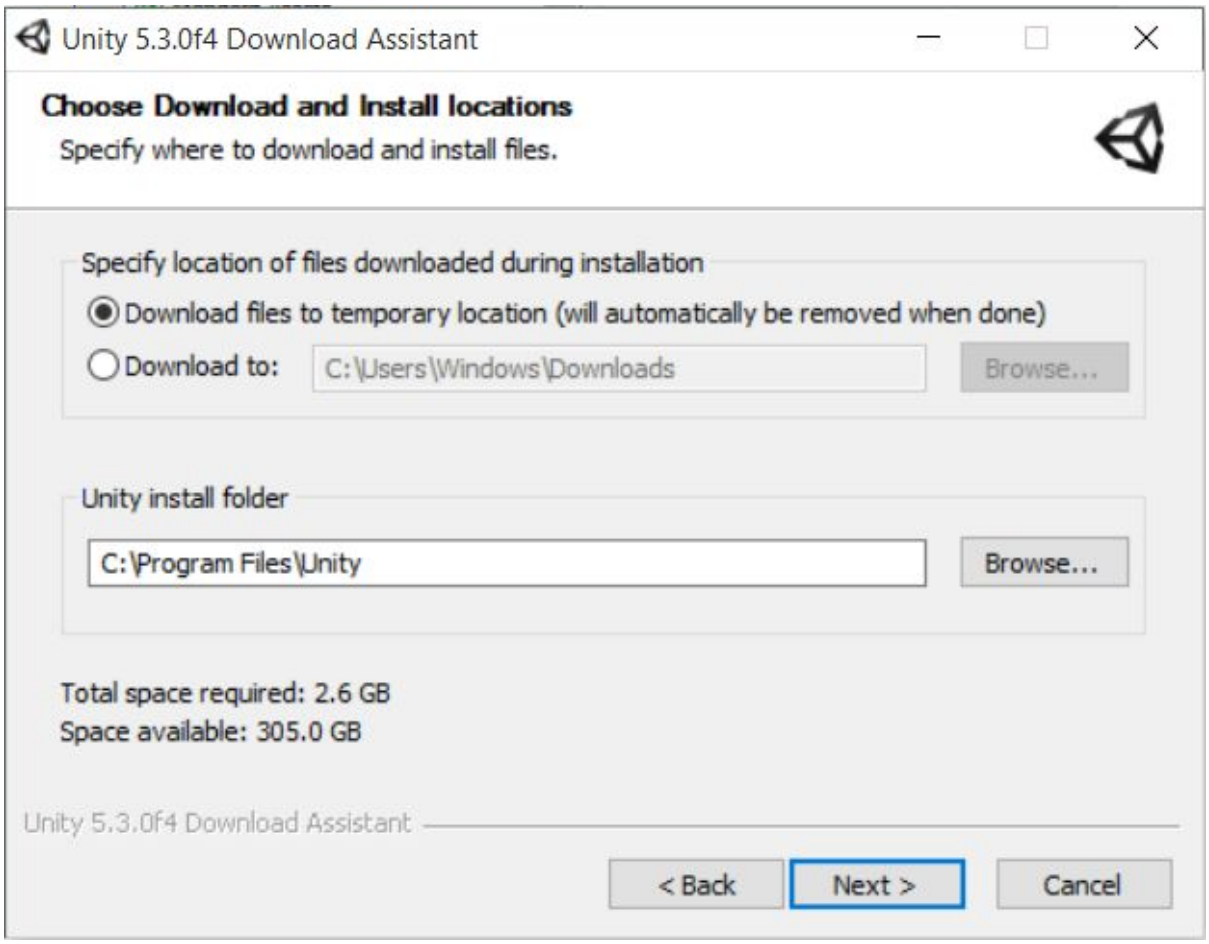


1. Cliquer sur Next

1. Pensez à cocher la case WebGL ce qui nous permettra d'exporter le jeu sur Chrome, Firefox et autres navigateurs.



1. Cliquer sur Next >



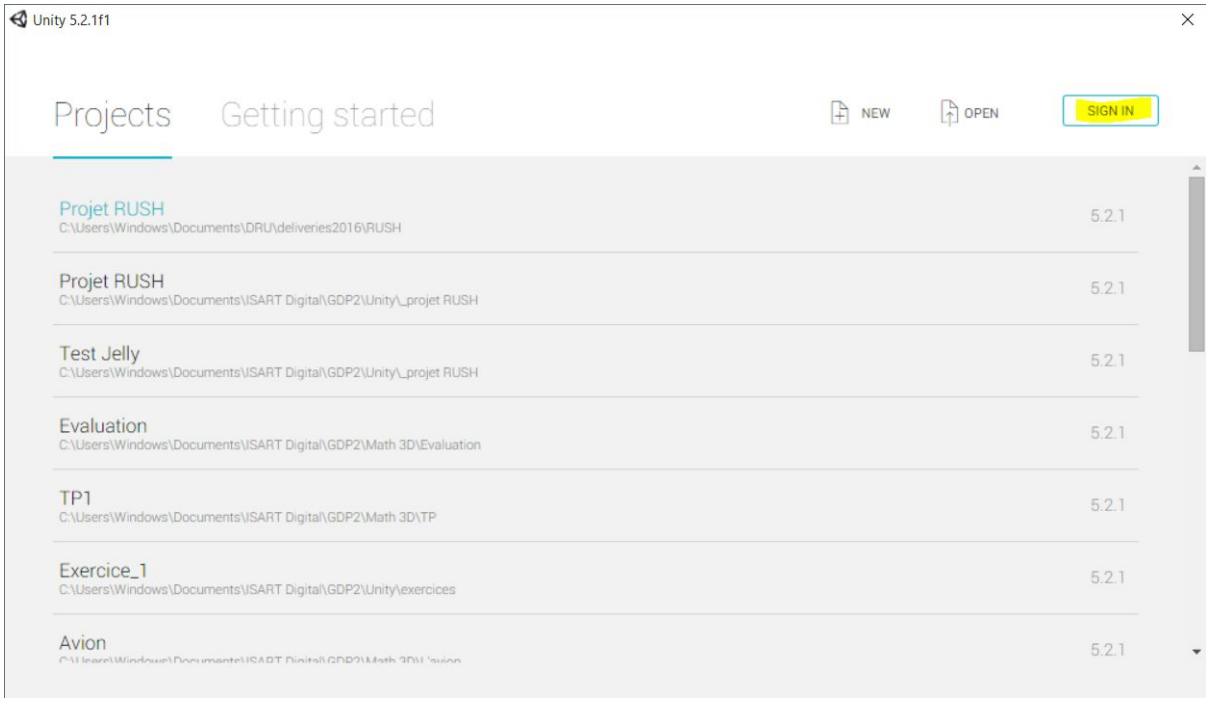
1. Cocher la case “I accept the terms of the License Agreement” puis cliquer sur Next > si une fenêtre vous le demandant apparaît.

1. Le téléchargement se lance ! A la fin de ce dernier, cliquer sur Finish. Unity est installé sur votre ordinateur !

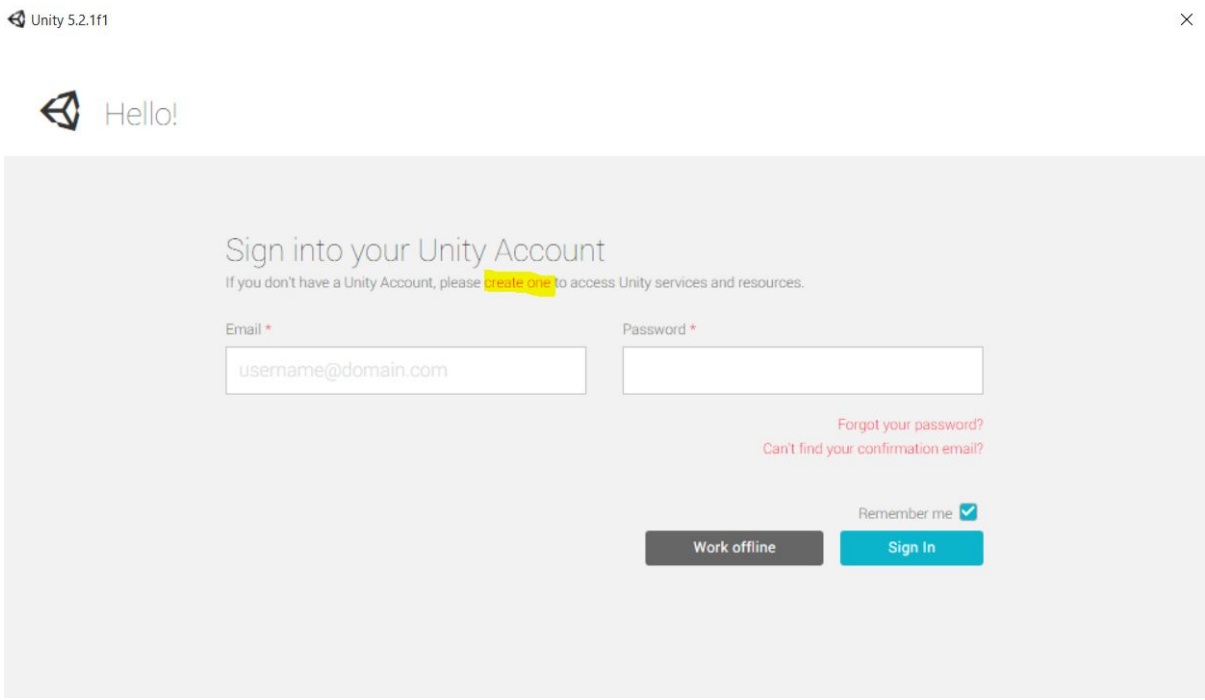
0.2 - Création de son compte

1. Lancer Unity

1. Cliquer sur le bouton SIGN IN

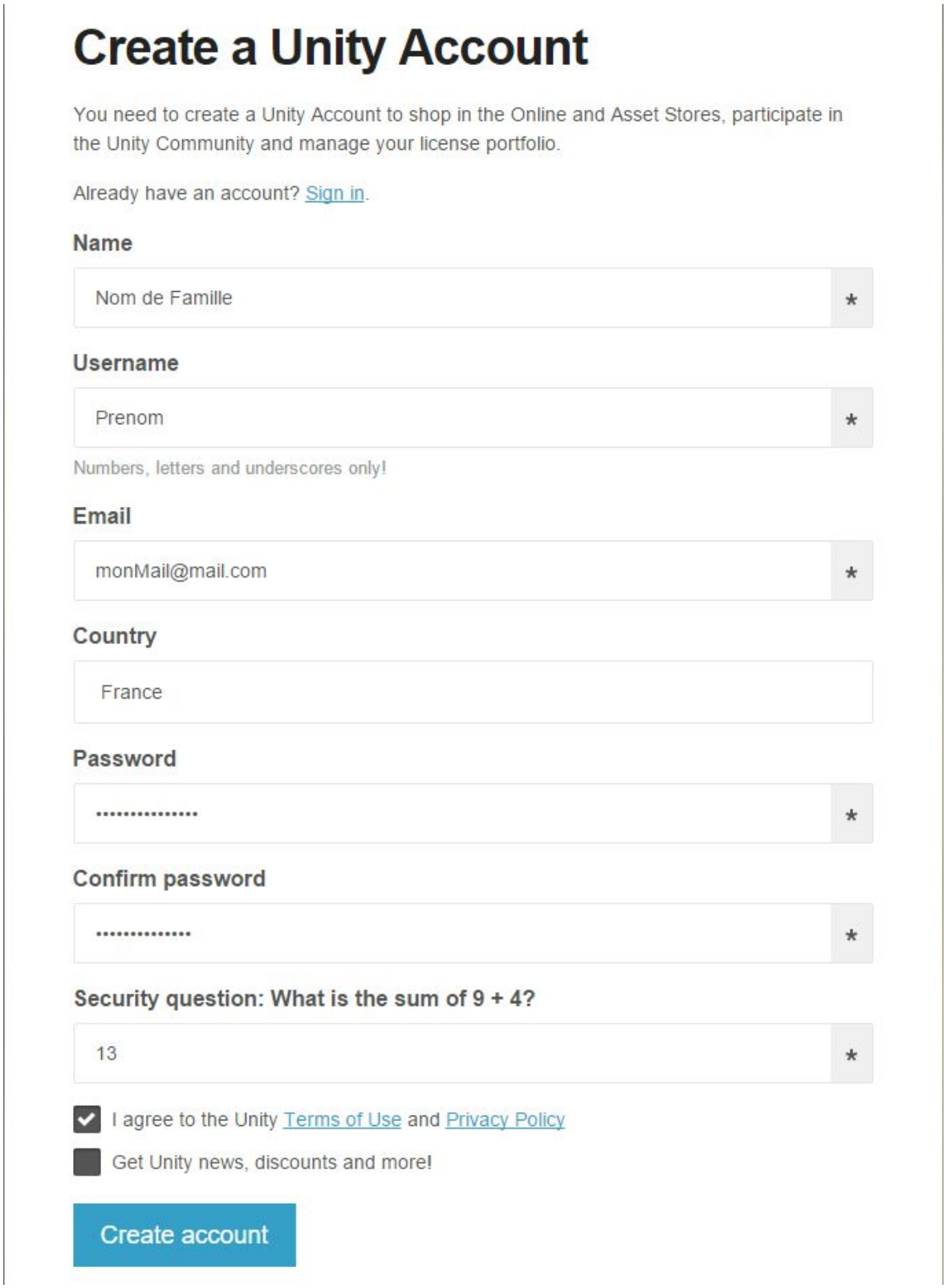


1. Si vous n’avez pas de compte avec lequel vous identifier, Unity vous propose de vous en créer un. Cliquez sur create one.

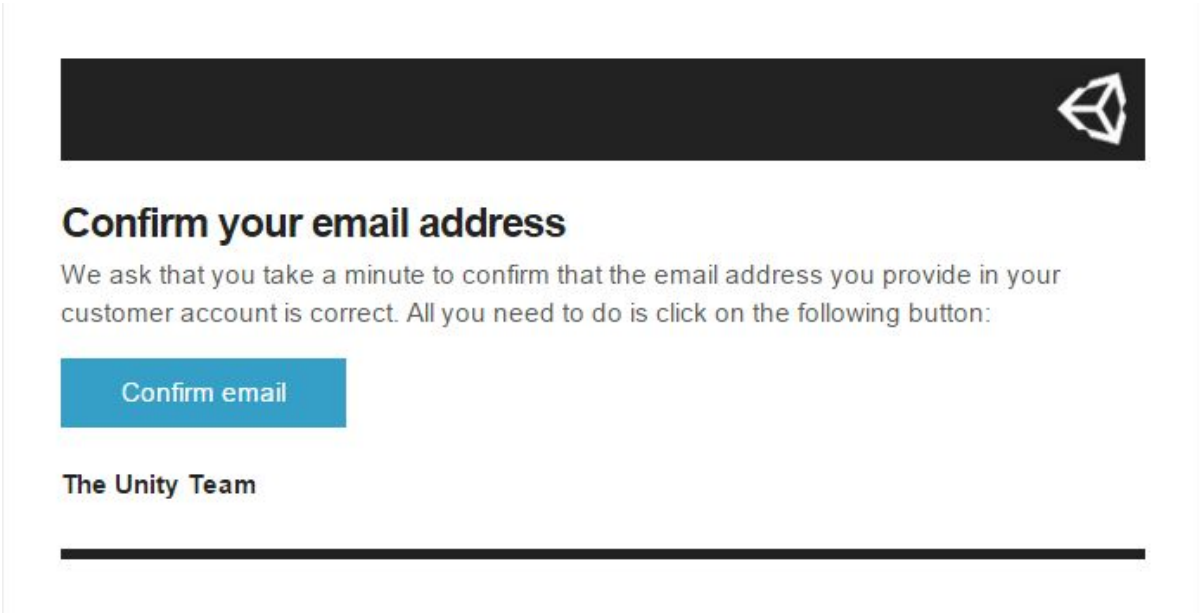


1. Unity vous ouvre alors une page internet où vous devez renseigner toutes les informations relatives à la création de votre compte.

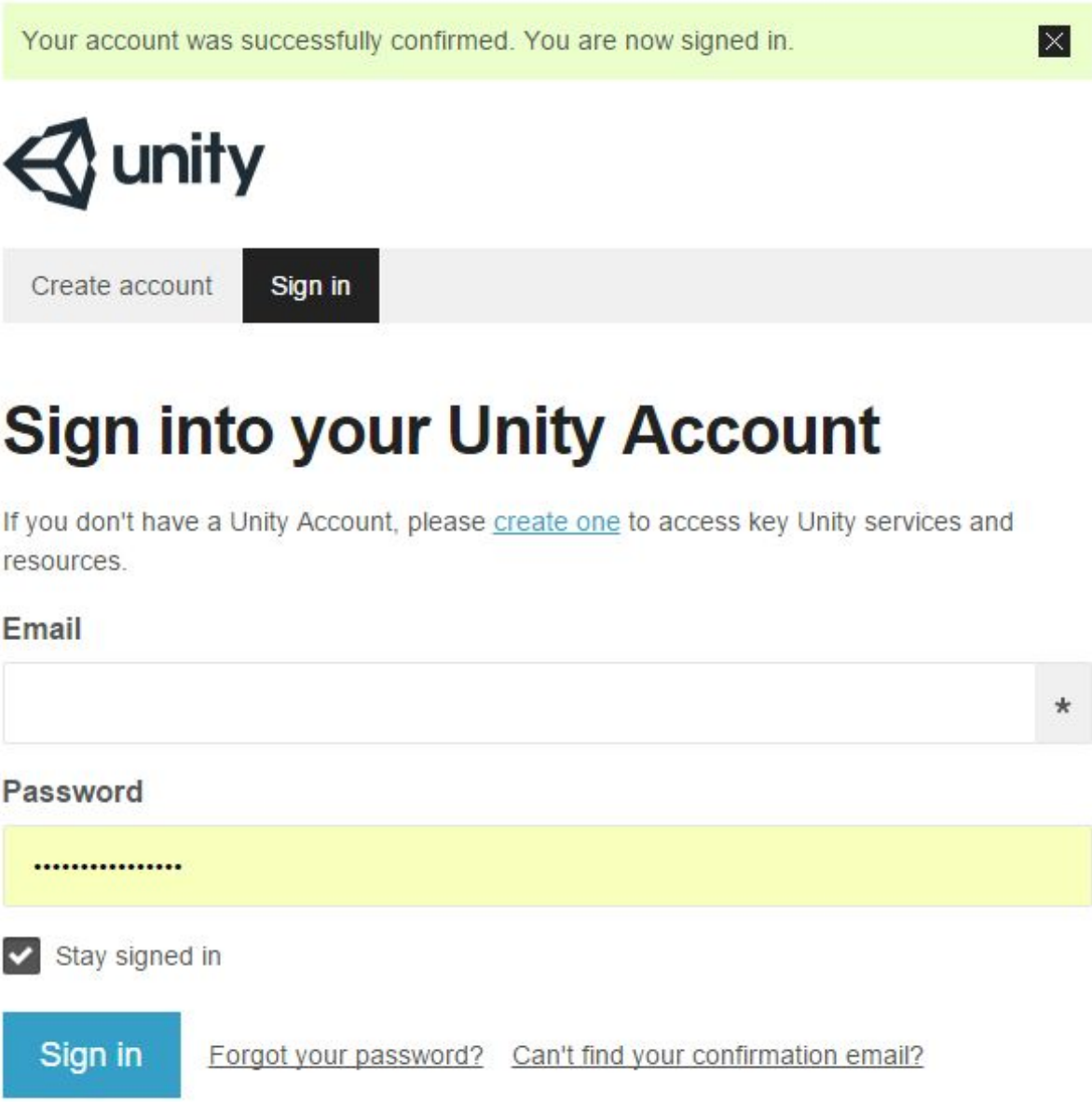
À noter que pour le mot de passe, il doit faire 8 caractères minimum, avec au moins une majuscule, une minuscule et un chiffre



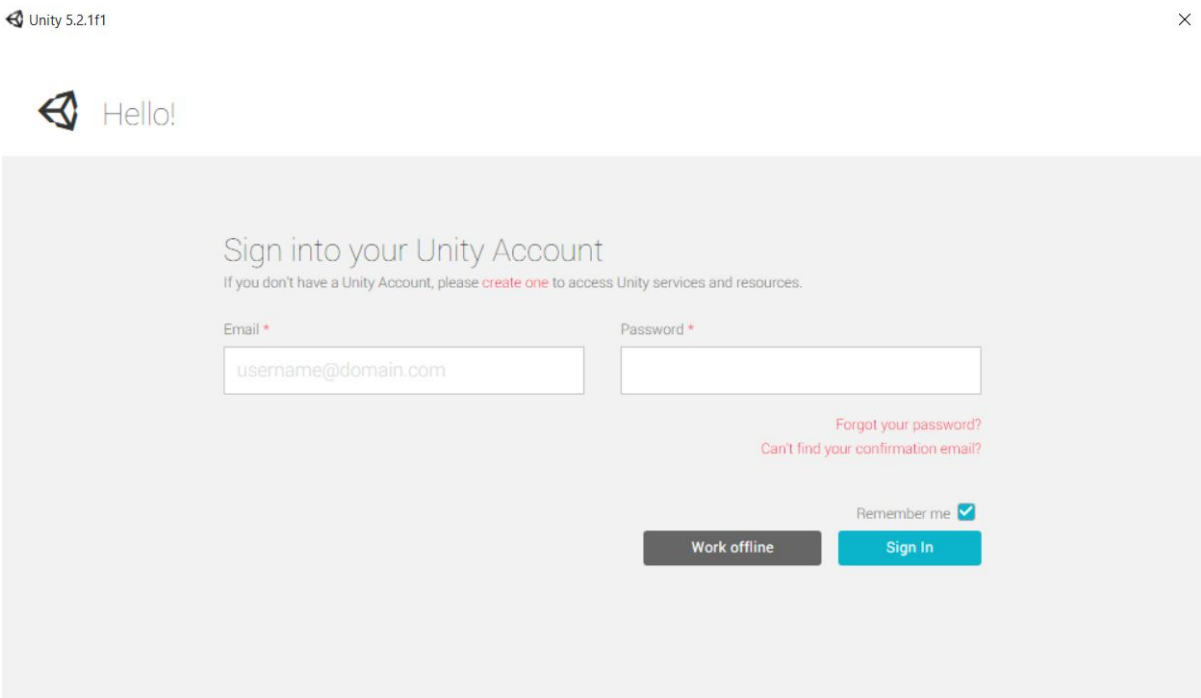
1. Vous recevrez un mail de confirmation sur votre boîte mail. Cliquez sur Confirm Email.



1. Une nouvelle page internet s’ouvre : votre compte a été créé !



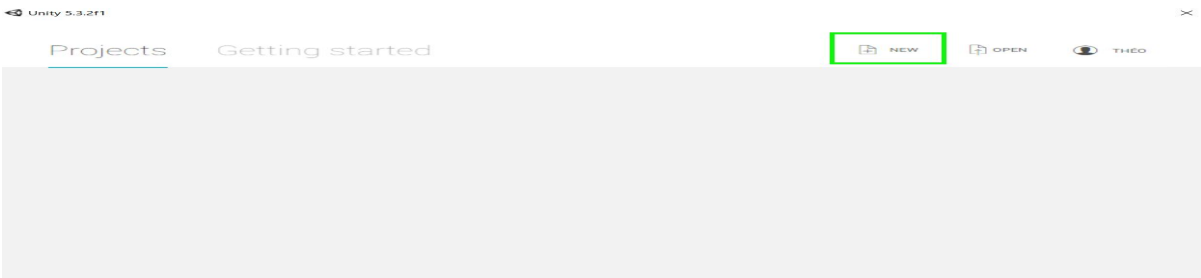
1. Retournez sur le logiciel Unity et entrez vos identifiants utilisateur : votre mail et mot de passe, puis cliquez sur Sign In.



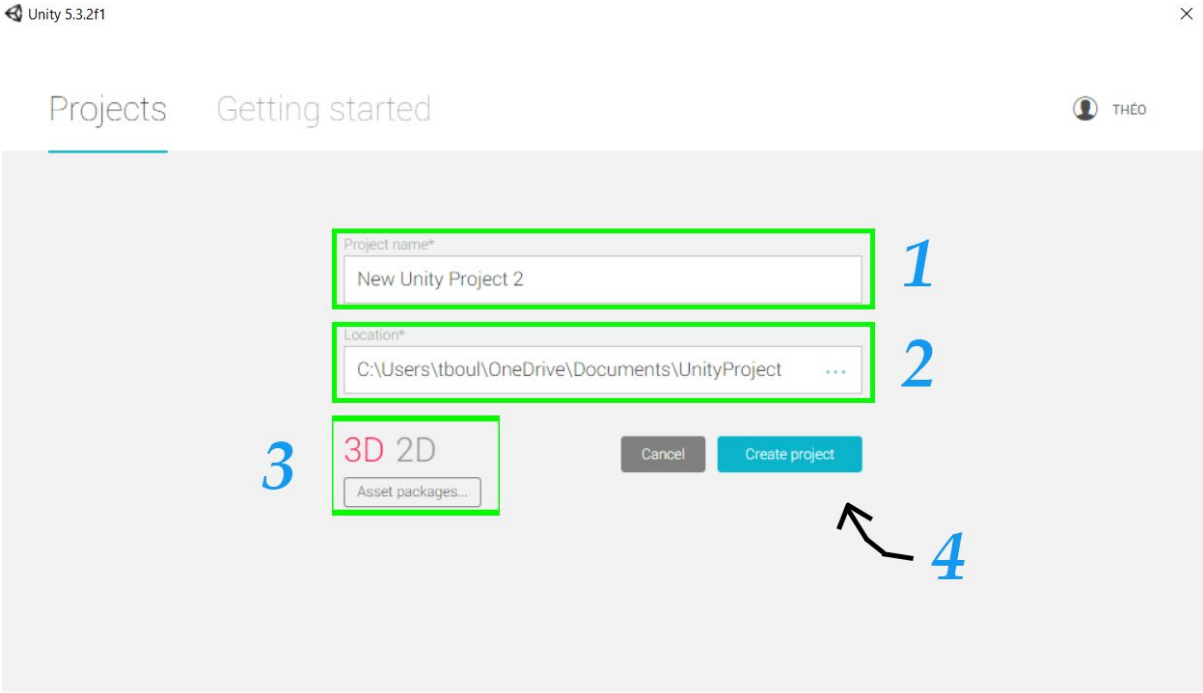
Prêt pour votre premier projet Unity ? :)

0.3 - Création d'un projet

1. Lancez Unity3D
2. Cliquez sur "New" comme ci-dessous:

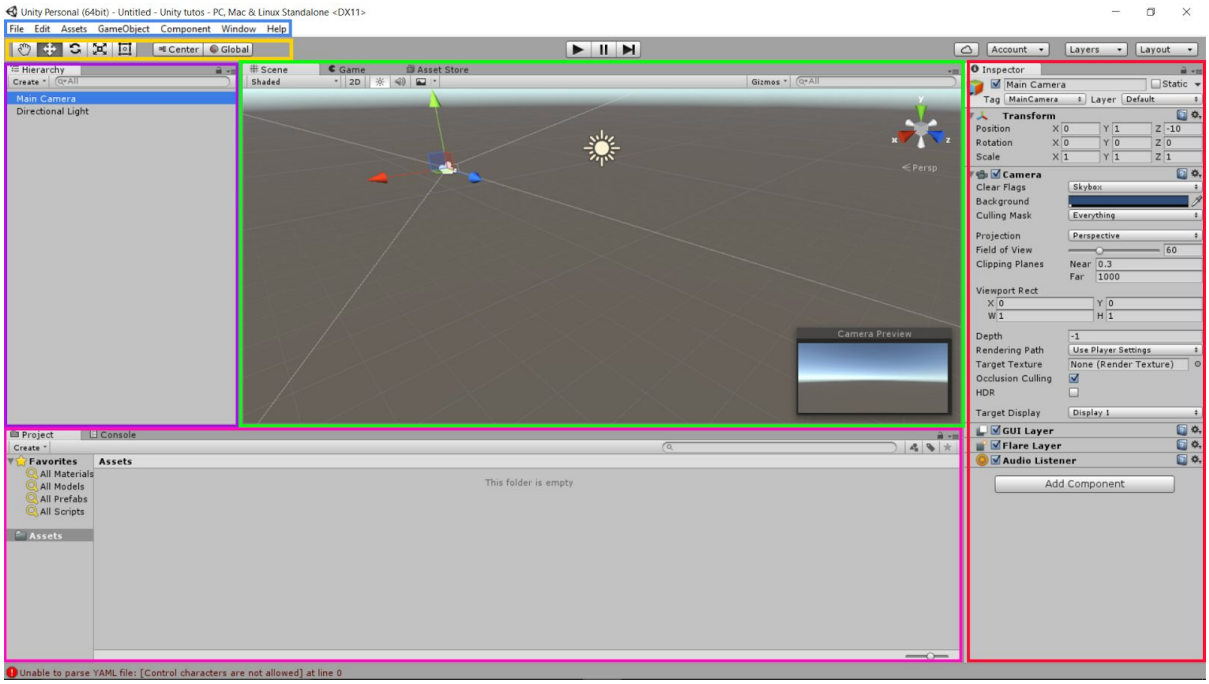


3) Suivez les étapes :



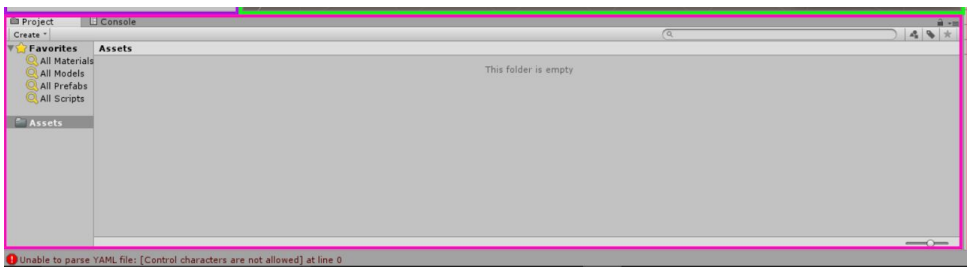
0.4 - La découpe de l'environnement de travail

1. Voici une vue d'ensemble de l'éditeur 3D d'Unity lors de la création d'un tout nouveau projet:



Cet ensemble peut être découpé en plusieurs blocs distincts ayant leurs spécificités propres :

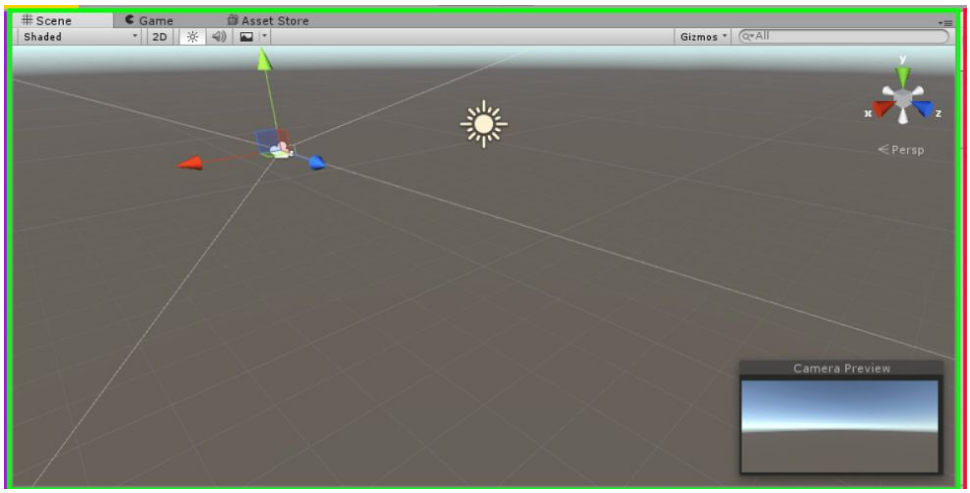
a. Le bloc projet :



Pour le moment il est vide, il se remplira tout au long de ce tutoriel. C'est à cet endroit que vous trouverez l'ensemble des fichiers enregistrés sur votre ordinateur correspondant à votre jeu.

Vous pouvez retrouver ces fichiers à l'endroit où vous avez choisi d'enregistrer votre projet.

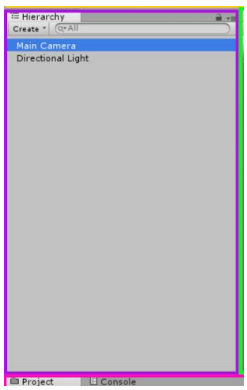
b) Le bloc scène :



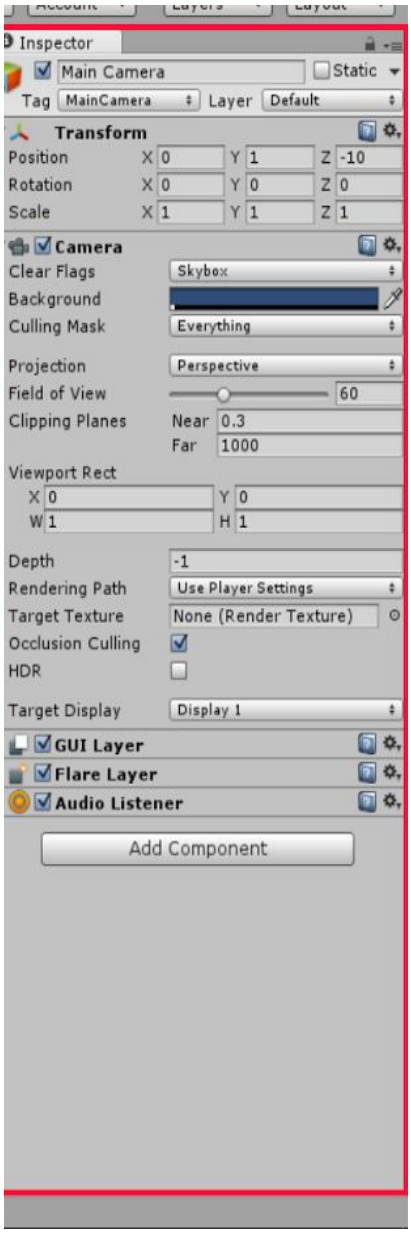
La scène correspond à votre plan de travail, c'est là que vous allez interagir avec les éléments de votre jeu. Par défaut il y a déjà deux objets placés dans votre scène : Une caméra et une lumière. Nous reviendrons plus en détail sur ces deux éléments dans la partie "Components" de ce tuto.
Comme vous pouvez le voir dans le coin en haut à droite de votre scène, Unity3D utilise un repère x, y, z avec l'axe y à la verticale.

c) Le bloc "Hierarchy"

Nous avons pu constater la présence de deux éléments ajoutés par défaut à notre scène, (la caméra et la lumière) que nous pouvons retrouver dans ce bloc. Ce bloc liste l'ensemble des élément présents sur votre scène pour s'y retrouver plus facilement par la suite.
Il est donc important d'avoir une hiérarchie bien organisée pour pouvoir retrouver un élément précis parmi les nombreux qui composeront votre scène (il n'est pas rare d'en avoir plus d'une centaine !).



d) Le bloc "Inspector" :



Si vous observez bien les deux images précédentes vous constaterez que dans la scène tout comme dans la hiérarchie ma caméra est sélectionnée. L'inspecteur d'éléments nous permet d'avoir plus d'information sur l'objet sélectionné.

Le Transform, c'est un élément commun à tout ce que vous pouvez poser dans votre scène, il donne des informations sur la position, la rotation et le "Scale" (les dimensions) de votre objet.

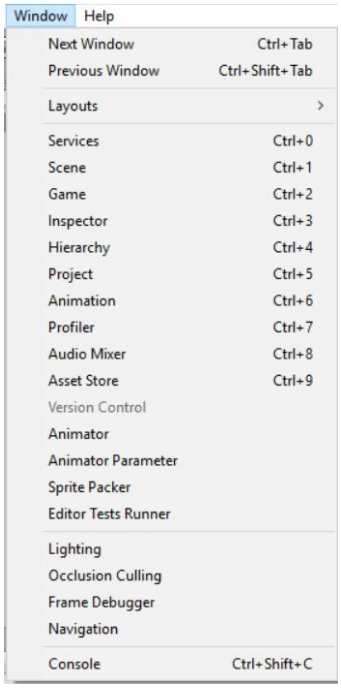
Vous avez des informations plus précises sur le type de caméra que vous utilisez ainsi que le type d'affichage.
Nous reviendrons plus en détails en temps et en heure sur les différents paramètres.

Ces trois éléments indiquent que la caméra peut :

- afficher les éléments de "l'interface utilisateur" ou UI. Plus d'information dans la documentation Unity (!en anglais)
- simuler des effets de lumière. Documentation unity à ce propos
- gérer du son. Documentation unity à ce propos

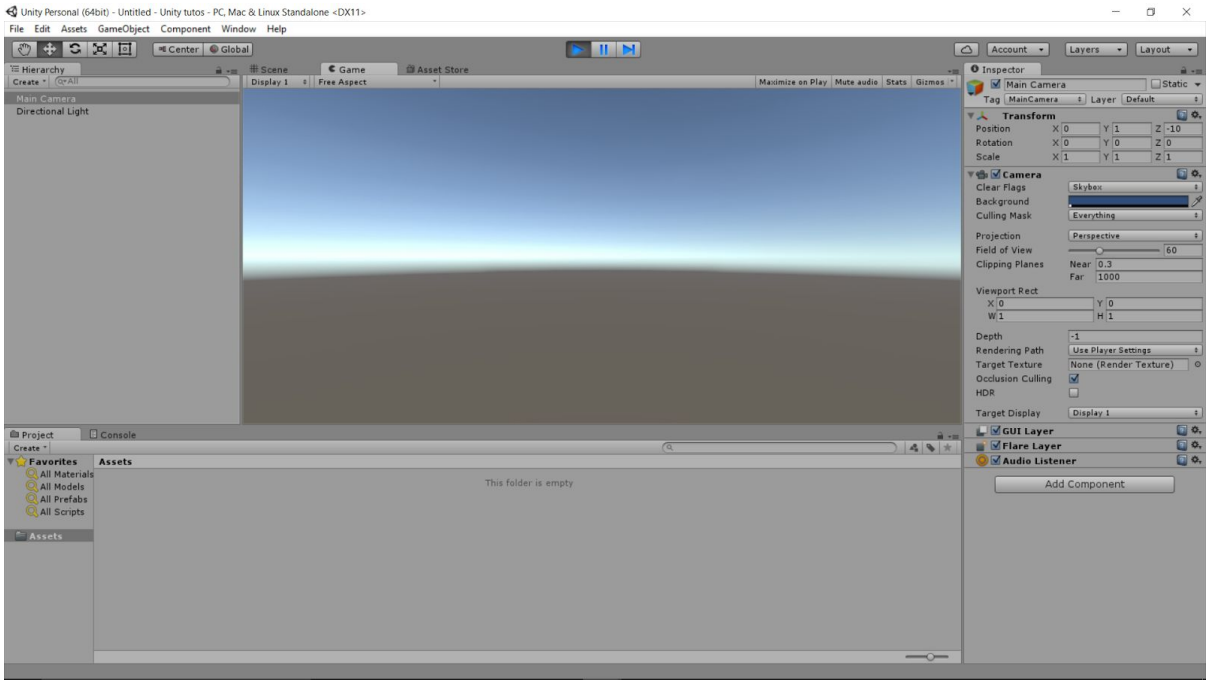
Nous reviendrons plus en détail sur le fonctionnement des "Components" car ils sont très nombreux.

Nous avons maintenant vu les éléments de base de l'environnement d'Unity3D. Il est très complet, et nous verrons au fur et à mesure les différentes options. Un petit aperçu des autres éléments (la plupart étant non affichés par défaut) :



Avant de quitter cette présentation de l'interface je vous invite à cliquer sur le bouton "Play" :

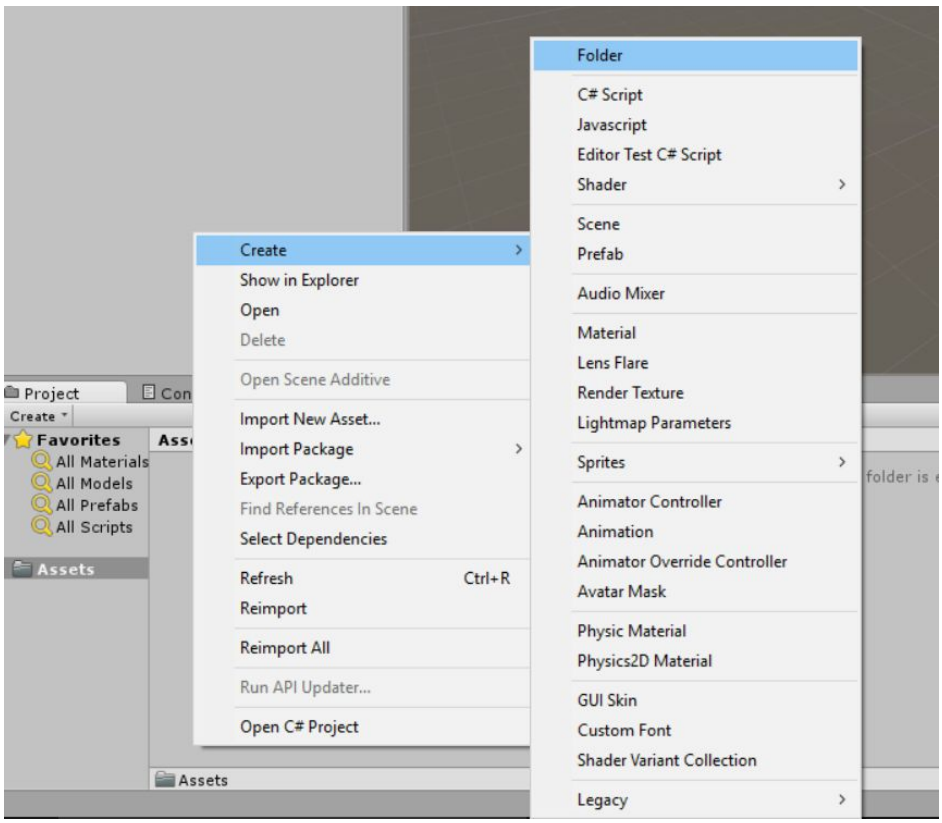
Vous devriez avoir un changement d'affichage au moment où vous appuyez sur le bouton :



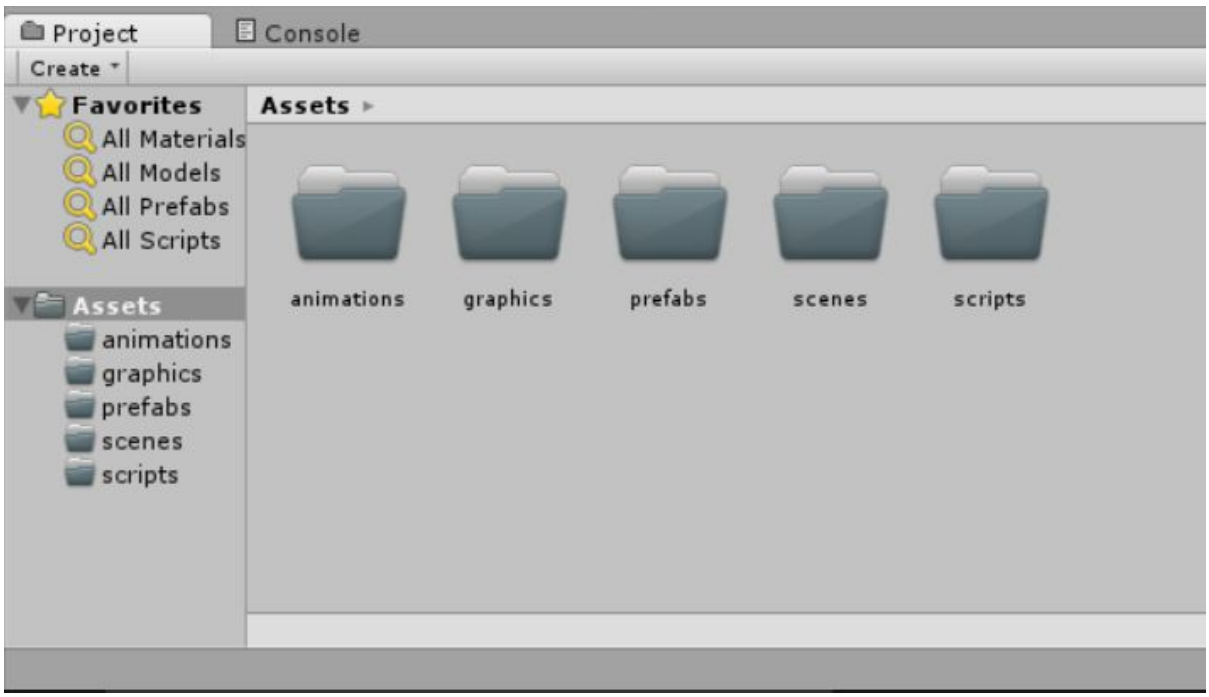
Le fond est plus sombre et nous ne voyons plus la scène mais le “Game”, c’est le rendu de votre jeu vu par votre caméra.
Vous pouvez toujours bouger les éléments dans votre scène a la main ou manipuler les paramètres dans l’inspecteur d’éléments mais aucun des changements ne sera sauvegardé lorsque vous quitterez le mode “Play”.

0.5 - Organisation simple des dossiers du projet

Nous allons rajouter quelques dossiers afin de faciliter l’organisation de notre projet. On va faire nos manipulations dans le bloc “projet” :



Nous verrons au fil des différents tutoriels à quoi servent tous ces dossiers. Créez les dossiers suivants :



1 - Plante le décor

Objectif

Pour construire un super jeu, il faut le faire par étapes. La première, c'est de créer les différents éléments que tu veux mettre à l'intérieur de ton jeu !!
Dans ce tutoriel, on va se concentrer sur le décor. A la fin du tutoriel, on aura un super environnement pour y accueillir tous nos personnages :)

Ce que tu vas apprendre

- Créer un terrain
- Ajouter une Skybox
- Découvrir les game objects
- Importer des graphismes de l'asset store

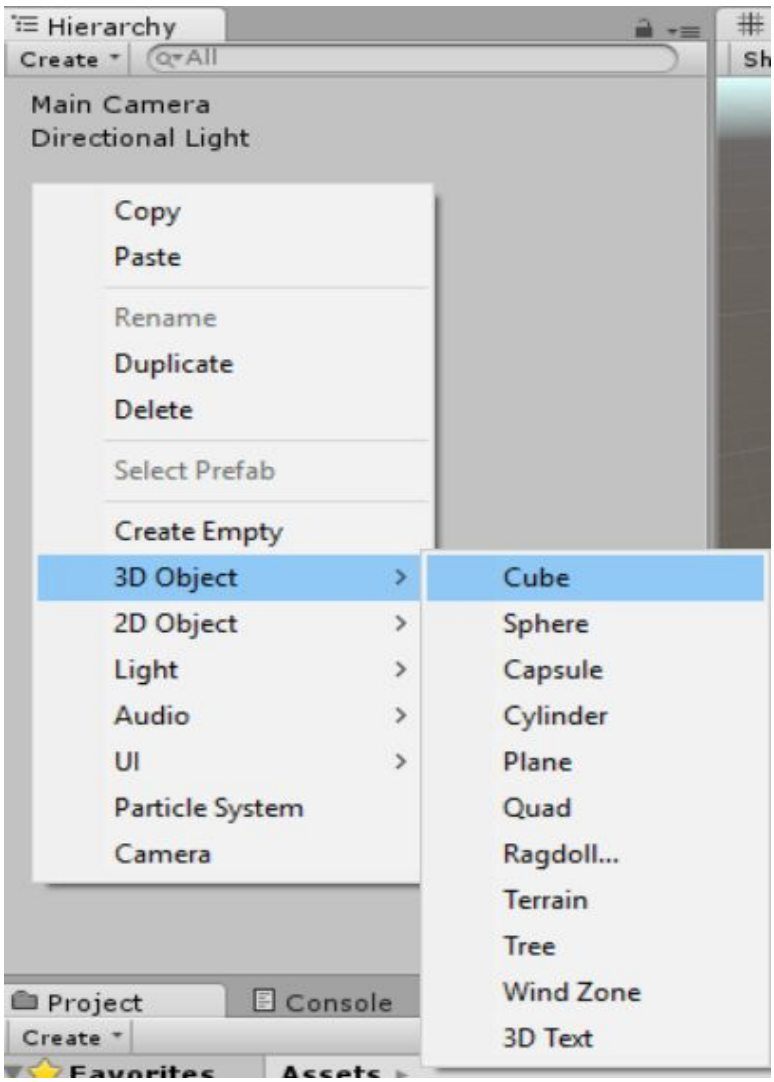
Méthodologie

Pour créer un beau décor, tu vas commencer par modéliser ton terrain.
Tu vas découvrir ce que sont les GameObjects et les Préfabs dans [Tuto GameObject et Prefabs].
Si tu désires utiliser des ressources produites par des designers, va jeter un coup d'oeil au [Tuto Asset Store].
Ensuite tu vas apprendre à donner de la matière à tes GameObject à travers des [Tuto Textures et Materials].
Tu auras ainsi tous les éléments pour créer ton terrain en suivant le tutoriel [Tuto Terrains].
Enfin tu vas vouloir entourer ton monde avec une [Tuto Skybox].

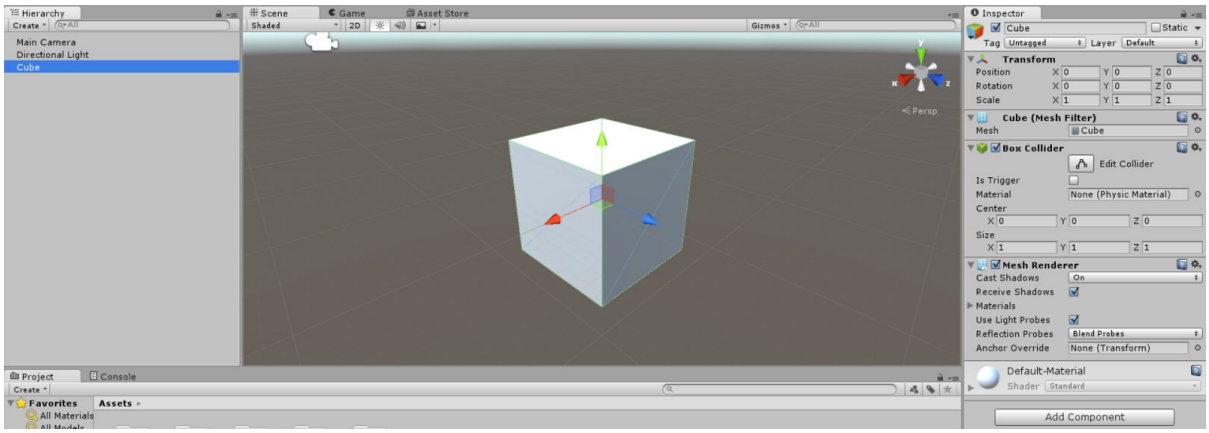
1.1 - Les gameObjects et les Prefabs

Unity3D n'est pas un logiciel de modélisation 3D ni de dessin, mais il existe quelques objets simples que l'on peut créer depuis l'éditeur. Pour les formes plus complexes il faudra passer par des logiciels plus spécifiques tel que Blender, Maya, ou 3Ds max.

Voyons quels éléments sont à notre disposition.
En faisant un clic droit dans la hiérarchie vous trouverez ceci :



- Cliquez sur "Cube" et observez ce qui ce passe sur votre écran.
  Dans la hiérarchie une nouvelle ligne qui correspond à votre cube s'est rajoutée ; en double-cliquant dessus la scène va se focaliser sur votre cube.

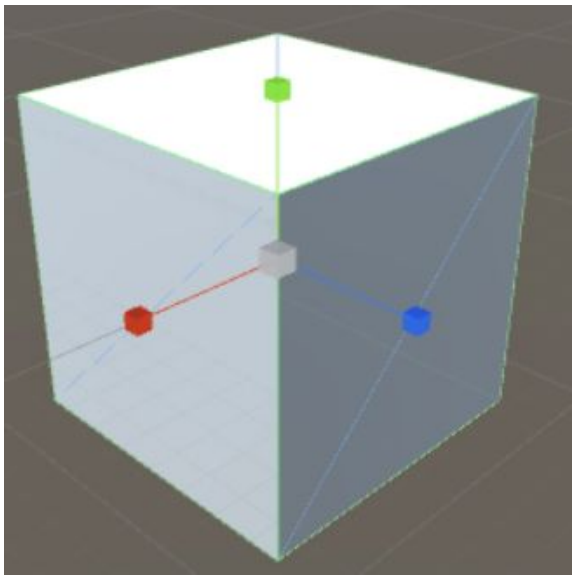


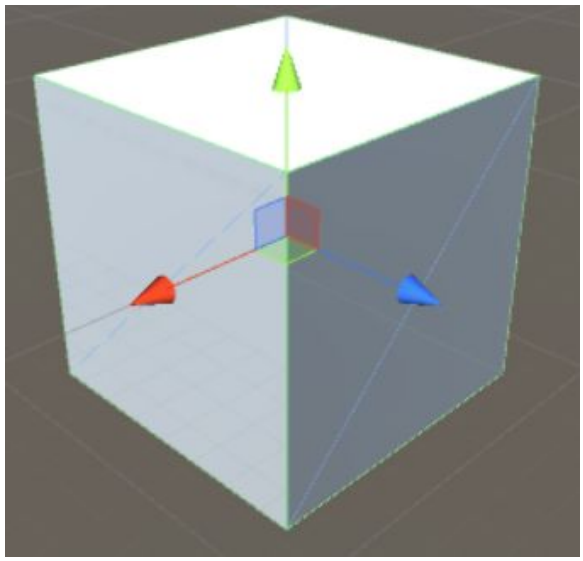
Vous avez désormais ceci :

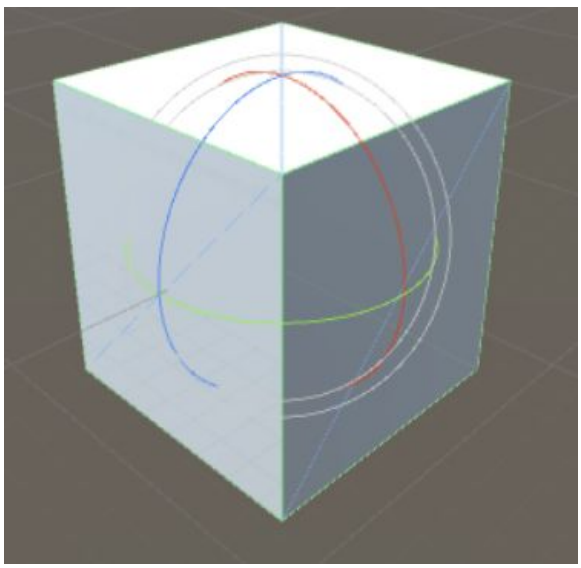
- Essayez de faire bouger votre cube depuis votre scène en tirant sur les flèches de couleurs qu'on appelle flèches gizmo.
- Maintenant essayez d'entrer directement des valeurs dans la partie position de l'inspecteur.
- Recentrez la scène sur le cube en double-cliquant dessus dans la hiérarchie.
- Modifiez la rotation et le "scale" dans "Transform" pour comprendre comment agir depuis l'inspecteur sur un objet.

Petite astuce : en haut à gauche, au dessus de la hiérarchie vous trouverez ces boutons :

Ces trois options changent les gizmos pour éviter d'avoir à toucher les valeurs de l'inspecteur. Vous pouvez maintenant déplacer, tourner, et redimensionner ("scale" en anglais) vos objets à la souris :

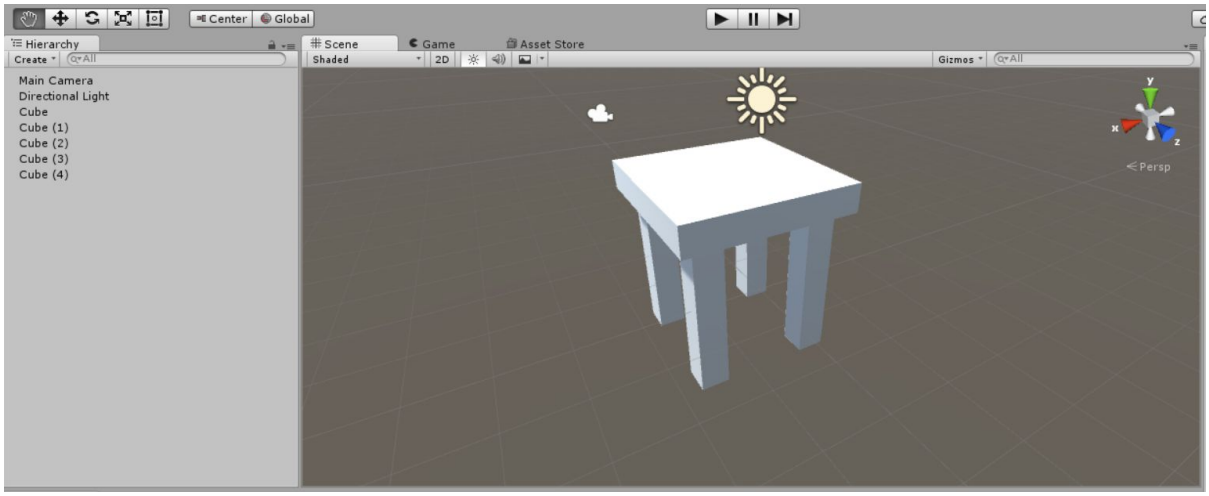






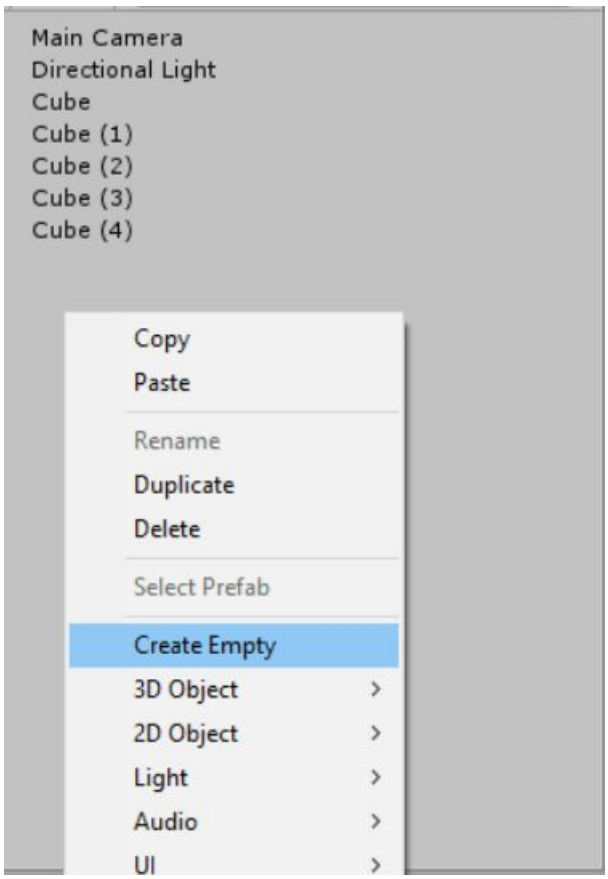
- Nous allons maintenant créer un tabouret. Pour celà, nous aurons besoin de 4 cubes supplémentaires.

  Bougez les valeurs de “scale” et de position des cubes jusqu'à avoir un résultat similaire :

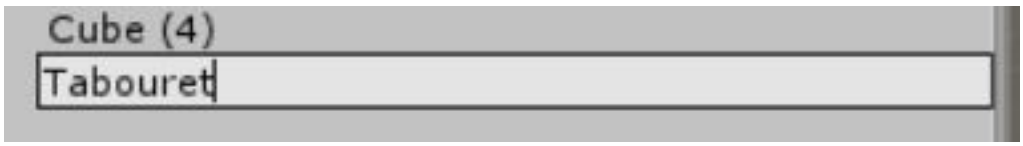


Nous allons maintenant réorganiser notre hiérarchie afin d'indiquer à Unity que le tabouret est un seul objet.

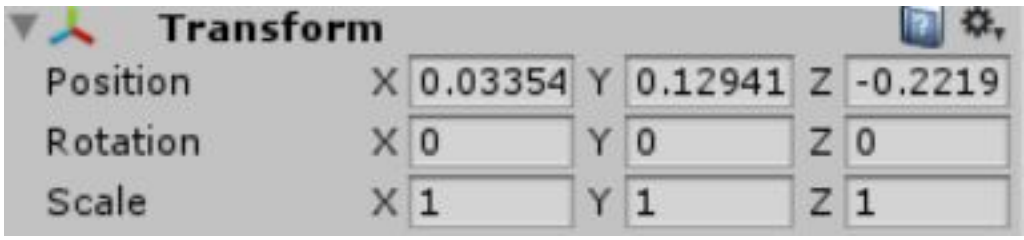
Pour cela, nous allons créer un "gameObject" vide. C'est un objet invisible sans consistance ; il n'a donc qu'un "transform" dans son inspecteur d'élément. Il est cependant bien pratique car on peut grouper des objets et les attacher à notre "gameObject" : on pourra donc bouger notre tabouret en bougeant uniquement le GameObject.



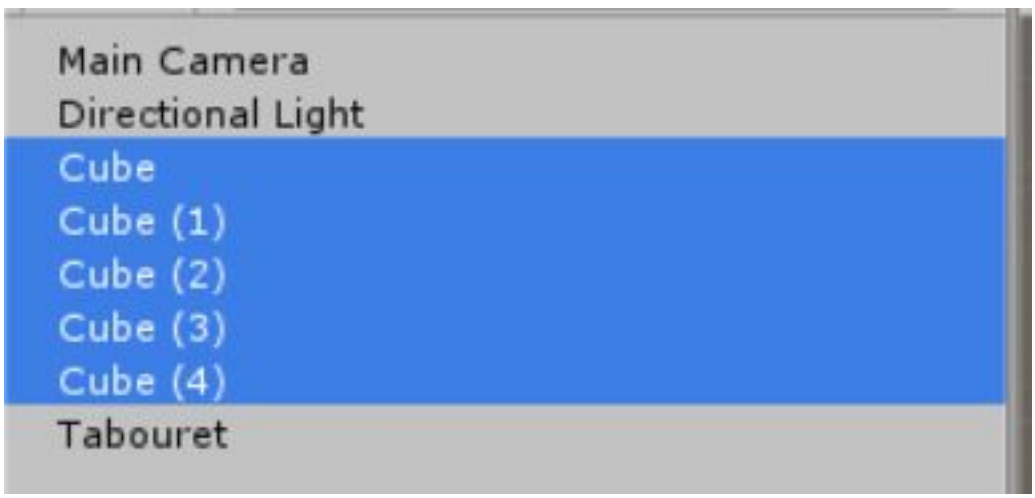
1. Créer un “gameObject” vide. (clic droit dans la hiérarchie)



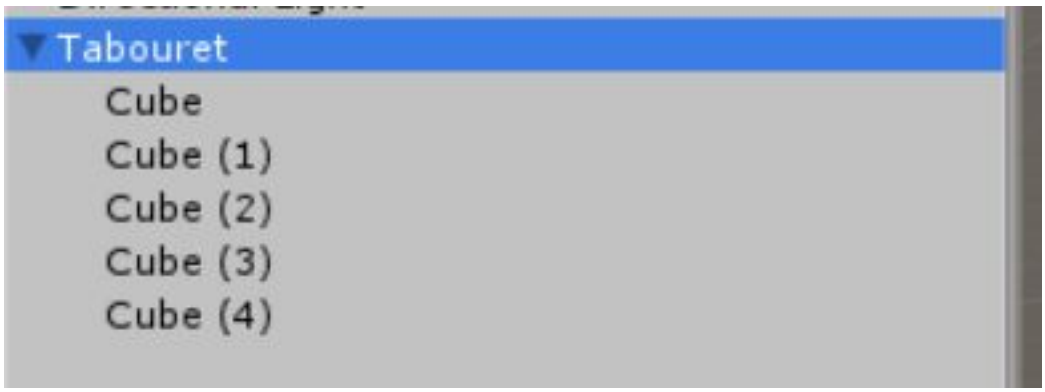
2) Renommer en tabouret



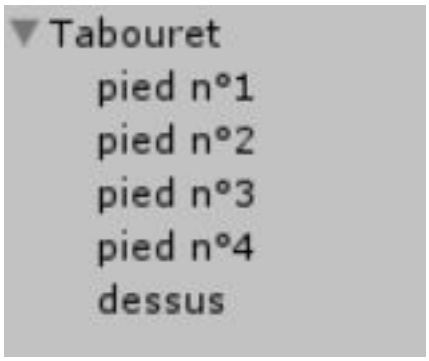
3) En haut à droite du transform cliquez sur le petit rouage et sélectionner “reset” ce qui va positionner votre Tabouret en 0,0,0 et mettre la rotation à 0,0,0 puis le “scale” à 1,1,1.



4) Sélectionner tous les cubes (clique sur le premier puis en maintenant la touche MAJ enfoncée sur le dernier).



5) Faire glisser tous les éléments dans le tabouret (notre “gameObject” vide).



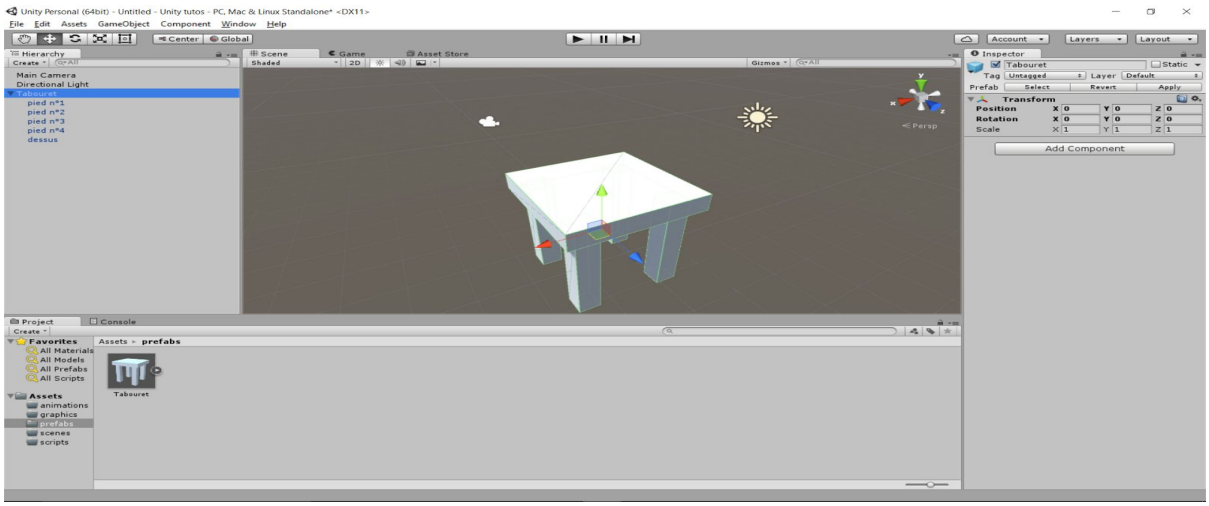
6) Renommer les cubes (ça permet de s’y retrouver plus facilement).

Maintenant sélectionnez le tabouret et déplacez le avec les gizmos ou l’inspecteur : tous les éléments bougent ensemble !
Un peu de vocabulaire : l’objet Tabouret est désormais “Parent” des “pieds” et du “dessus”.
On dit d'ailleurs que ceux-ci sont des “Enfants” de Tabouret.

!nous allons créer un Prefab.

Qu'est ce que c'est et à quoi ça sert ?
Et bien il s'agit d'un objet modèle qui va être enregistré dans les fichiers du projet. On va pouvoir en faire des copies (appelées "instances") de notre objet, par exemple si on veut rajouter plusieurs tabourets dans notre pièce.

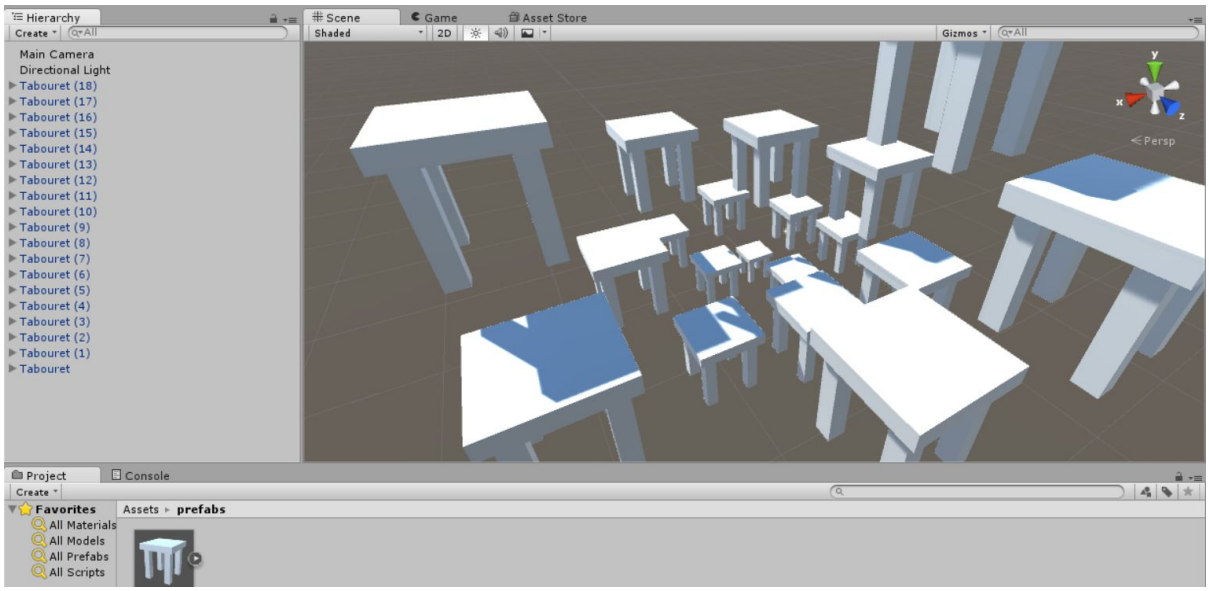
Comment créer un Prefab ?
Il suffit de faire un glisser-déposer depuis la hiérarchie de notre objet "tabouret" vers le dossier "Prefabs" que nous avons créé dans le tutoriel précédent.



Votre tabouret est désormais d'une couleur bleue dans la hiérarchie ainsi que tous ses éléments.

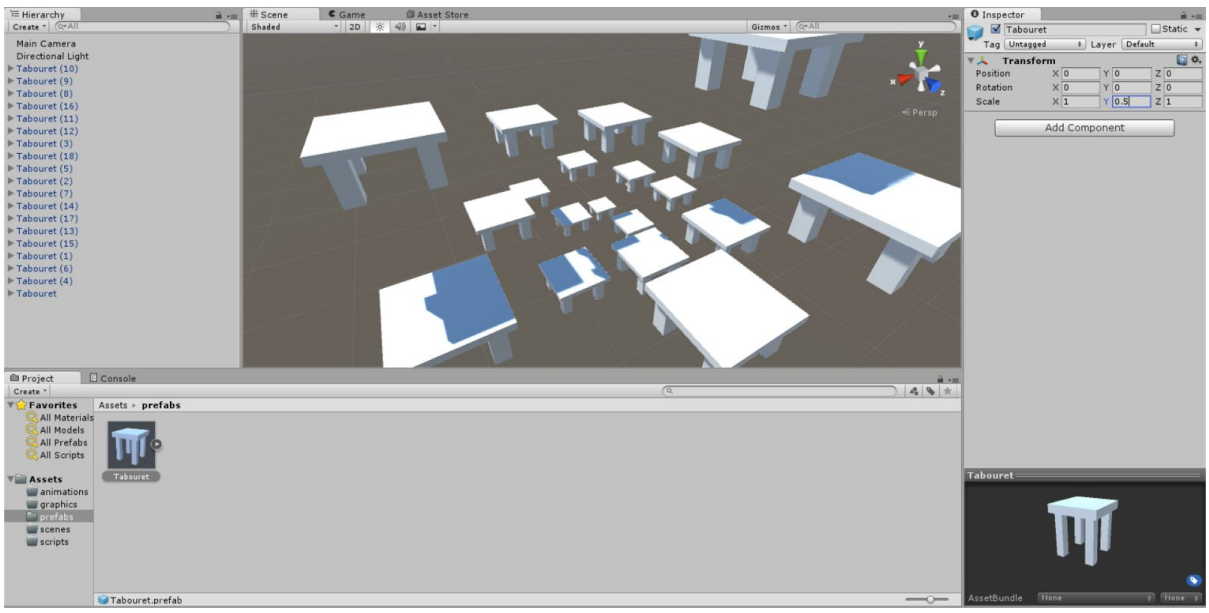
Astuce: Pensez à sauvegarder votre projet (Fichier > Sauvegarder la scène ou Ctrl+S) régulièrement pour éviter de perdre vos modifications en cas de problème ! (sélectionnez le dossier "scene" pour enregistrer votre progression).

Par un glisser-déposer ajoutez des tabourets dans votre scène depuis votre "Prefab" qui se trouve dans votre dossier de projet.



Important : Les tabourets ici font tous la même taille mais certains sont plus éloignés donnant l'impression qu'ils sont plus petits. Vous pouvez utiliser l'inspecteur pour modifier individuellement les instances si vous désirez qu'ils aient une taille différente.

Vous pouvez aussi modifier directement le "Prefab" pour modifier toutes les instances du tabouret :



Si tu veux plus d'informations sur les objets 3D de base, tu peux aller sur le manuel (attention, il est en anglais !).

Si notre projet contient trop d’objets, on peut réorganiser la hiérarchie en créant un “gameObject” “Tous les tabourets” qui contient tous les tabourets. On le crée et l’utilise comme on a fait avec le tabouret et les 5 cubes.

Microsoft & MAGIC MAKERS

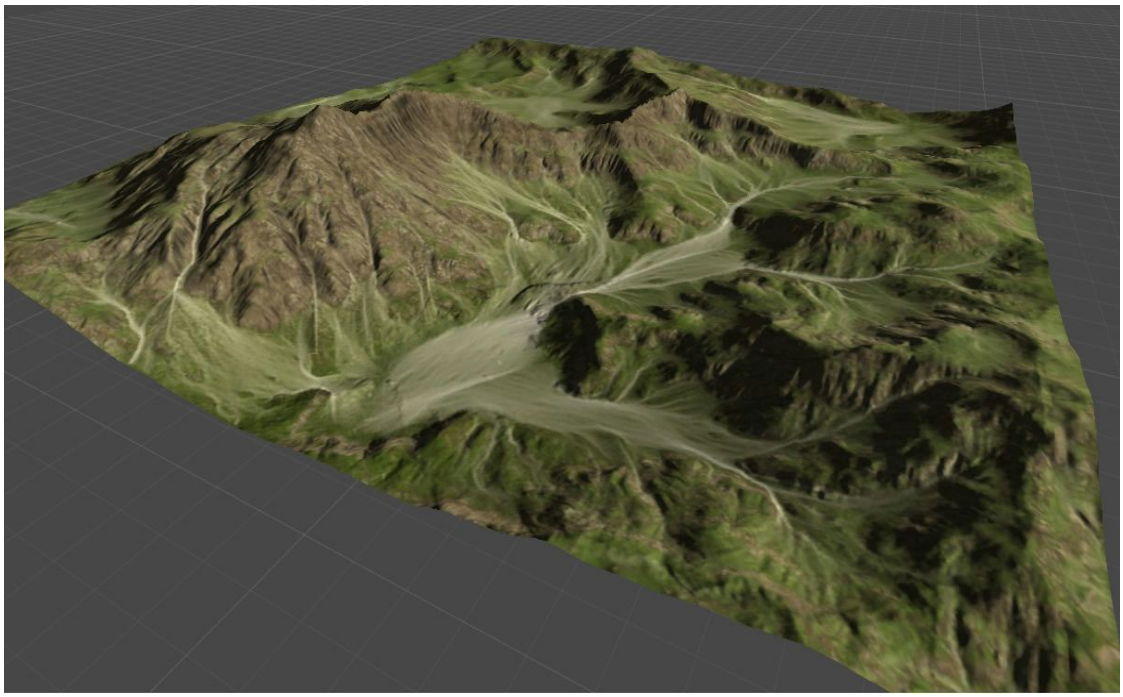
présentent

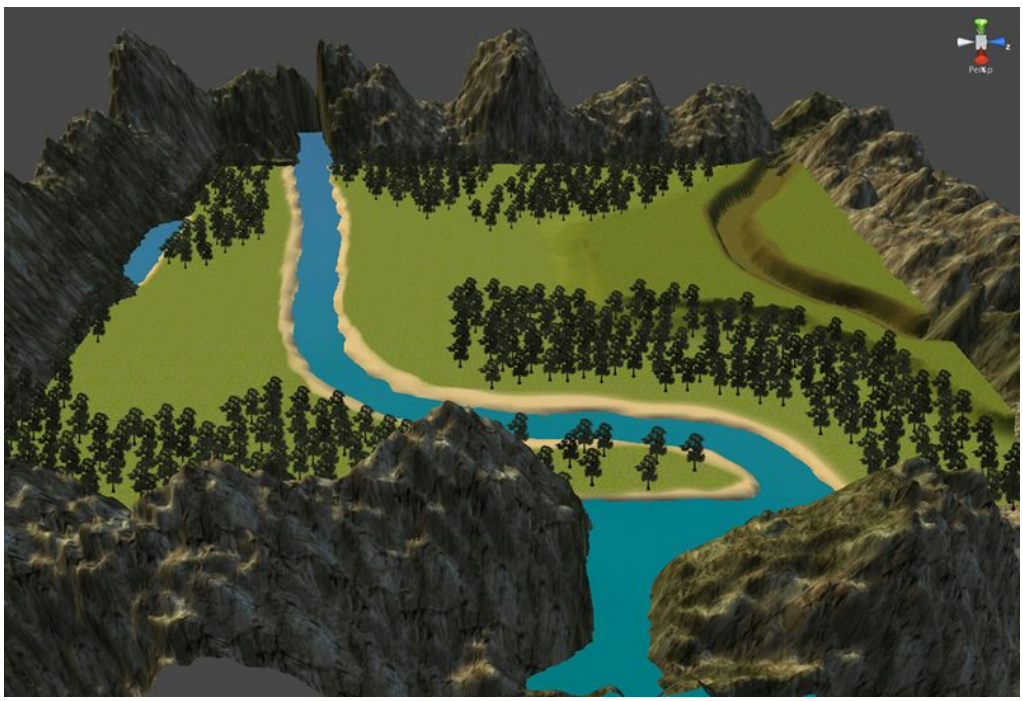
ImagineCup Junior

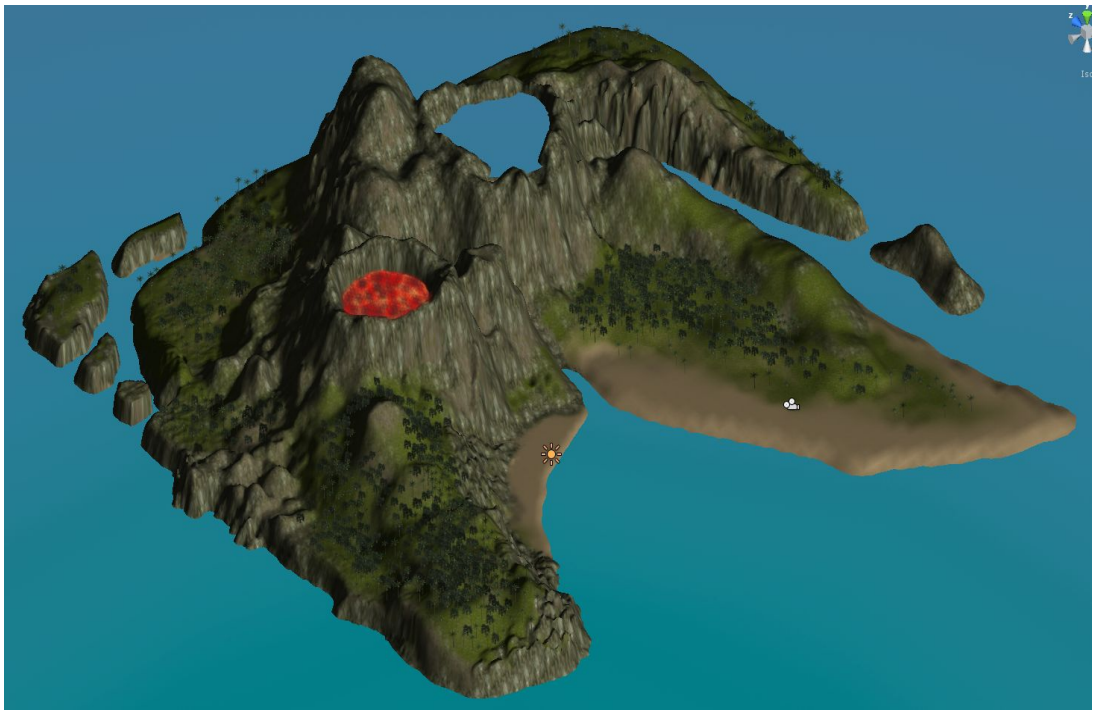
1.2 - les terrains.

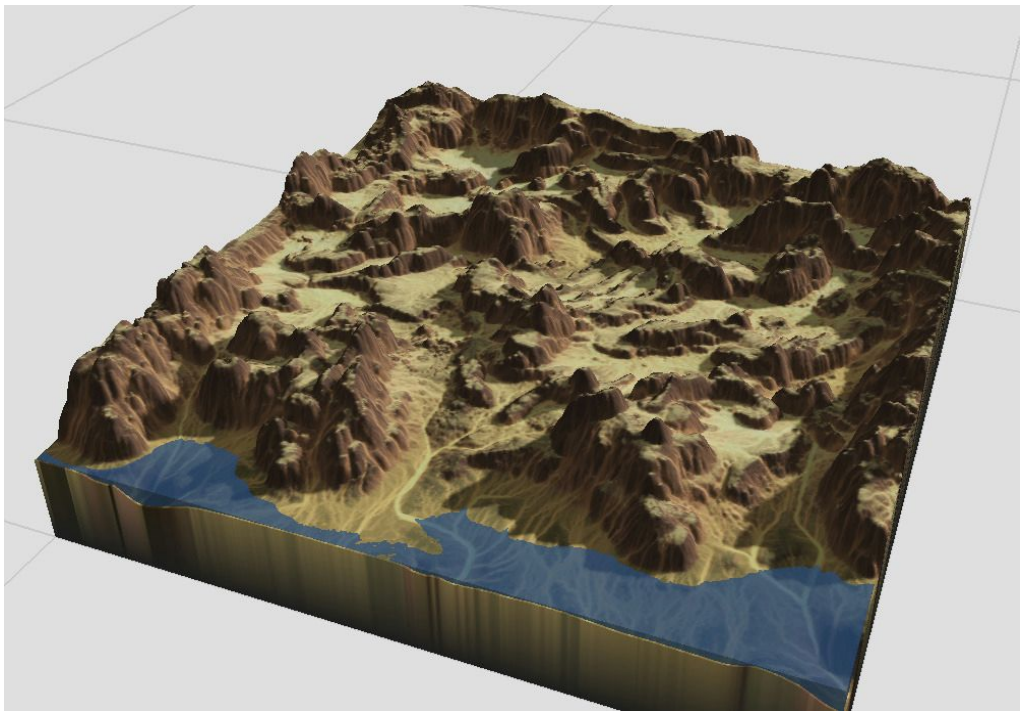
- Pré-Requis: Tutoriel sur les Textures & Materials

Qu’est ce qu’un terrain ?









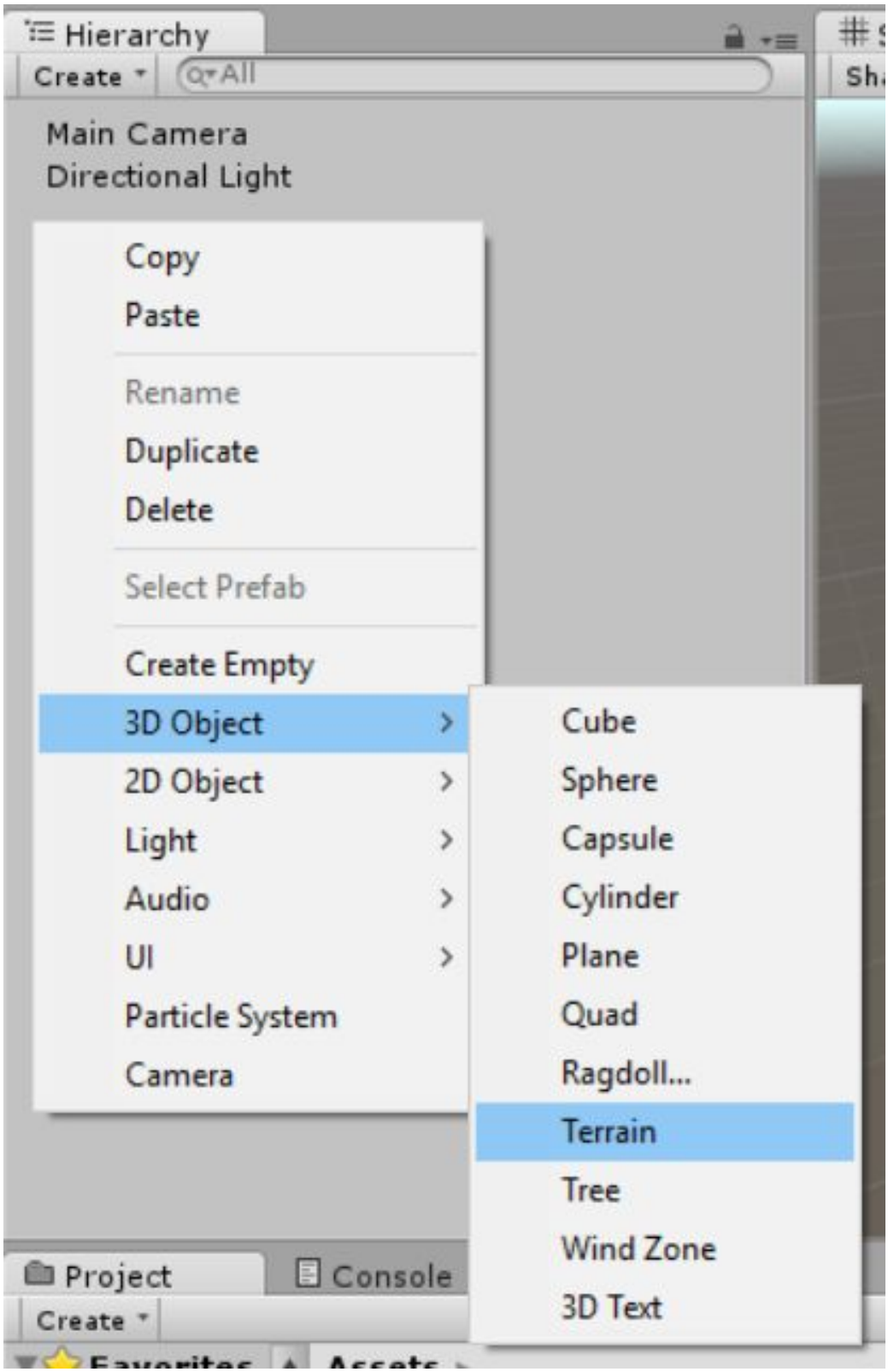
Un terrain est tout simplement une surface que vous allez pouvoir modeler et texturer à votre guise pour obtenir votre univers 3D.

On peut faire énormément de choses avec l'outil, mais il faut respecter certaines restrictions :

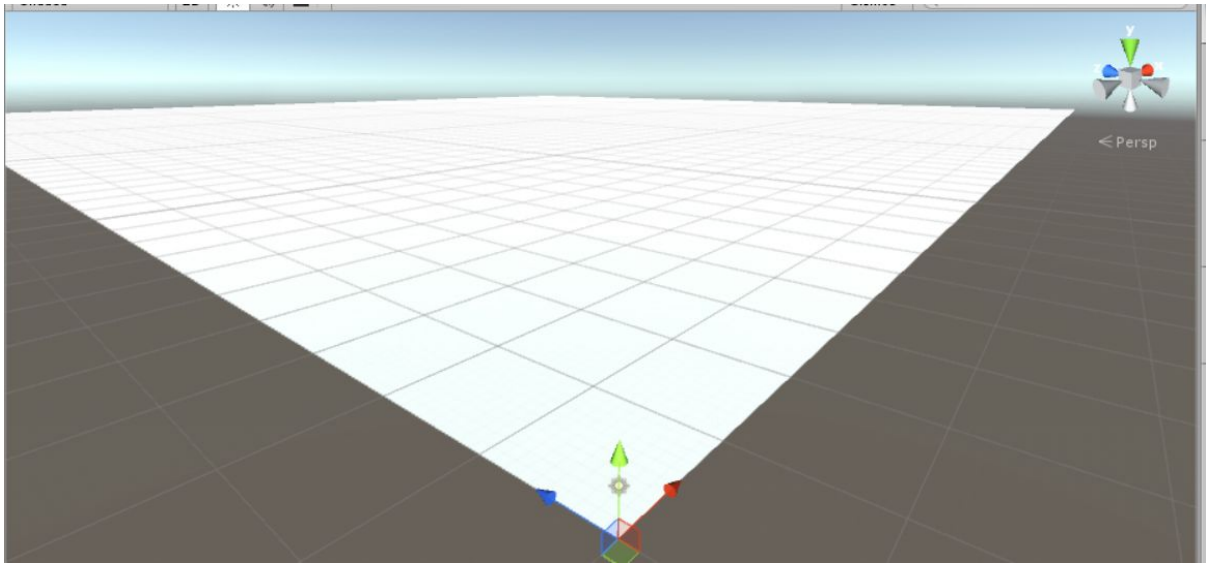
- Un terrain est une surface plate que l'on étire => pas de grottes, d'arche ou de ponts. (on doit rajouter des modèles 3D pour ça).

- On ne peut pas creuser en dehors de notre terrain ; si on veut un trou, un cratère ou une faille il faut commencer l'édition du terrain par surélever l’ensemble.
- La taille et l'échelle du terrain doivent rester réalistes : posez un cube sur le terrain et comparez, sachant qu’un cube fait 1 mètre.

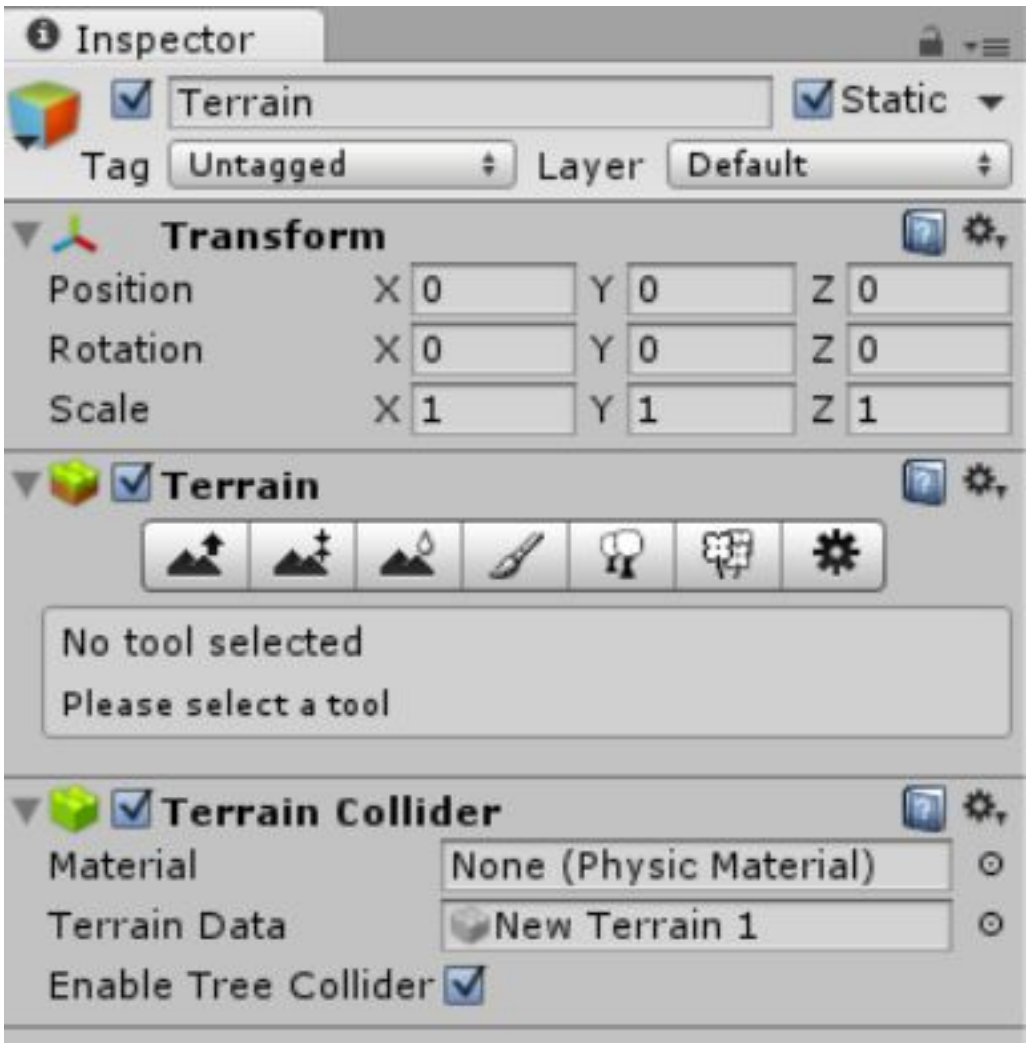
  Comment crée-t-on un terrain ?
  Faites un clic droit dans la hiérarchie , sélectionnez 3D Object puis Terrain.


Hierarchy
Create
All
Main Camera
Directional Light
Copy
Paste
Rename
Duplicate
Delete
Select Prefab
Create Empty
3D Object
2D Object
Light
Audio
UI
Particle System
Camera
Cube
Sphere
Capsule
Cylinder
Plane
Quad
Ragdoll...
Terrain
Tree
Wind Zone
3D Text
Project
Console
Create

Par défaut un terrain est une très grande surface blanche, posée sur la “couche” 0.



Regardons dans l’inspecteur les différents paramètres de notre terrain.



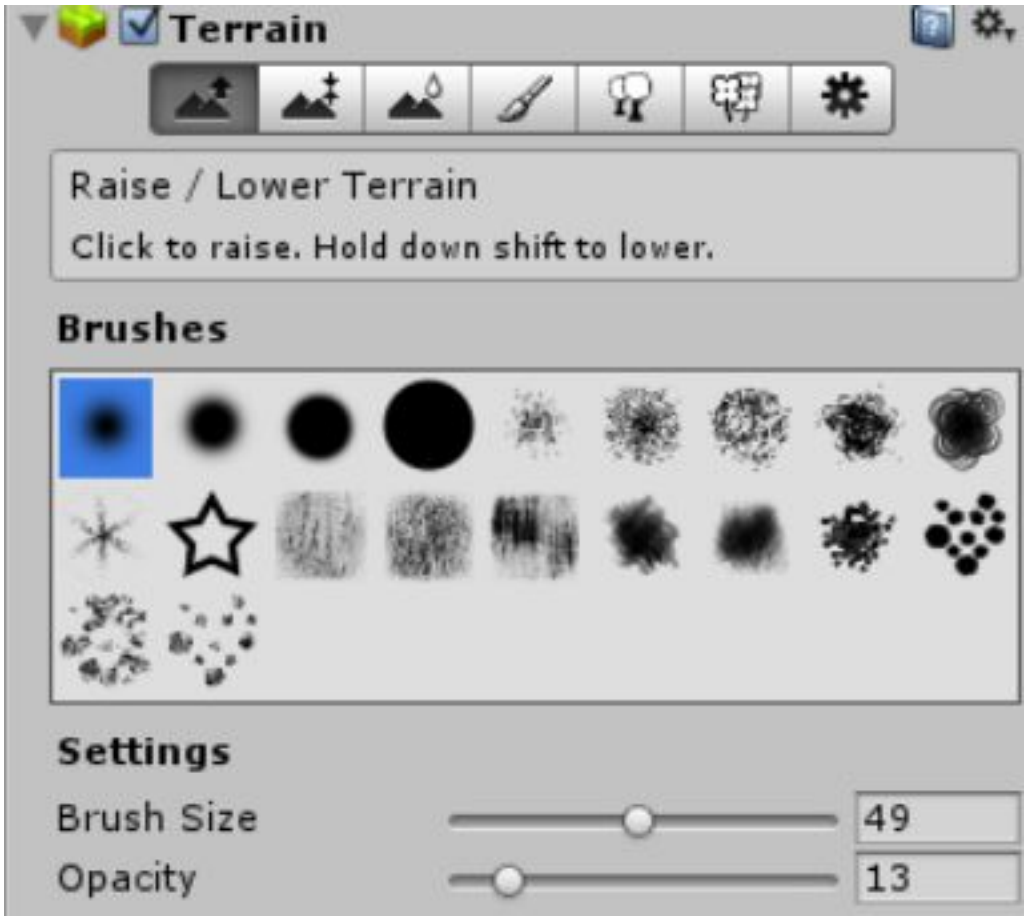
Un Transform comme pour tout objet d'unity.

Un composant Terrain avec les outils d'édition.

Un Terrain Collider pour gérer les collisions avec les autres objets.

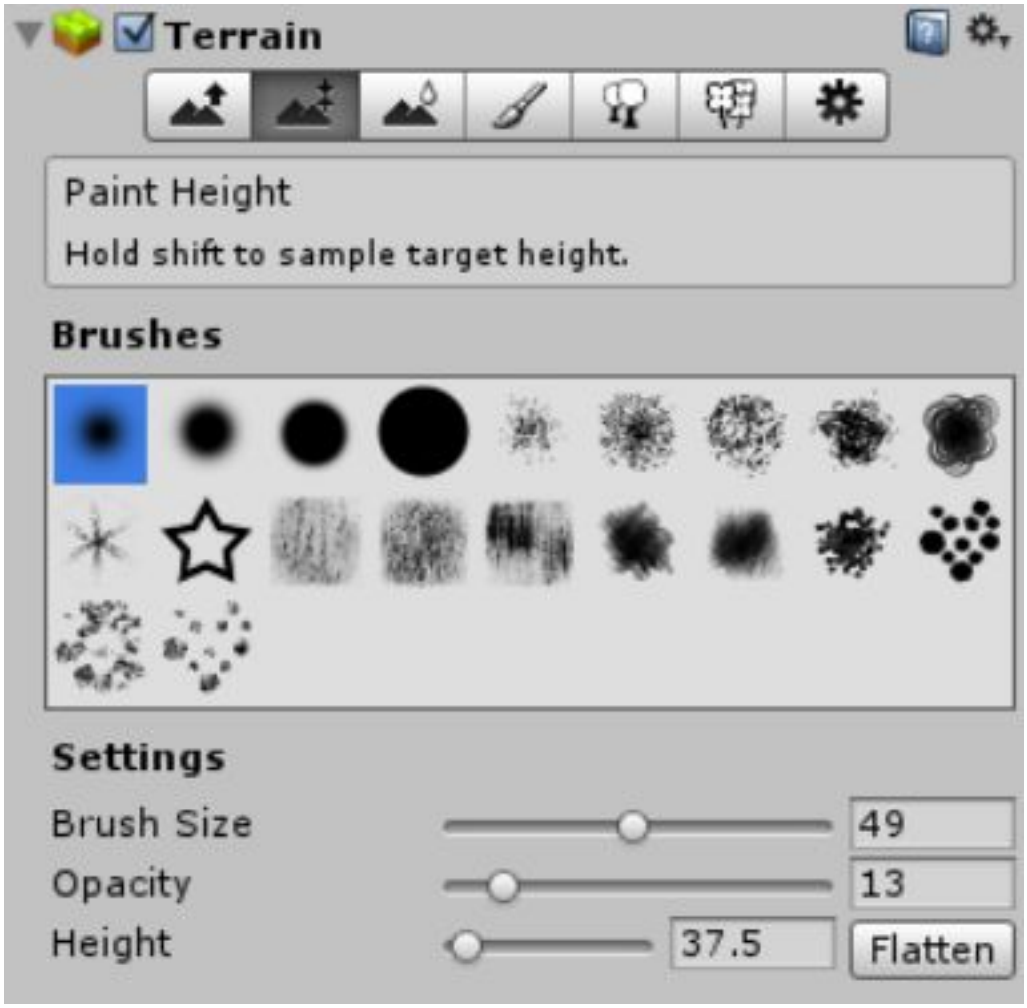
Regardons plus en détail les outils d'édition de terrain :

1. Augmenter / diminuer la hauteur du terrain



Différents pinceaux dont on peut modifier l’opacité et la taille.
Simple clic pour monter le terrain, MAJ (“shift” en anglais) +Clic pour le baisser.

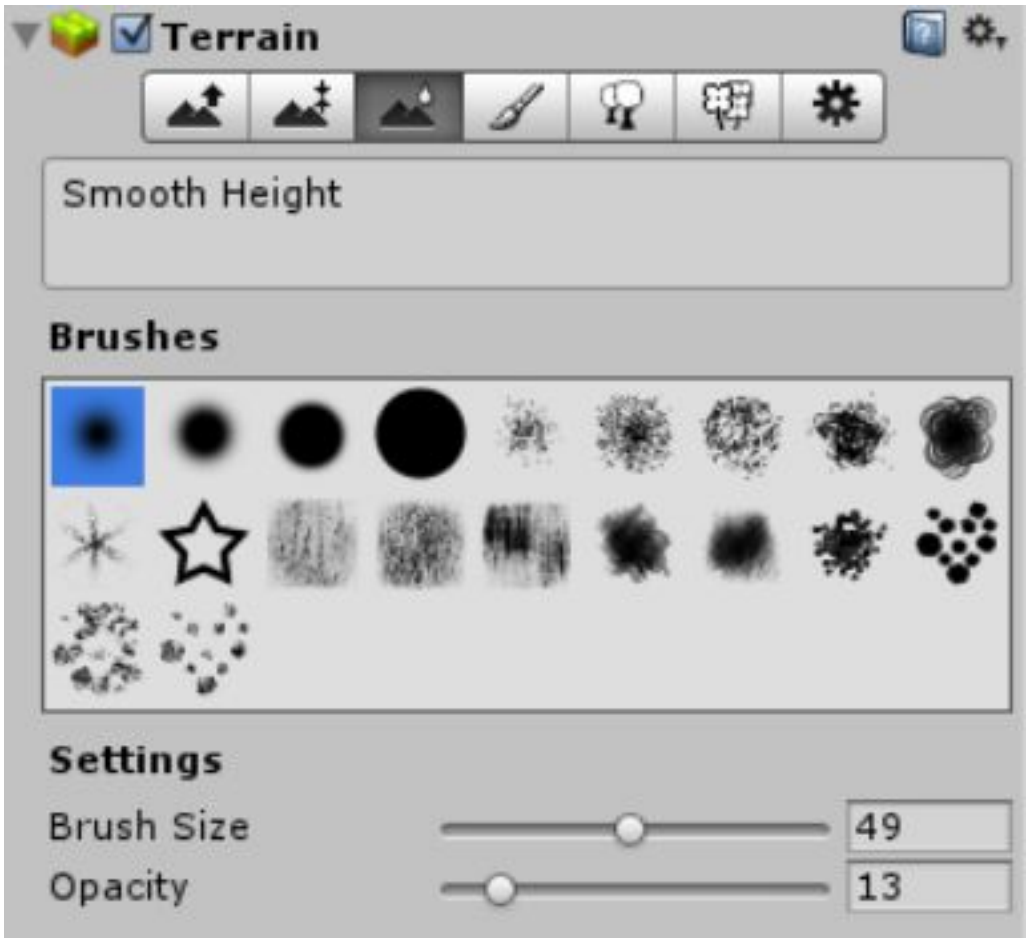
2. L'égaliseur :



Dans le mode “Flatten”, vous pouvez mettre l’ensemble de votre terrain au niveau voulu (le paramètre “height”). Ici, il vaut 37.5 donc les zones modifiées auront une hauteur de 37.5

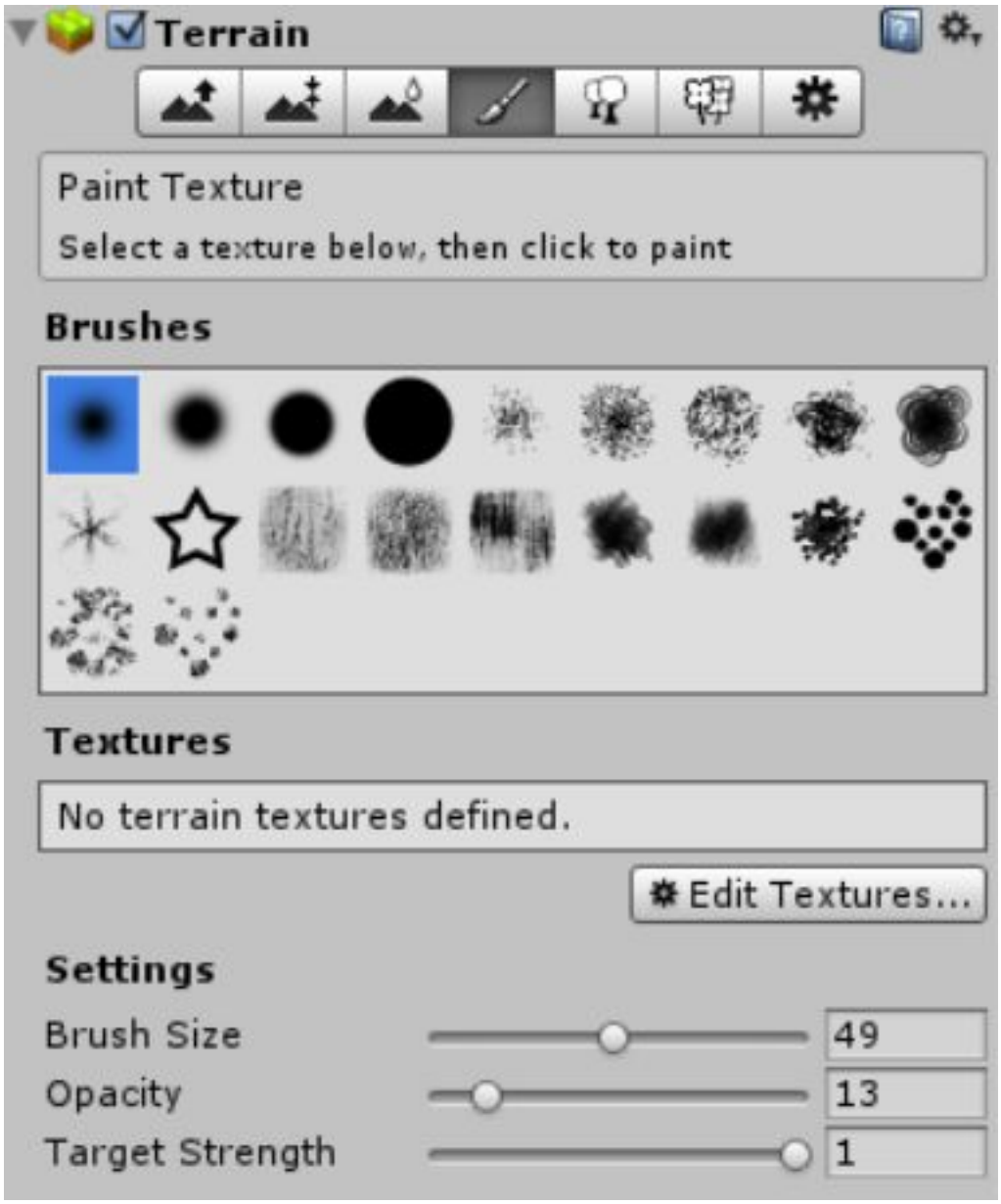
Si vous prévoyez de faire des trous, failles, cratères, etc le mode “flatten” est très pratique pour réhausser votre terrain, avant d’y creuser.

3. L'adoucisseur d’angles



Permet d’arrondir les angles dans la zone. Utile si vous avez des objets très rectangulaires (cubes, rectangle …) dans votre projet.

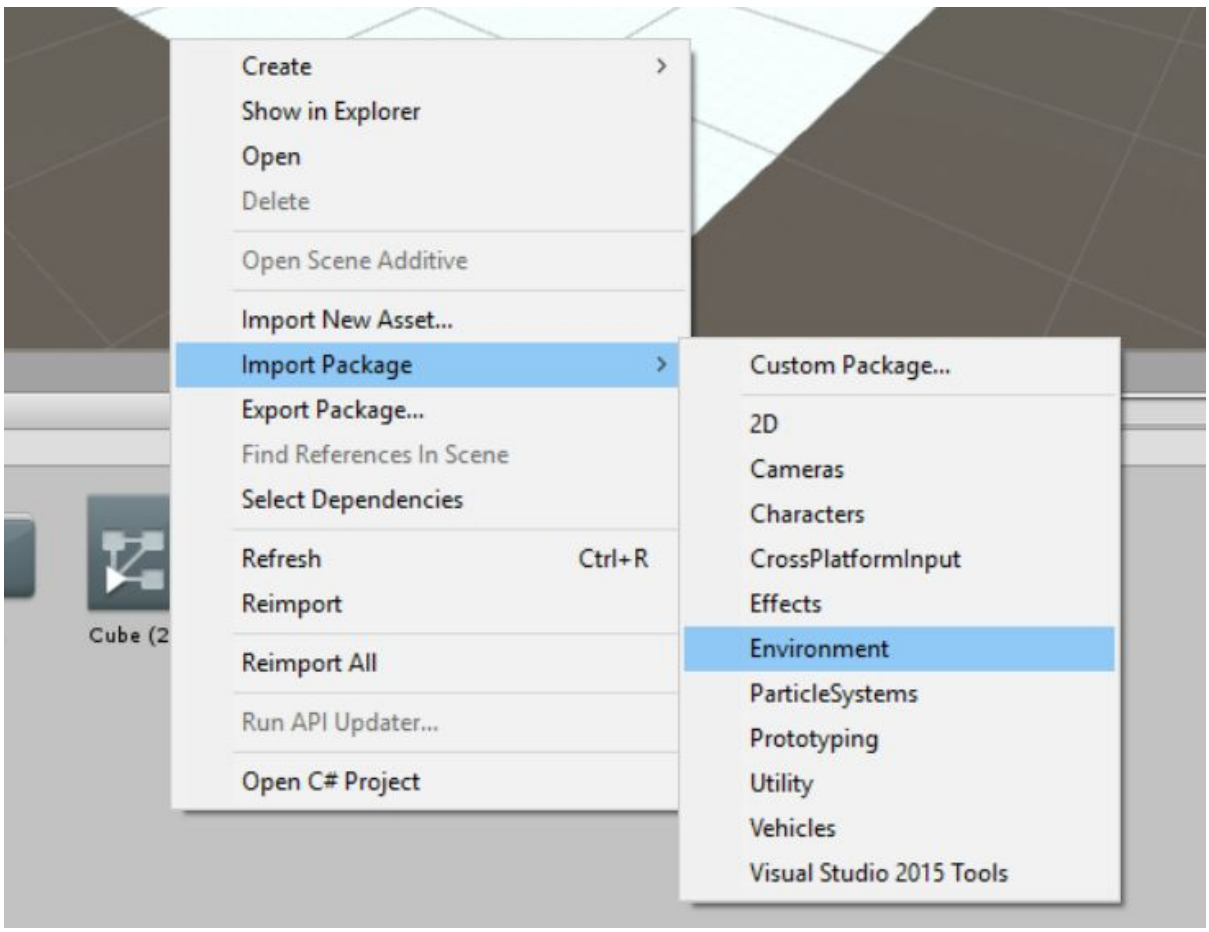
4. Les textures



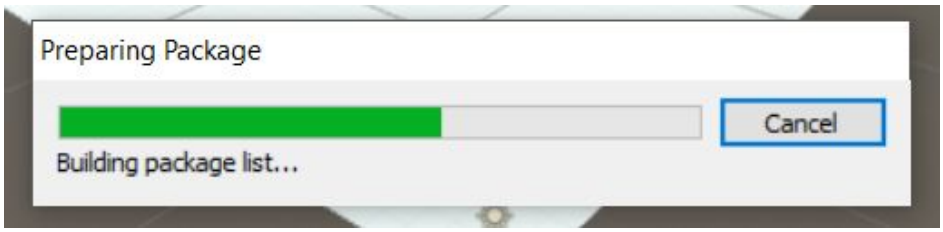
Par défaut il n'y a pas de texture dans votre terrain, il faudra donc les ajouter dans le projet.
La première texture que vous ajouterez recouvrira tout votre terrain et sera votre "texture de base".
Les autres textures seront à peindre par dessus.

Comment rajouter des textures a mon projet ?

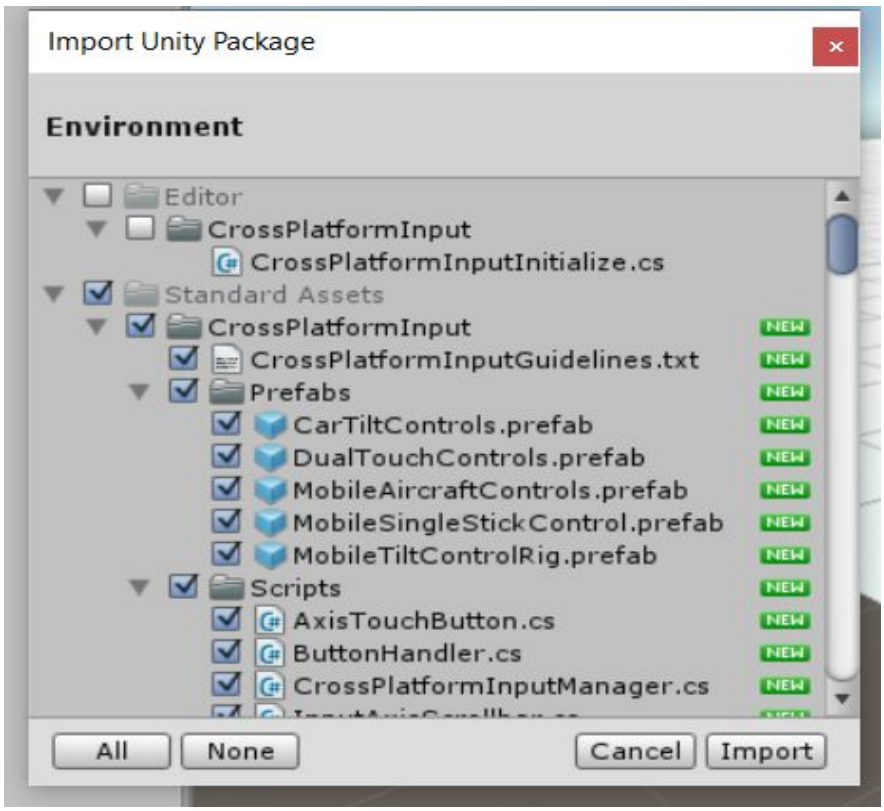
Tout d'abord je vous conseille de (re)lire le [Tuto textures & materials].
Dans unity on peut importer (les rajouter) de nombreux "Package". Un package peut contenir tous les types d'objets de Unity. On peut les télécharger en ligne, pour cela lisez le [Tuto Asset Store].
Il existe quelques "packages" directement présent dans Unity ; nous allons utiliser ici le package "Environment" qui contient des terrains et tout ce qui touche à des décors 3D.
Dans le projet, clic-droit / Import Package / Environment :



Laissez charger:



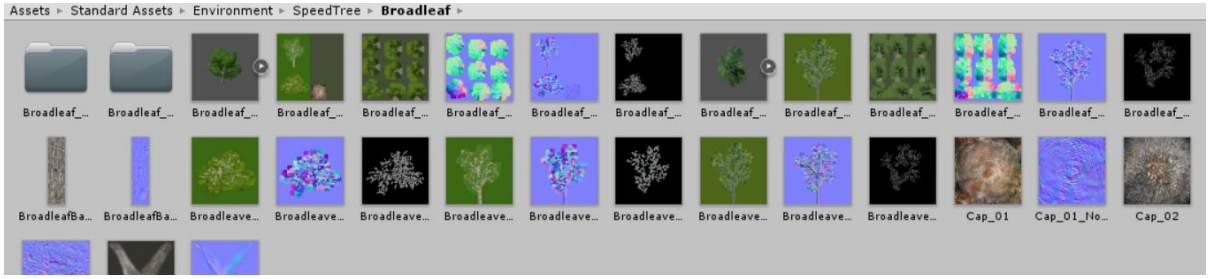
Puis cliquez sur “Import”:



Dans le dossier vous trouverez :

- des textures de base,

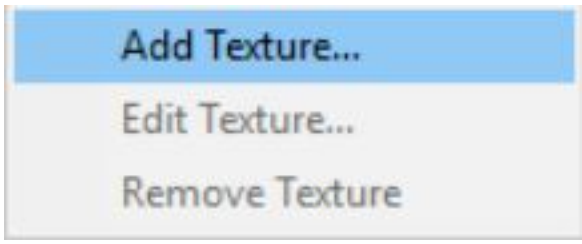
- des modèle d’arbre de base ainsi que des herbes et plantes,



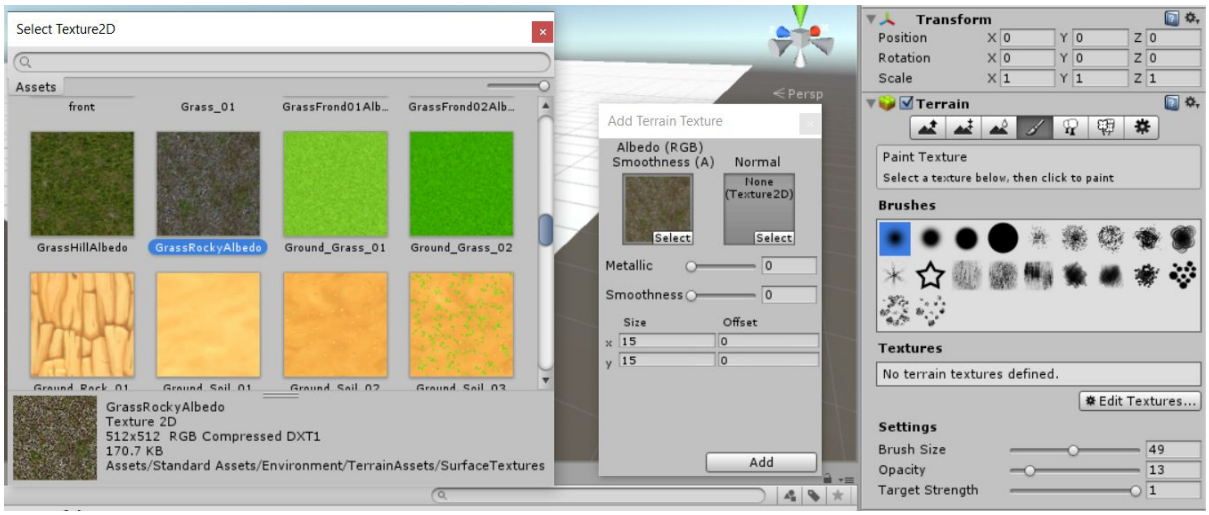
- des prefabs qui permettent d’avoir une eau réaliste

Une fois que ce package est importé il est temps de placer ces éléments dans notre éditeur de terrain.

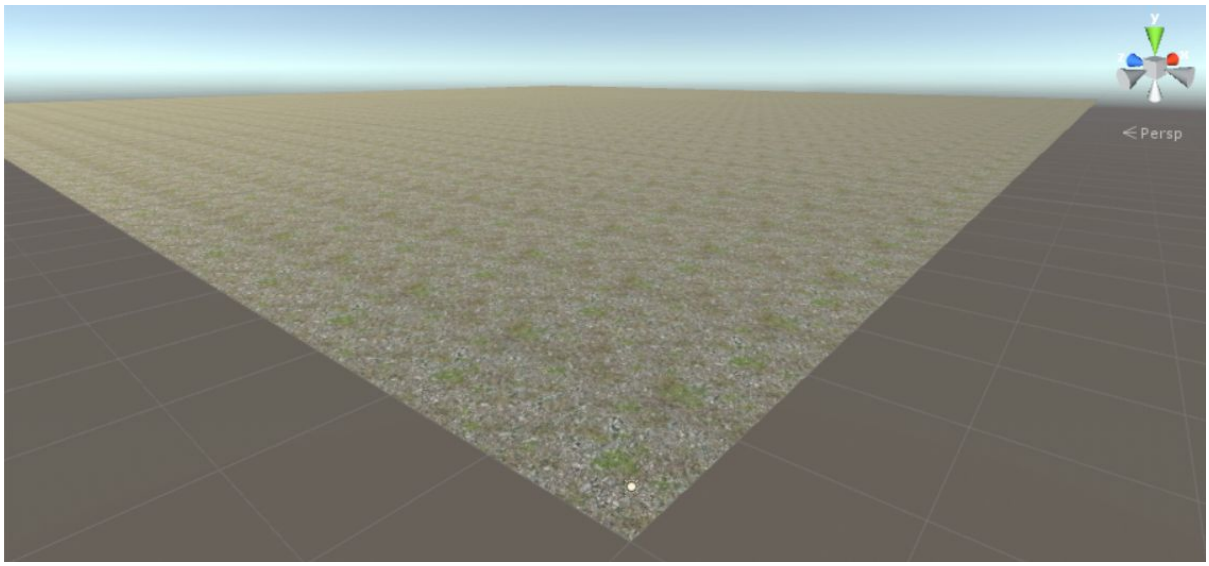
Puis :



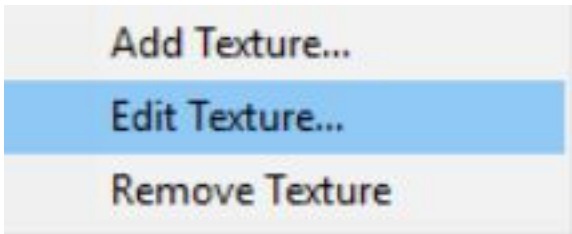
Cliquez sur Select dans la case Albedo puis choisissez la texture de votre Terrain



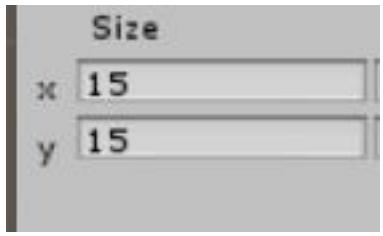
Une fois que vous avez cliquez sur “Add” votre terrain sera recouvert par la première texture que vous avez choisi :



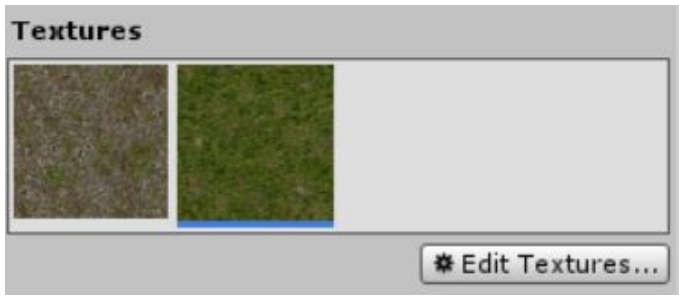
A tout moment vous pouvez modifier la taille de la texture à appliquer sur votre terrain:

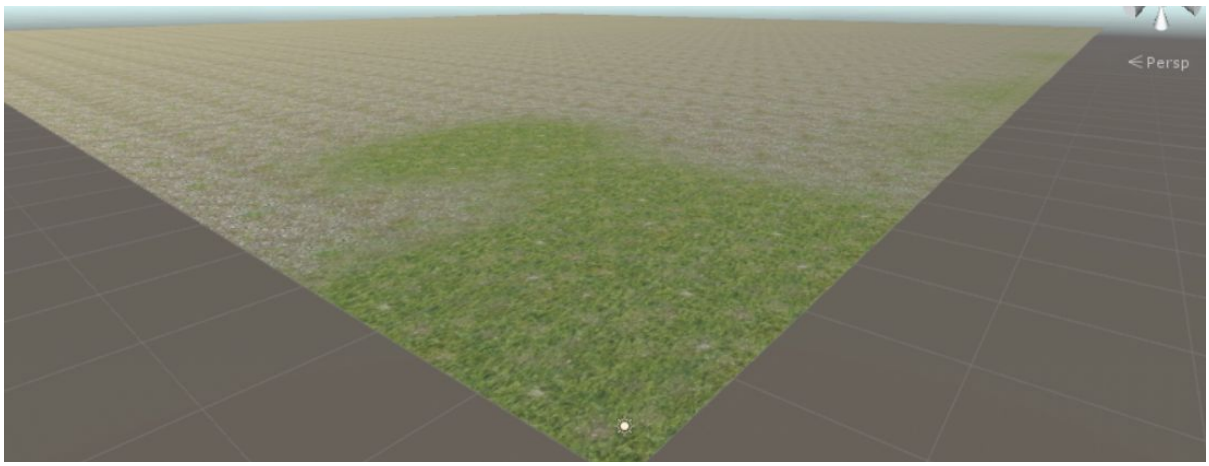


Vous pouvez diminuer dans “Size” le x et le y pour avoir un meilleur rendu (texture moins floue)

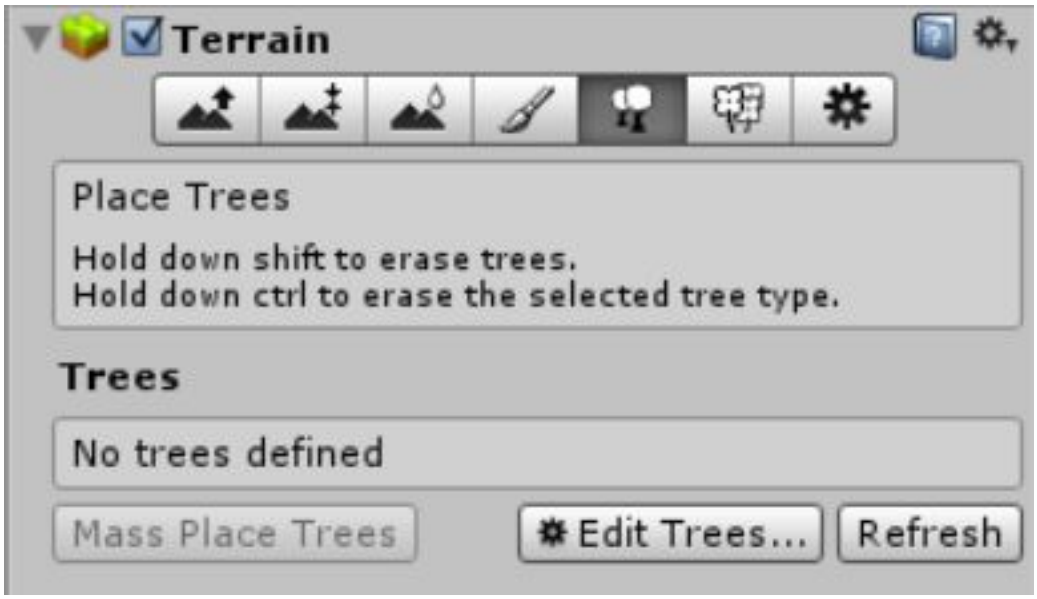


Vous pouvez rajouter d’autres Textures pour les peindre par dessus la texture de base :

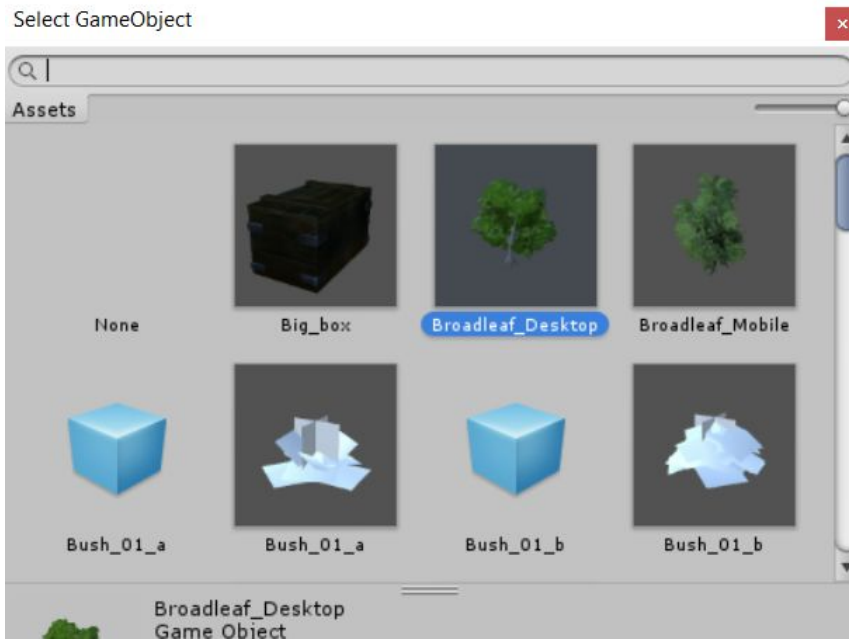




5. Les arbres



Tout comme pour les textures vous devez d'abord ajouter des arbres via "Edit Trees".



Vous avez désormais accès à plusieurs propriétés pour gérer les arbres :

Brush Size : taille du pinceau.

Tree Density : gère le nombre d'arbres

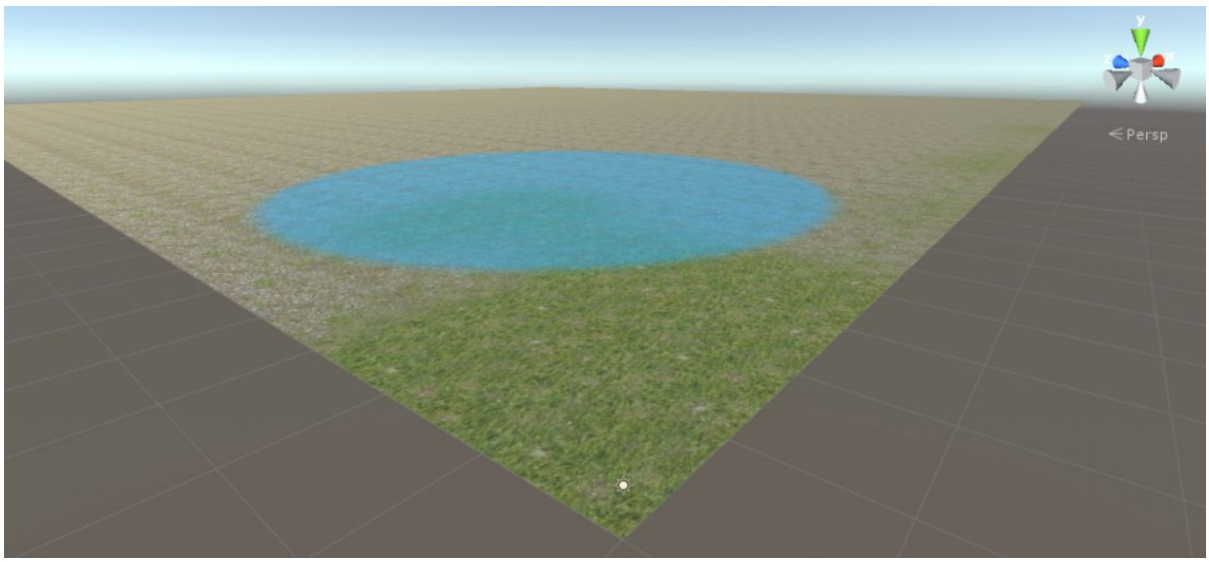
à faire apparaître dans la zone peinte.

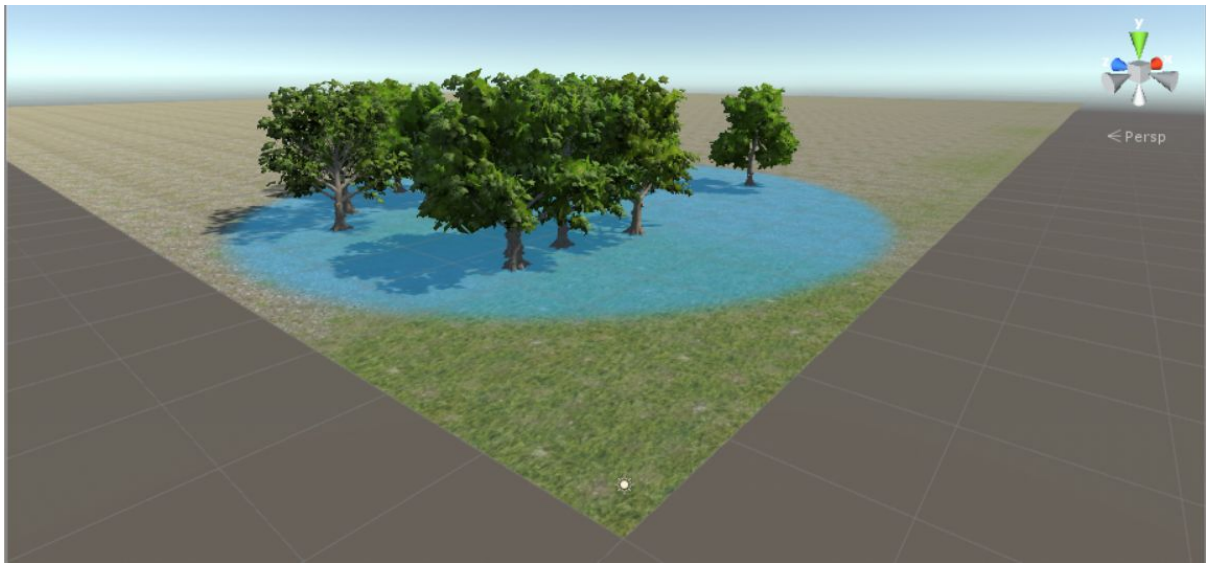
Tree Height : vous pouvez étirer les bord fléchés pour augmenter l'intervalle de taille aléatoire.

Lock Width to Height : préserve les ratios lors des changements de taille.

Random Tree Rotation : change la rotation des arbres de façon aléatoire.

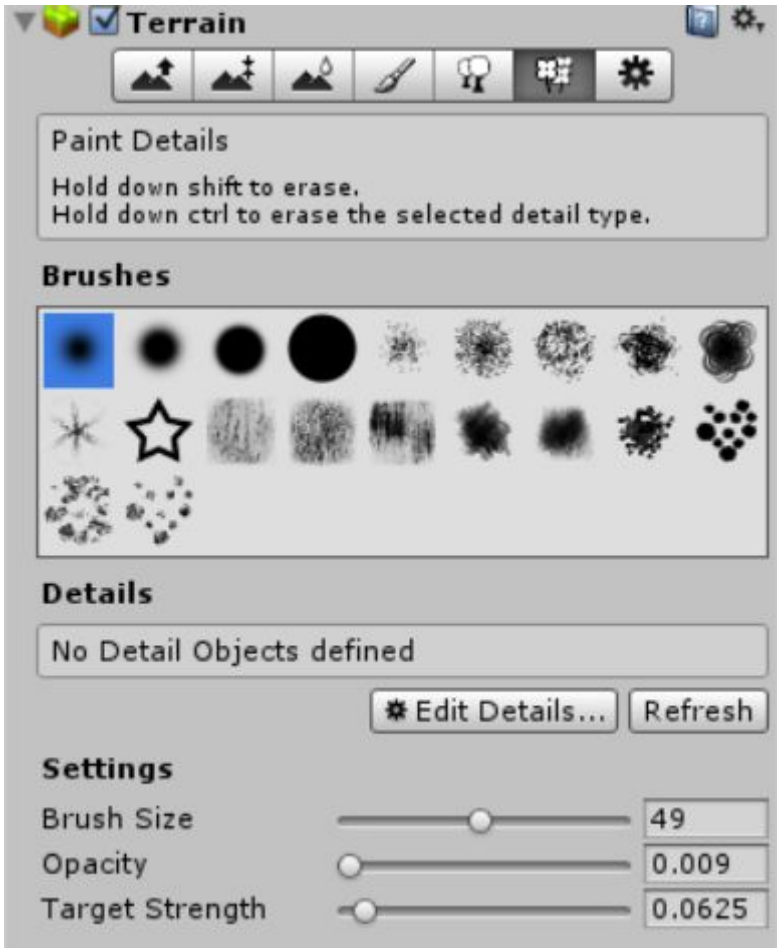
Choisissez une zone :



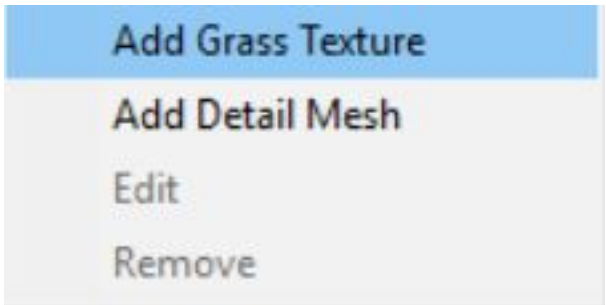


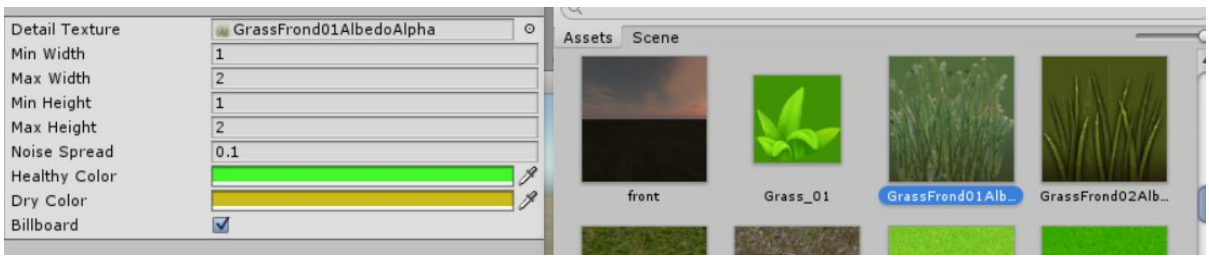
Peignez par dessus pour faire apparaître votre forêt.

6.L’herbe

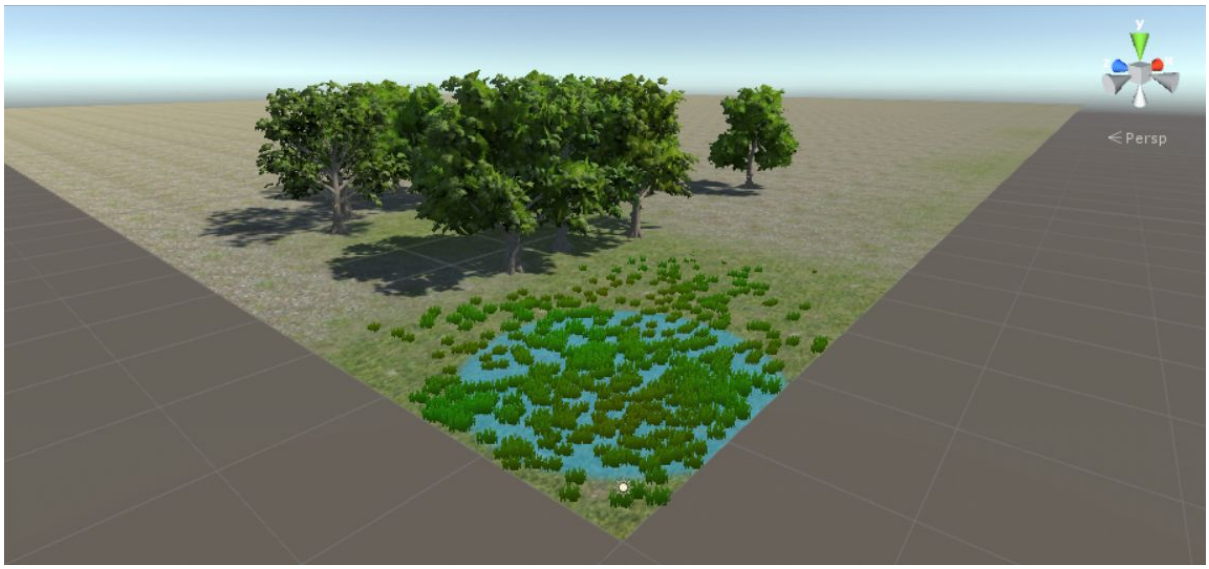


On rajoute de l'herbe en cliquant sur "Edit Details" comme pour les arbres et les textures :





Il suffit de sélectionner une zone et de peindre par dessus, comme les arbres :

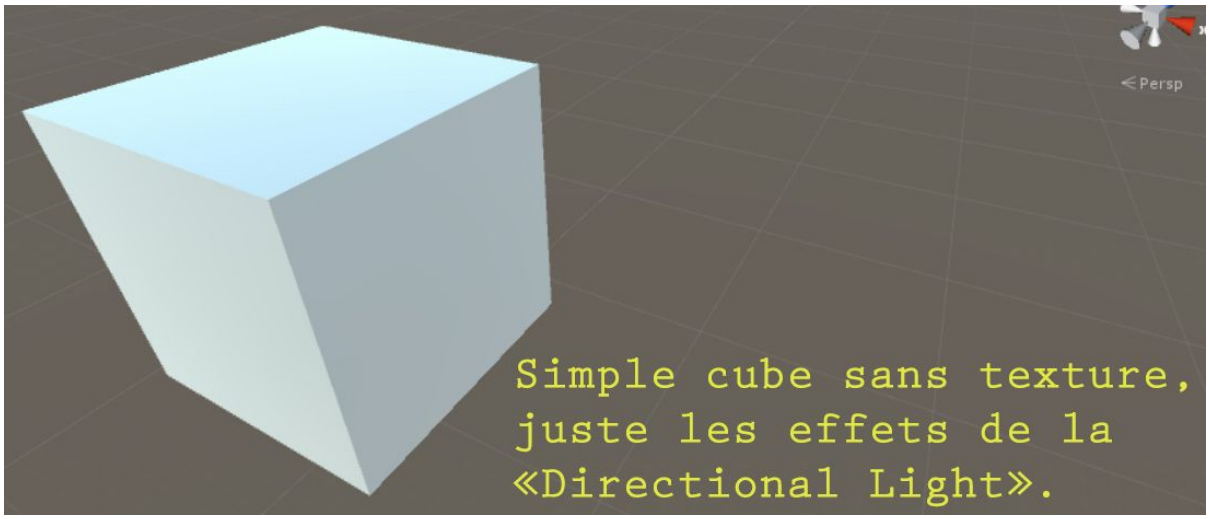

Persp

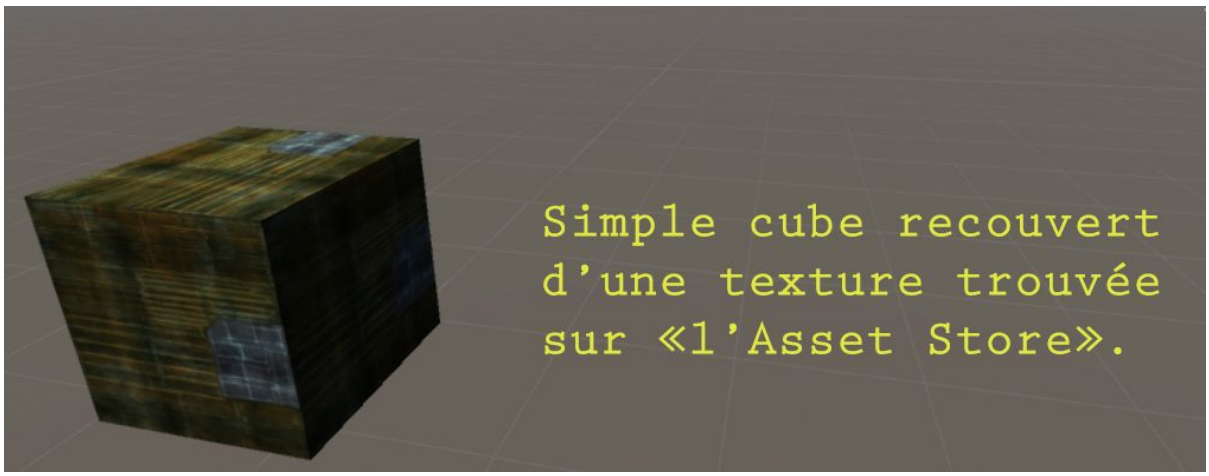
1.3 - Textures et Materials

Une texture est une tout simplement une image utilisé par Unity de plusieurs manières.

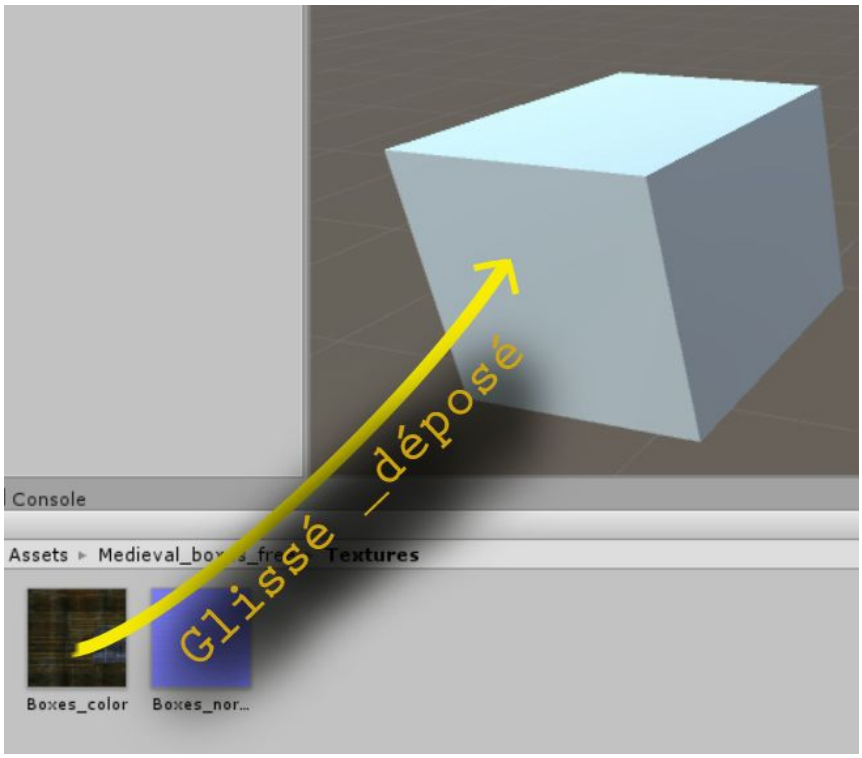
1. Changer la surface

L'application la plus courante pour une texture ; on la colle sur chaque côté : c'est ce qu'on avait fait pour le terrain (qui n'a qu'un côté, le dessus).



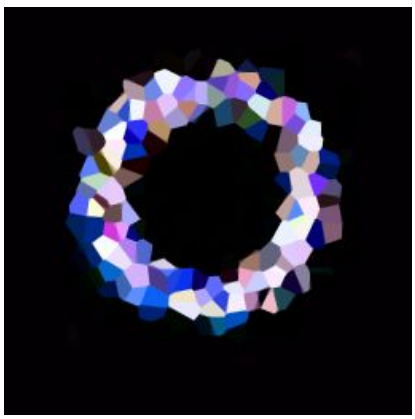


Comment appliquer une texture à un objet ?
Choisissez une texture que vous avez importé soit depuis "l'Asset Store" soit depuis votre ordinateur (glisser-déposer depuis l'endroit où votre texture est sauvegardée jusqu'au dossier "graphics" de votre projet).
Double-cliquez dans la hiérarchie sur l'objet de votre choix, puis faites glisser votre texture sur l'objet dans votre scène.

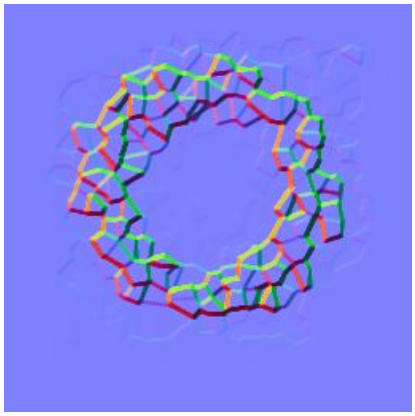


1. Création de “normal map” et “height map”

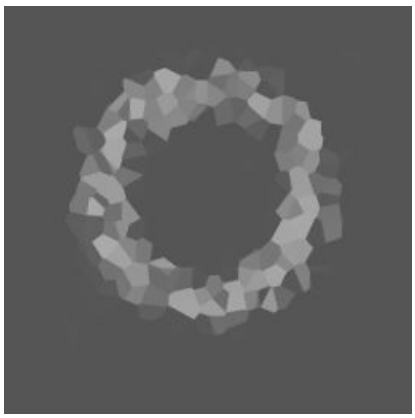
Qu’est ce qu’une “Normal map” et une “Heigth map” ?
Ce sont des image presque identiques à la première mais dont les couleurs qui les composent servent à déterminer le relief de la texture.

Exemple: 

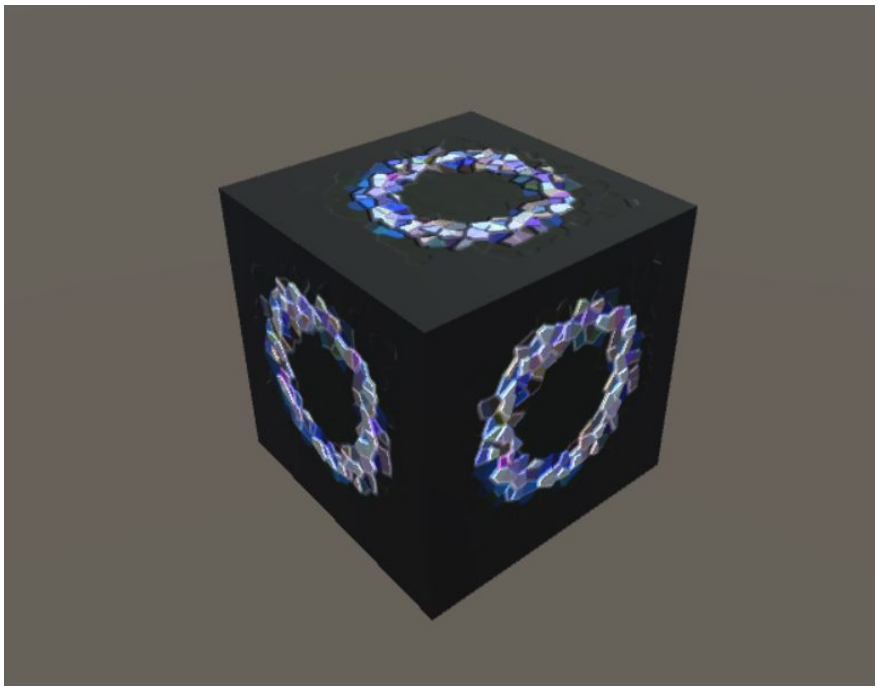
- ma texture :



- ma “Normal map”:



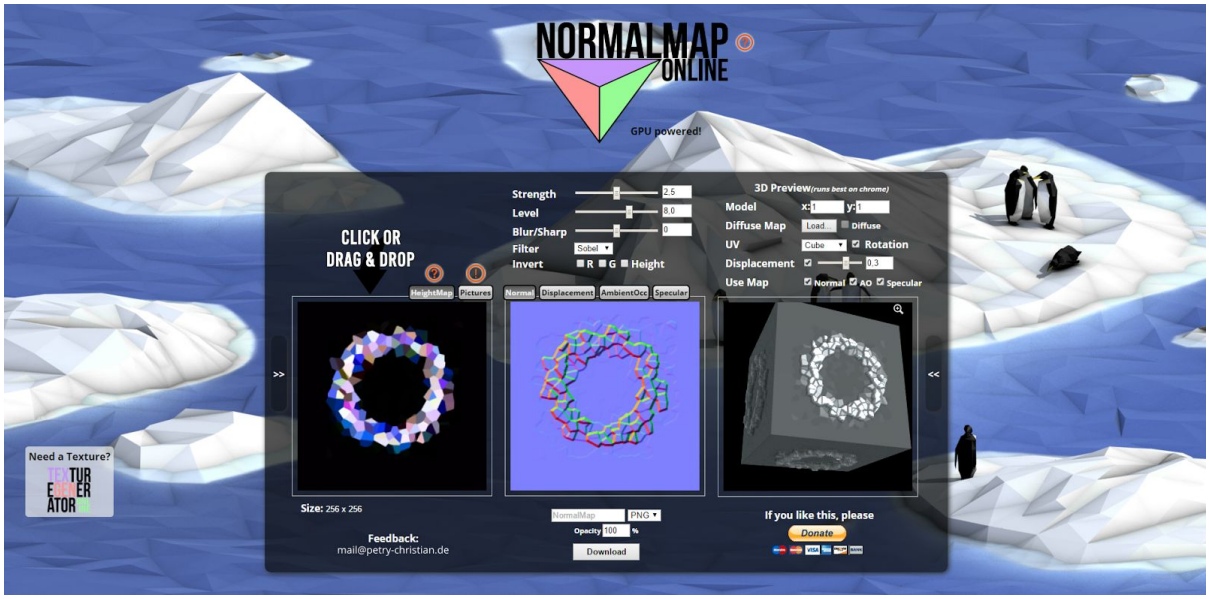
- ma “Height map”:



- Mon rendu final :

Comment créer une “Normal map” ou une “Height map” à partir d’une image ?

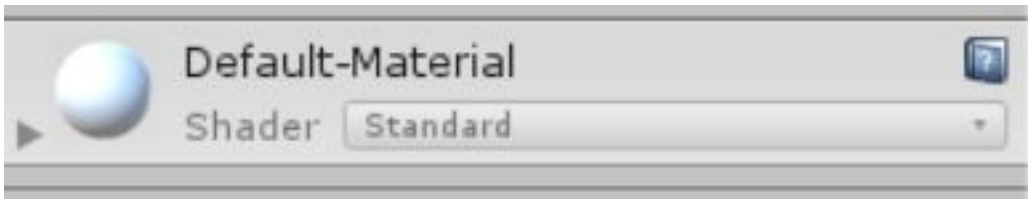
Le plus simple est d'utiliser un générateur automatique de normal map tel que "Normal-Map online".



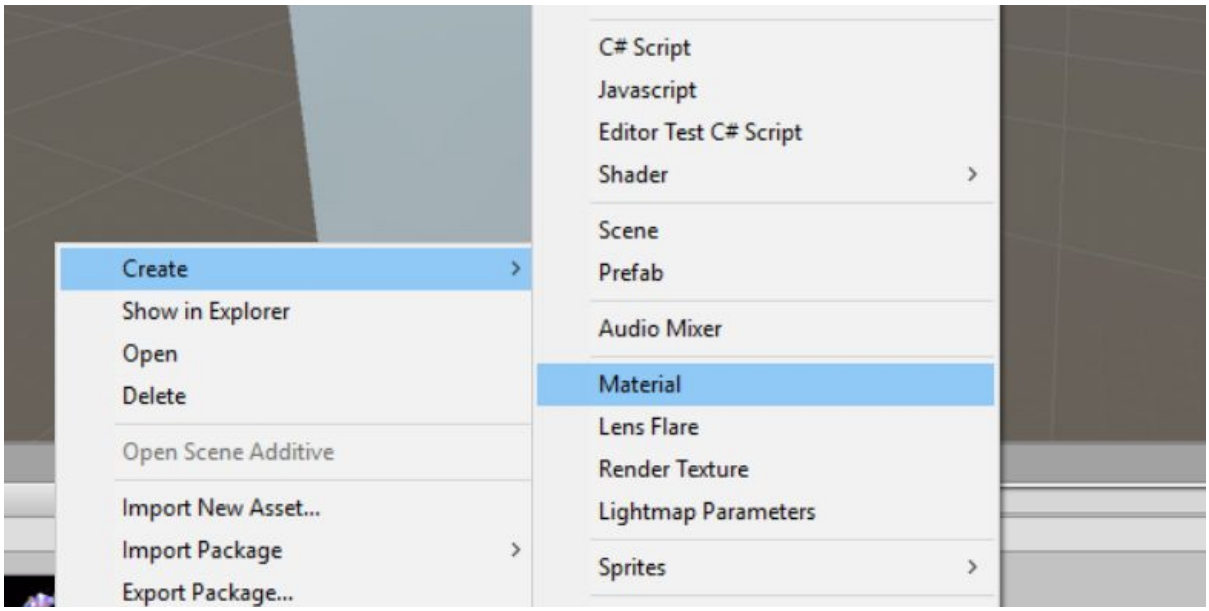
/!\ Attention : Sur certaines applications (comme ce site), "Height map" s'appelle aussi "DisplacementMap".

Comment appliquer une "Normal map" ou une "Height map" a mon cube ?

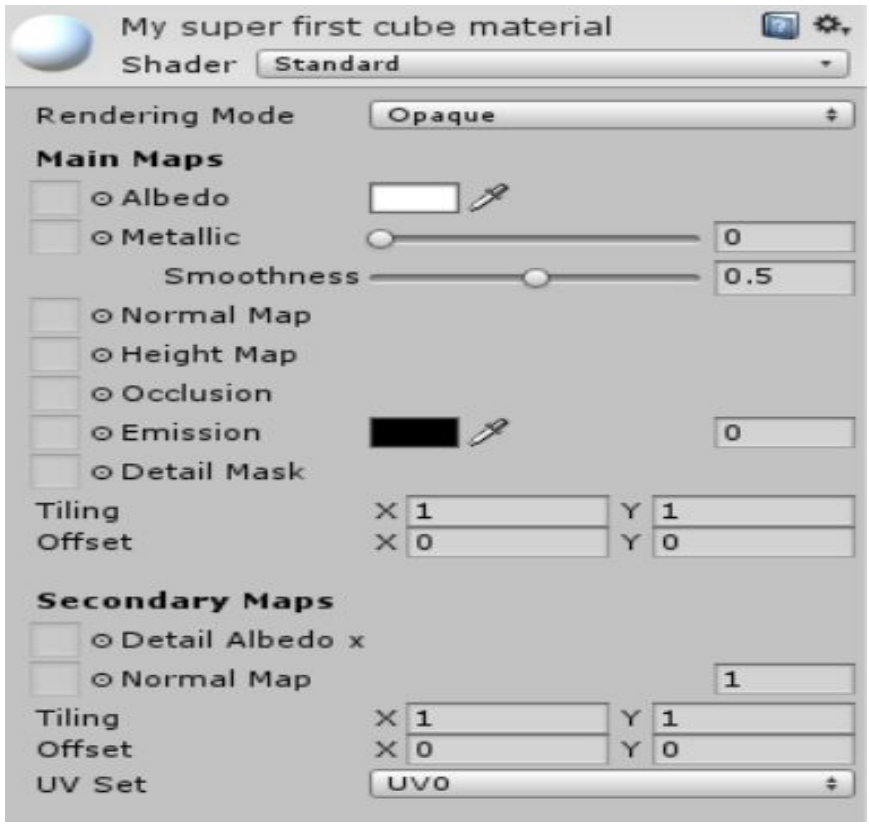
Dans l'inspecteur, vous pouvez voir que votre cube possède un composant appelé "Default material":



De base, si vous n'avez pas fait de glisser-déposer de texture dessus toutes ses propriétés sont grisées.
Il va falloir créer un nouveau "Material".
Dans votre dossier "graphics" faite un clic droit et créez un nouveau "material" que vous allez nommer "my first super cube material" et l'attacher à votre cube en faisant un glisser-déposer sur le cube.



Une fois créé et attaché à votre cube, sélectionnez le et rendez vous dans l'inspecteur.

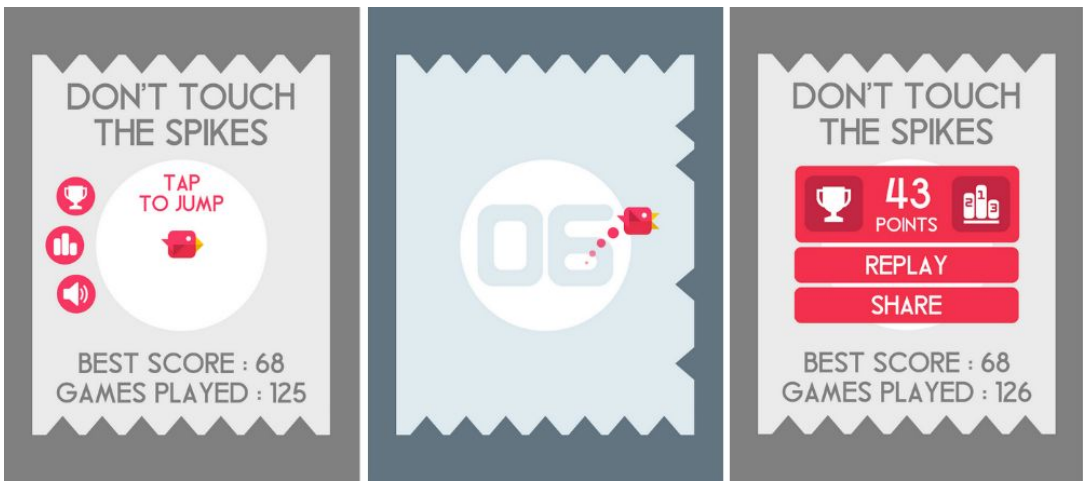


Parmi tous les paramètres disponibles nous allons en utiliser que 3 :

- "Albedo" : c'est la case correspondante à notre texture (comme vu dans le tuto texture)
- "Normal Map": Comme son nom l'indique

- “Heigh Map” : Idem
  Pour placer vos images dans les différentes catégorie un simple glisser déposer sur la case grise à gauche du nom de catégorie suffit.

-> Astuce : Il n’est pas nécessaire d’avoir des graphismes très poussés, on peut faire un jeu très agréable à jouer avec des textures et des modèles simples. Quelques exemples :

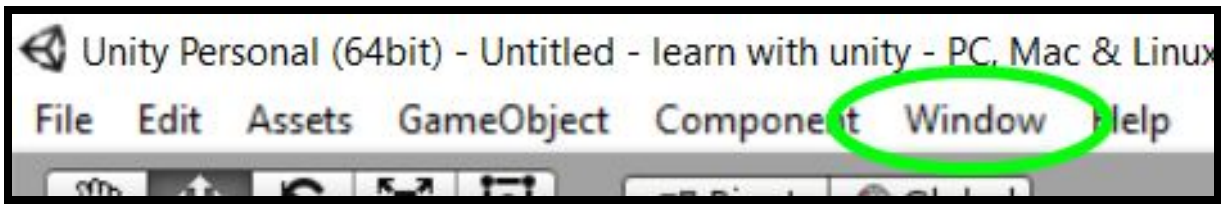


1.4 - La Skybox.

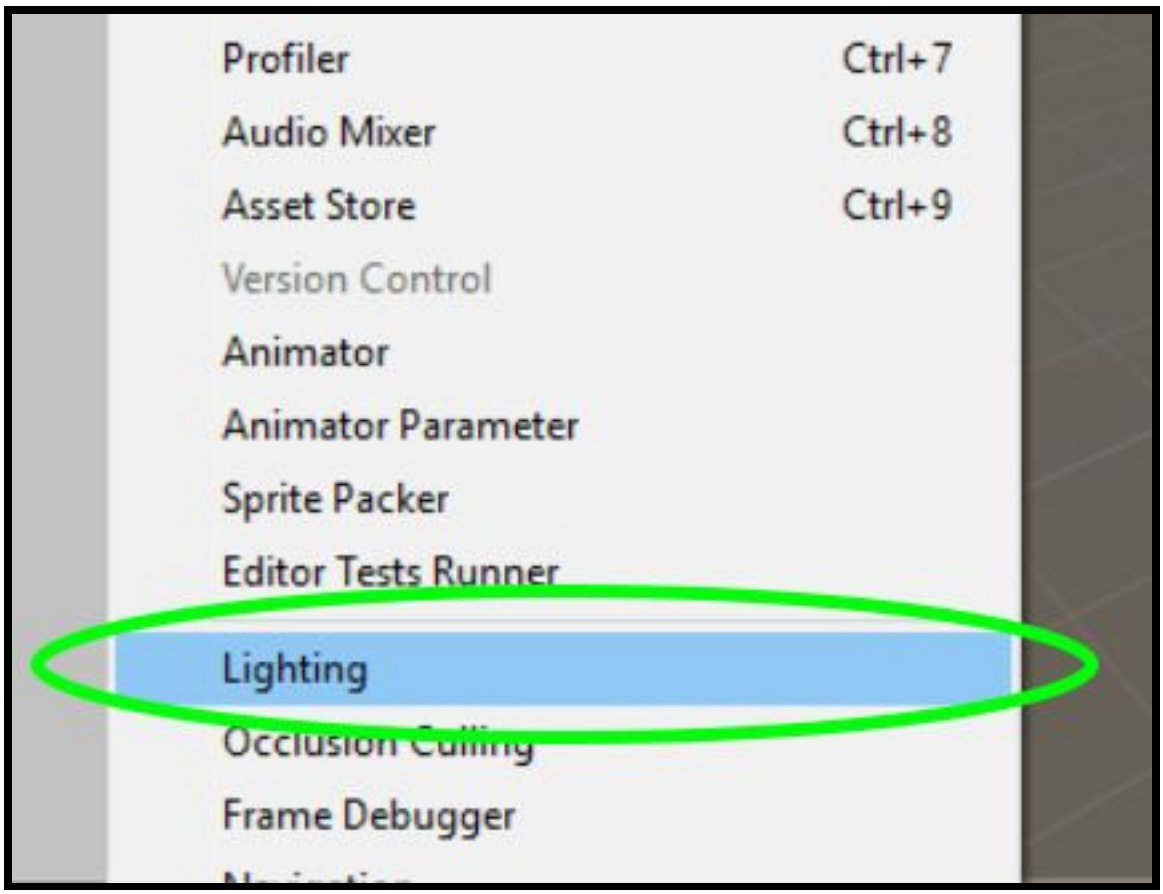
Comment changer le ciel de mon jeu ?

Nous avons téléchargé une “skybox” lors du tutoriel sur l’Asset Store. Nous allons maintenant la rajouter dans le jeu.

1. Cliquez sur Window:



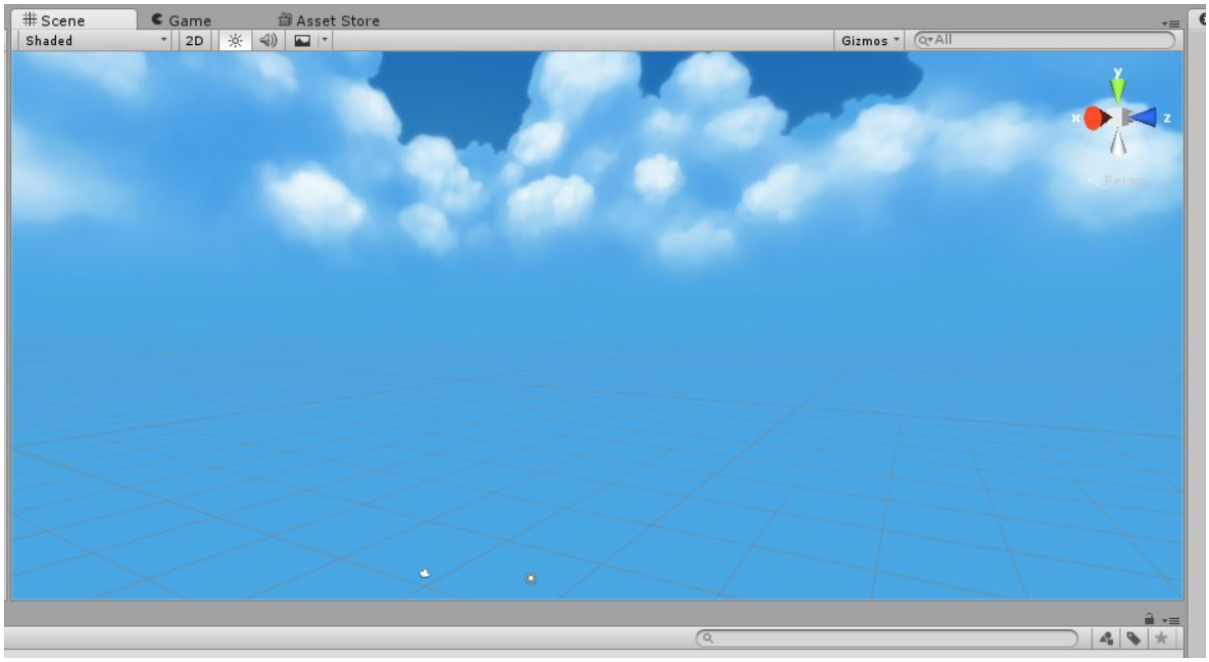
2. Ouvrez Lighting:



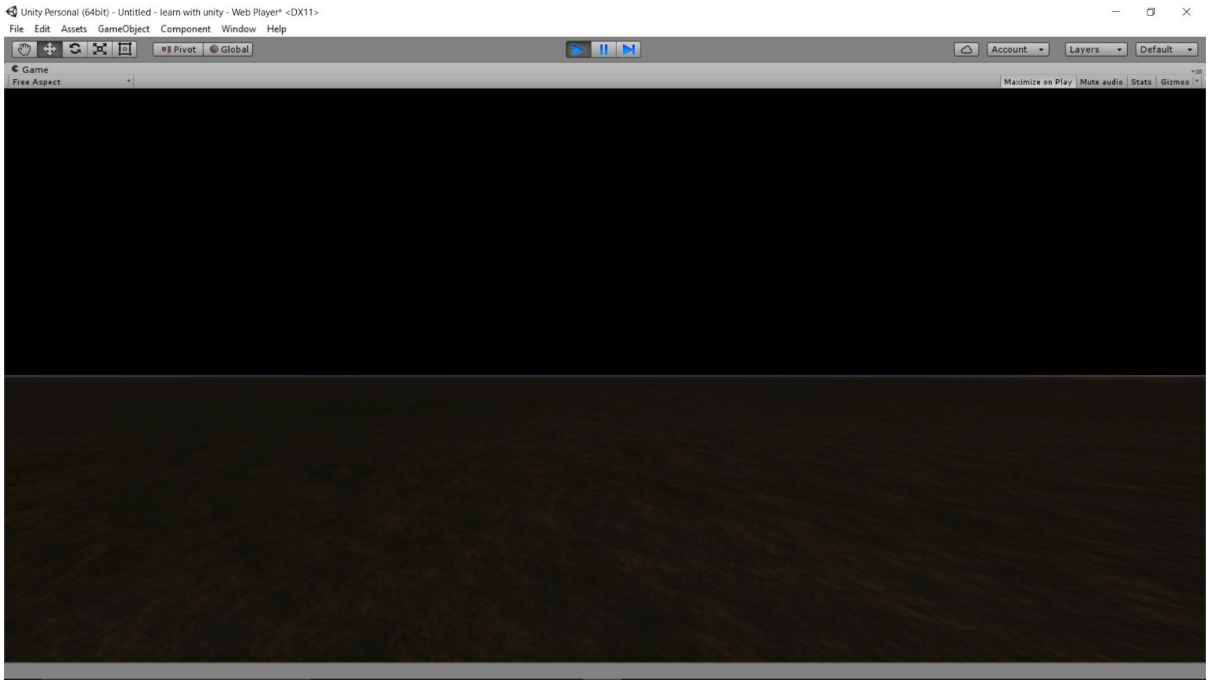
3. Cliquez sur le point à coté de l’encadré Skybox (chercher un fichier dans le projet) :

4. Parmi les différentes textures et materials qui se trouve dans le projet cherchez les élément qui correspondent à votre skybox:

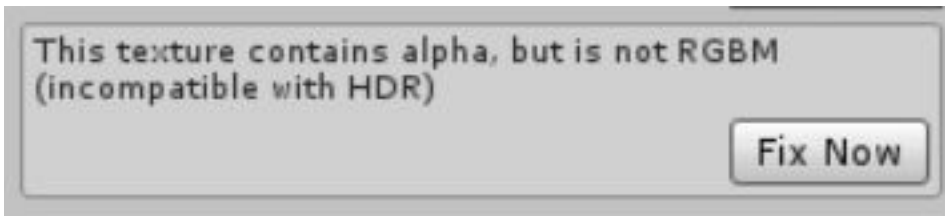
Vous devriez avoir une scène comme suit :



/!\ Si ce n'est pas le cas et que vous avez quelque chose qui ressemble à ça :



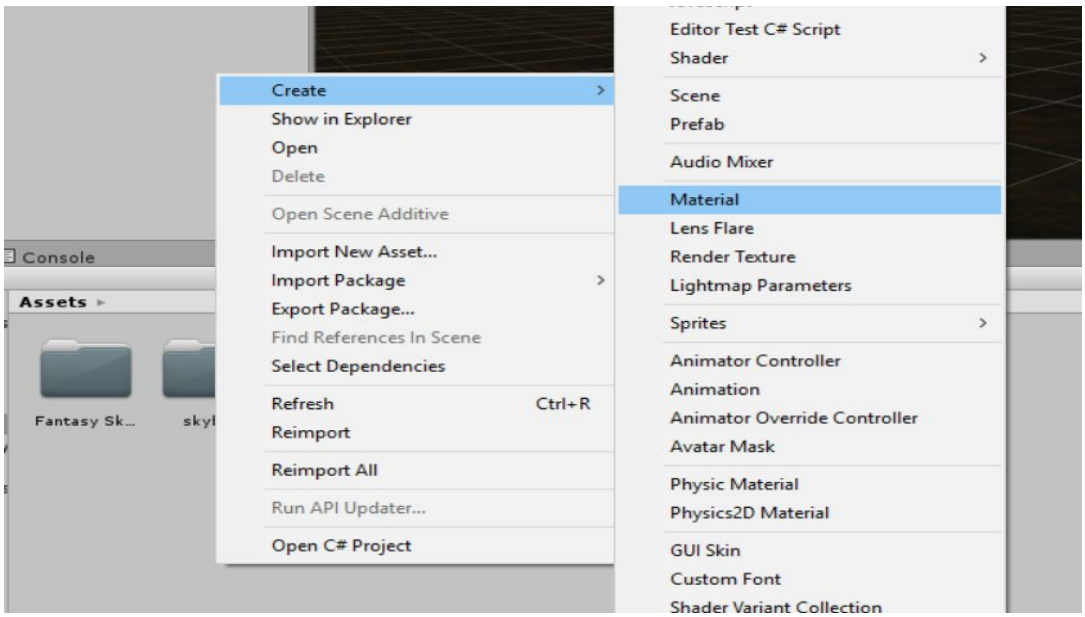
C’est que la “skybox” utilise un vieux style de rendu non adapté a la version actuelle.
Pour corriger cela ouvrez le dossier de votre “skybox” et trouvez la “skybox” globale (pas les 6 images correspondant aux 6 faces). Puis dans l’inspecteur cliquez sur “Fix Now” sur chacune des faces :



Comment créer sa propre skybox ?
En effet vous serez peut être amenés à créer votre propre “skybox” si vous n’en trouvez pas une à votre goût sur “l’Asset Store”. Dans unity, une “skybox” est simplement un cube dont les 6 faces sont recouvertes de textures à l’intérieur (et non pas à l’extérieur comme c’est habituellement le cas.)


Inspector
sky,startday
Shader
Skybox/6 Sided
Tint Color
Exposure
1
Rotation
0
Front [+Z] (HDR)
Select
Back [-Z] (HDR)
Select
Left [+X] (HDR)
Select
Right [-X] (HDR)
Select
Up [+Y] (HDR)
Select
Down [-Y] (HDR)
Select
sky,startday
AssetBundle
None
None

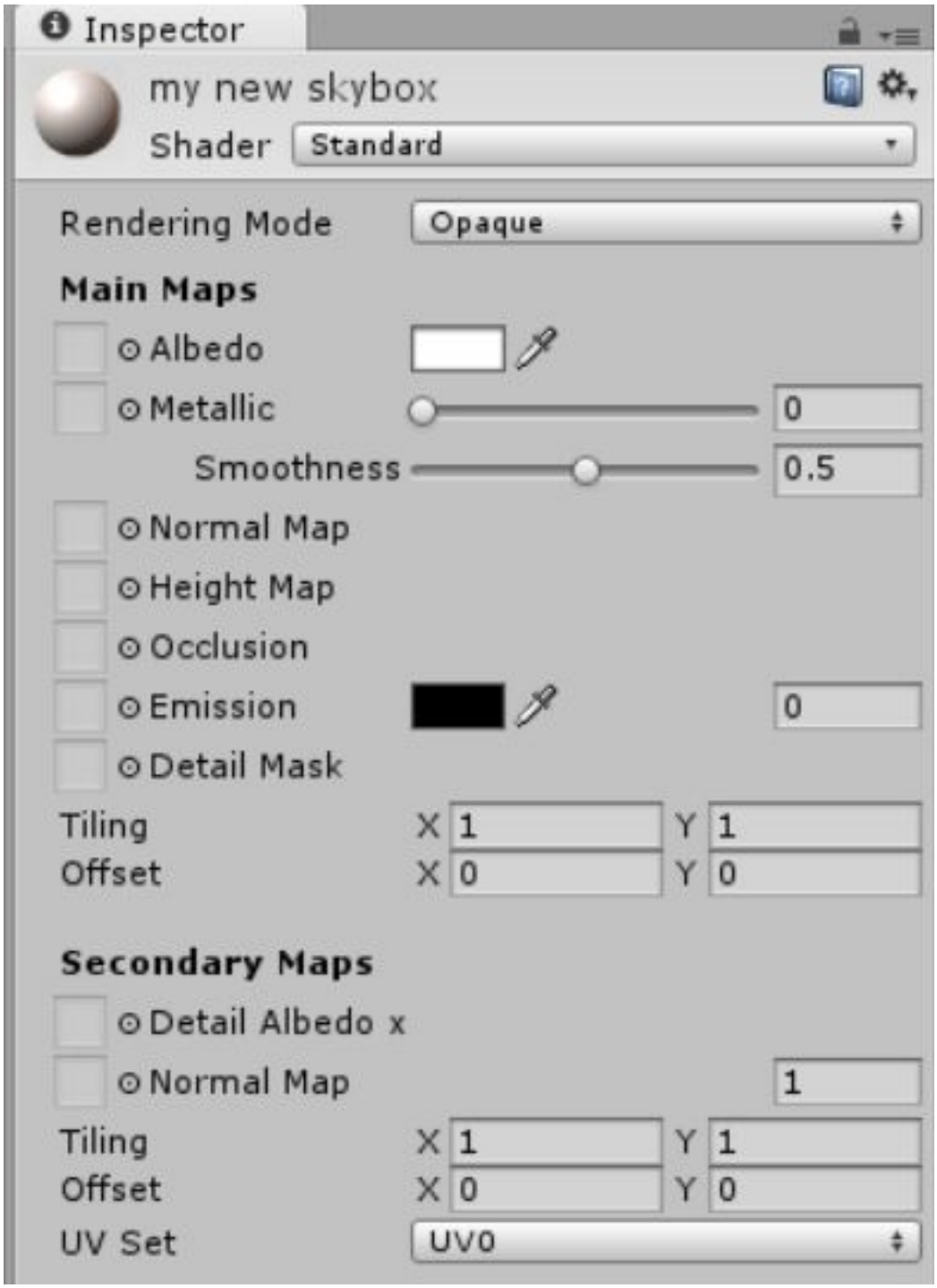
- Comme c’est indiqué en dessous du nom de votre “skybox”, il s’agit d’un type de “Shader”.

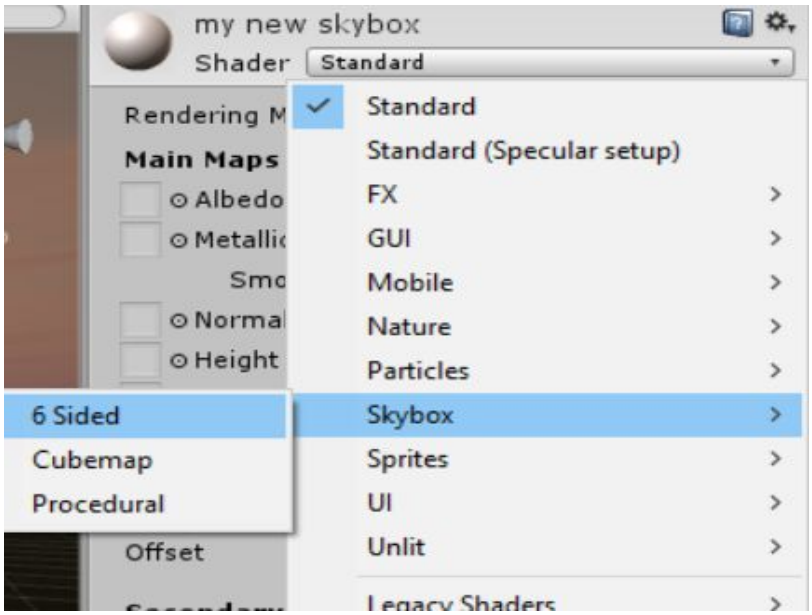


- Commençons par créer un “Material” dans le projet (clic droit dans le dossier “graphics”)

- Ce qui, dans “Project”, nous donne :

- Et dans “Inspector” en sélectionnant le material :

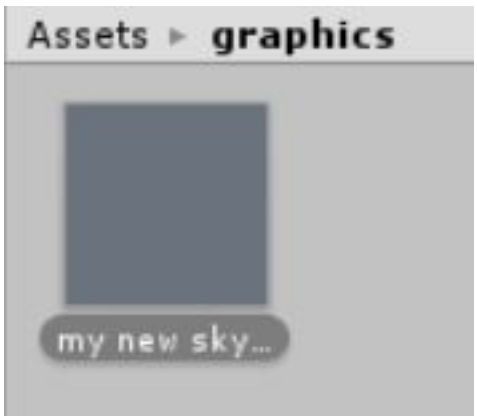

Inspector
my new skybox
Shader
Standard
Rendering Mode
Opaque
Main Maps
Albedo
Metallic
0
Smoothness
0.5
Normal Map
Height Map
Occlusion
Emission
0
Detail Mask
Tiling
X 1
Y 1
Offset
X 0
Y 0
Secondary Maps
Detail Albedo x
Normal Map
1
Tiling
X 1
Y 1
Offset
X 0
Y 0
UV Set
UV0



- Dans shader nous allons sélectionner le bon type de shader puis il ne vous restera plus qu'à remplir les différentes faces :

  Ce qui vous donne cela :

- Dans "Project":



- Et dans "Inspector" :


Inspector
my new skybox
Shader
Skybox/6 Sided
Tint Color
Exposure
1
Rotation
0
Front [+Z] (HDR)
None (Texture)
Select
Back [-Z] (HDR)
None (Texture)
Select
Left [+X] (HDR)
None (Texture)
Select
Right [-X] (HDR)
None (Texture)
Select
Up [+Y] (HDR)
None (Texture)
Select
Down [-Y] (HDR)
None (Texture)
Select
my new skybox
AssetBundle
None
None

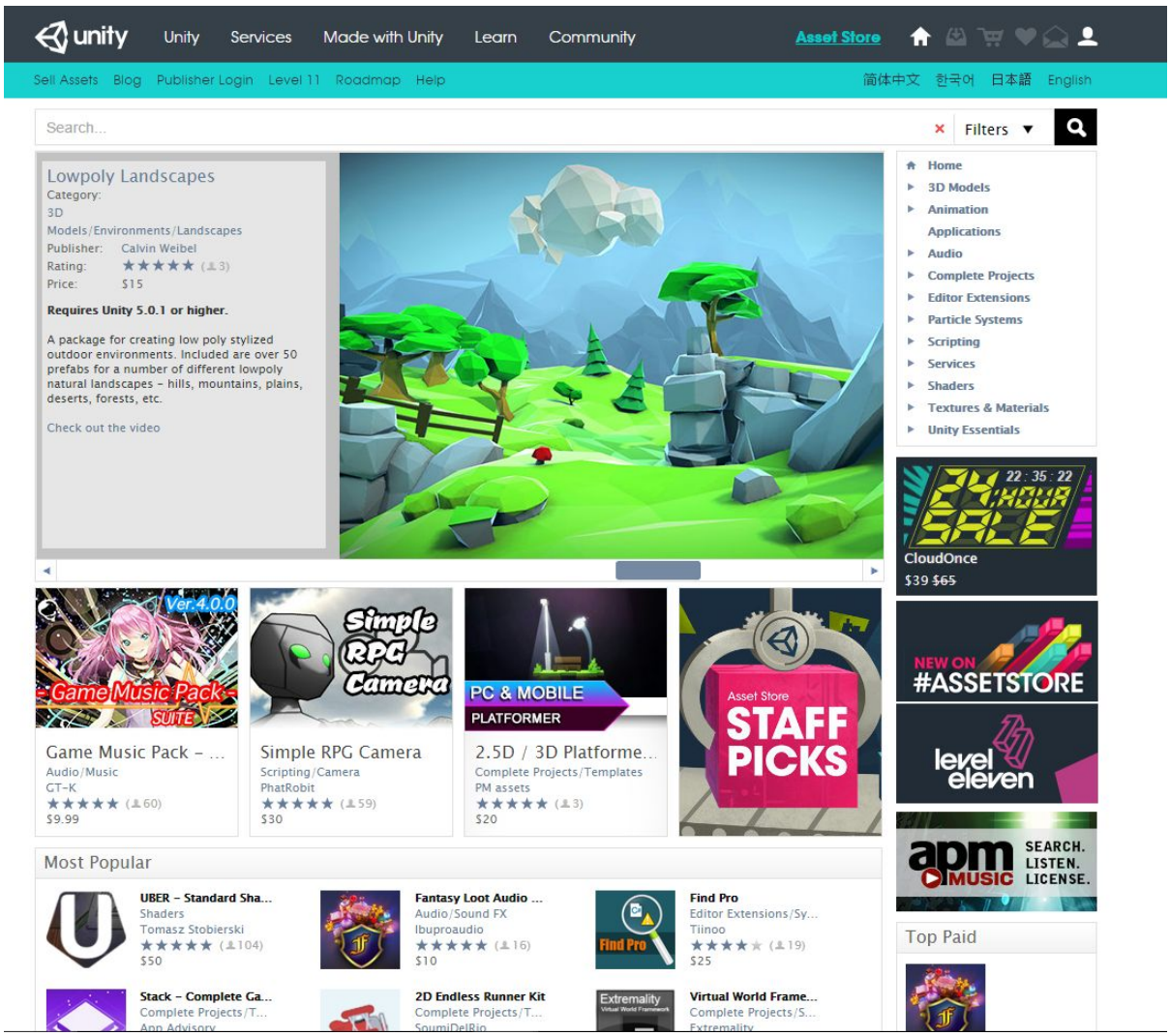
1.5 - Asset store

L'asset store d'unity est un site de partage de fichiers unity. Vous y trouverez des modèles 3D, des effets de particules, des textures, des jeux, des scripts … Tout ce qui est peut être utilisé dans un projet Unity ! Ils sont fournis sous forme de "Package", c'est à dire un dossier contenant plusieurs fichiers (souvent dans des sous-dossiers qui indiquent leur type).

Comment aller dans l'asset store ?

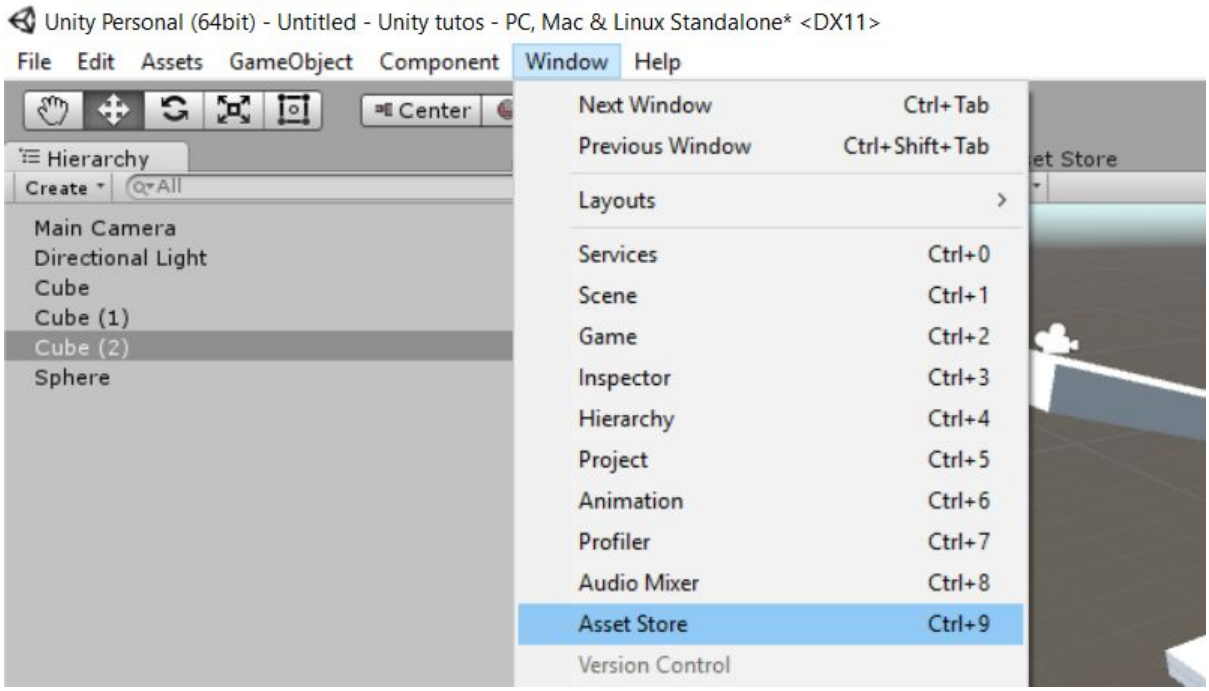
Vous pouvez soit :

1. Aller directement sur le site d'unity dans la rubrique "Asset Store" en suivant ce lien : https://www.assetstore.unity3d.com/en/.

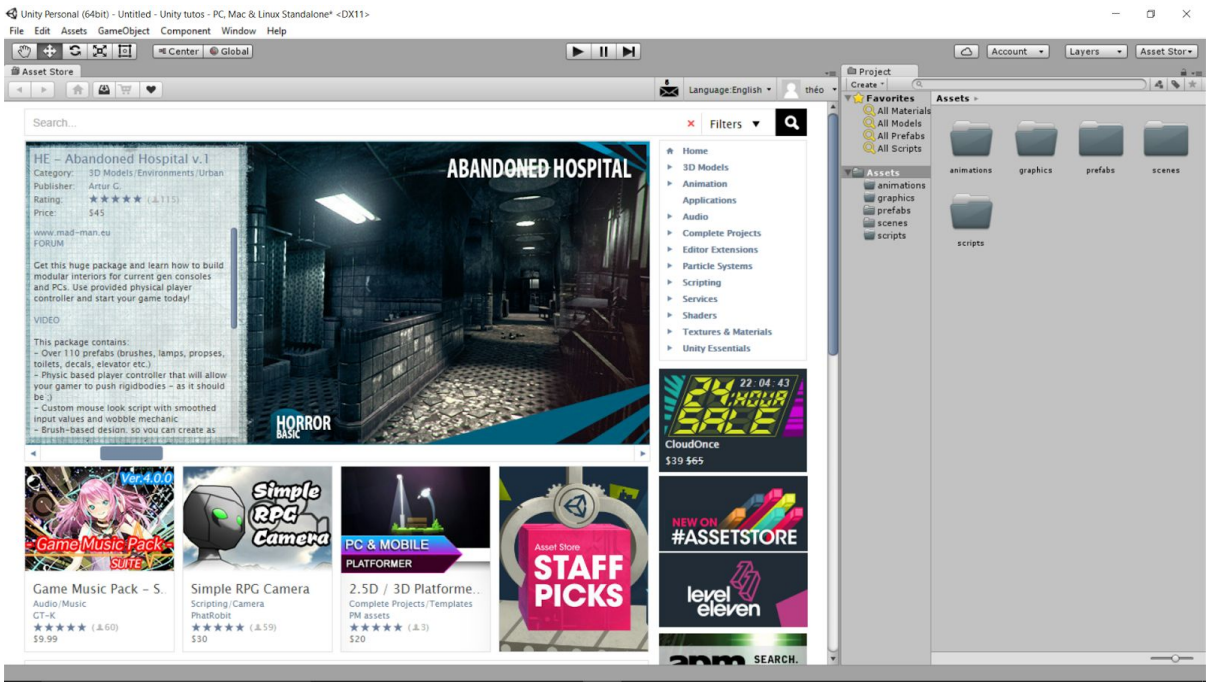


1. Le consulter depuis Unity3D

- Ouvrez la fenêtre de l’asset store depuis l’onglet “Window” dans la barre des tâches en haut à gauche.



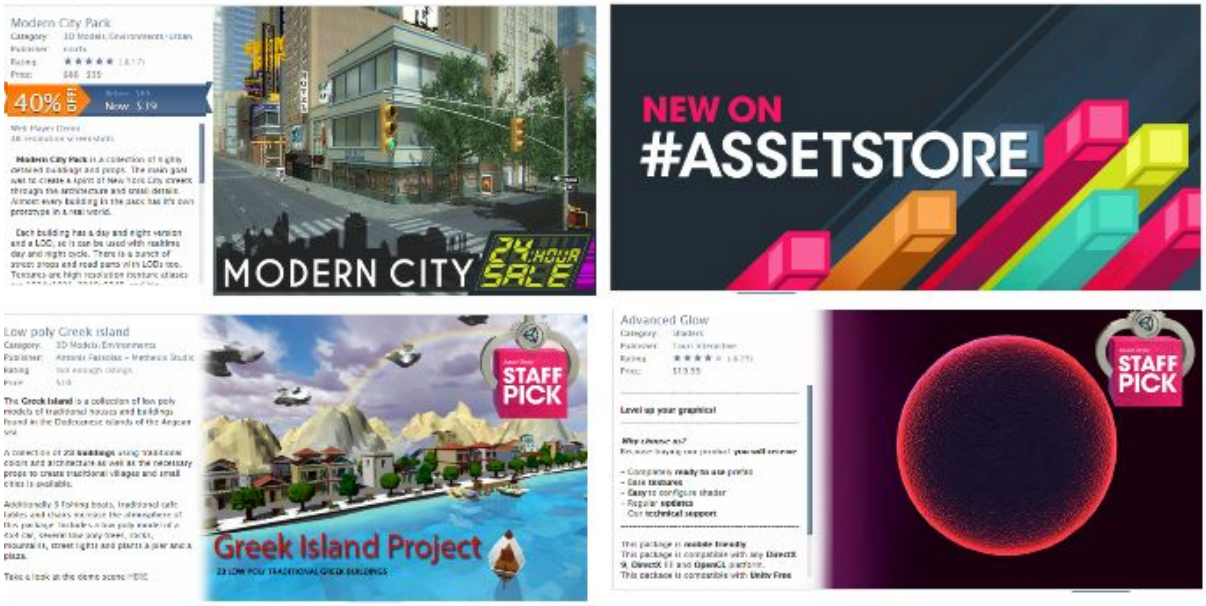
- Choisissez le Layout “Tall” qui facilite la lecture de la page.



Comment se repérer dans l’Asset Store ?

- Une barre de recherche pour y mettre des mots clefs

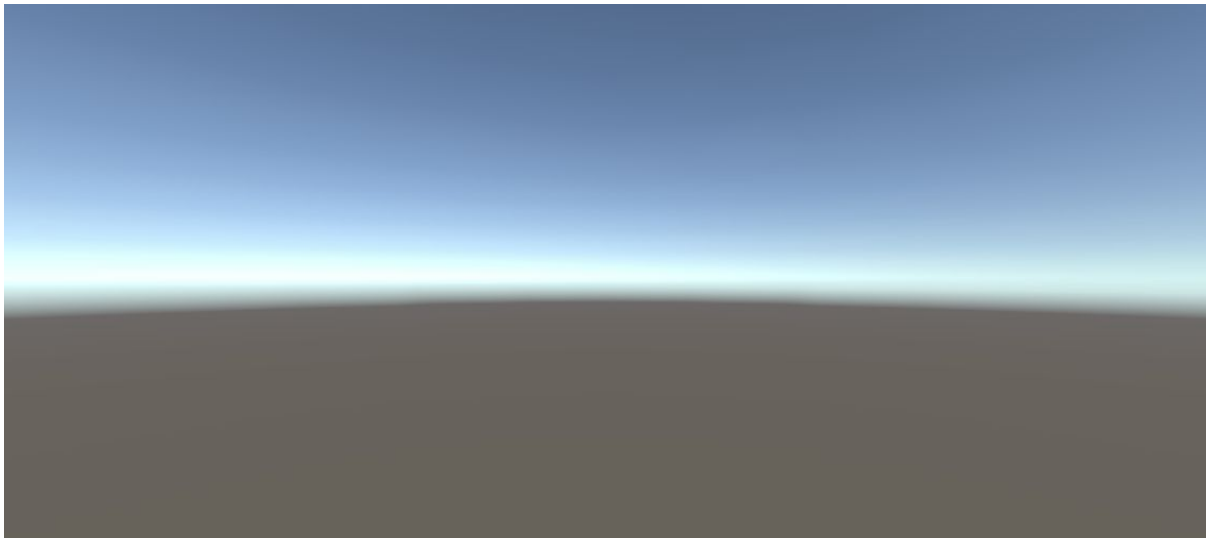
- Des annonces de top tendances .



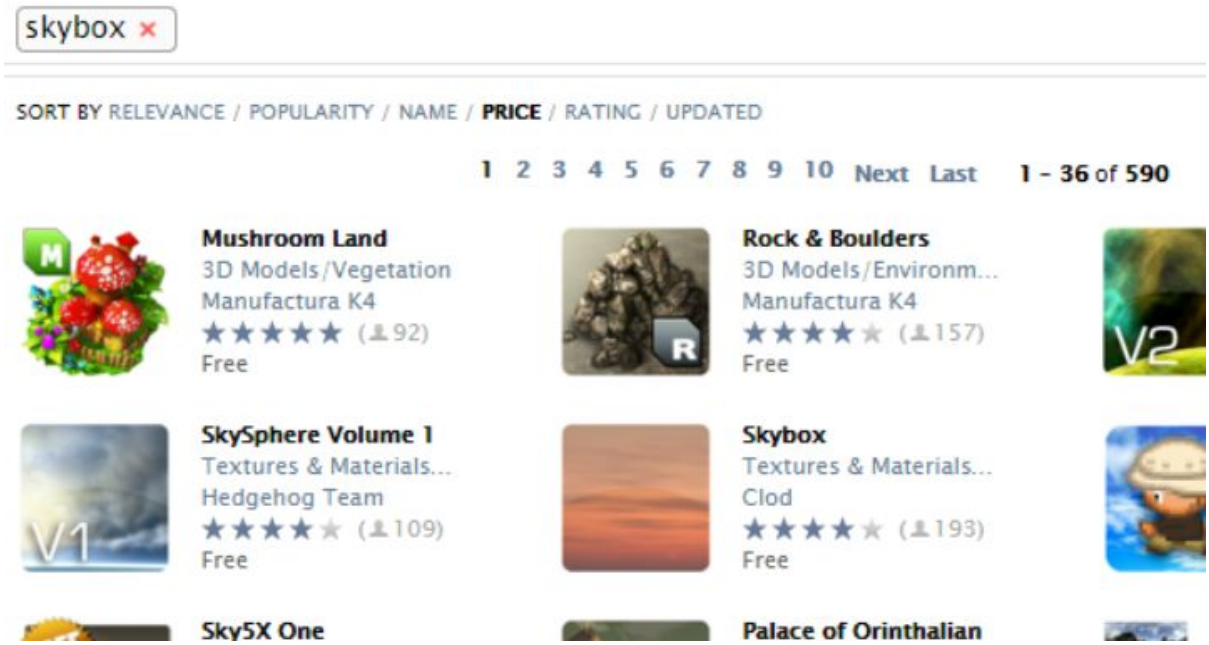
- Un index de recherche.

Comment télécharger depuis l’asset store ?

Nous allons trouver une “Skybox” (que nous utiliserons dans le [Tuto Skybox])c’est à dire une texture de ciel, pour remplacer le ciel par défaut, un exemple :



- Dans l’Asset Store cherchez le mot-clef “Skybox” et cliquez sur “price” pour mettre en avant les assets gratuits :



Il est possible que certains (très) vieux assets ne soient plus compatibles avec les récentes version d’Unity ; si jamais vous avez des problèmes pour utiliser un asset téléchargé sur le store, pensez à vérifier la version (et en prendre un autre plus récent si besoin).

- Cliquez sur download

- Il ne vous reste plus qu'à importer les fichiers dans votre projet en cliquant sur "Import" :


Import Unity Package
Fantasy Skybox FREE
Fantasy Skybox FREE
_Demo
Demo.unity
Sources
Materials
Environment.mat
Meshes
Bush_01_a.fbx
Bush_01_b.fbx
Bush_01_c.fbx
Bush_02_a.fbx
Bush_02_b.fbx
Bush_02_c.fbx
Mushroom_01_a.fbx
NEW
All
None
Cancel
Import

2 - Construit tes personnages

Objectif

Maintenant que tu as un décor génial, il faut lui donner vie ! On va voir comment rajouter pleins de supers personnages dedans et leur ajouter une SpriteSheet pour créer des animations.

Ce que tu vas apprendre

- Créer des personnages
- Leur appliquer une texture
- Créer des personnages avec plusieurs images (pour des animations)

Méthodologie

Pour commencer, on va recréer des Prefab pour nos personnage, relis donc le tutoriel [Tuto GameObject et Prefab].

Tu voudras probablement récupérer des images d'autres personnes, relis le [Tuto Asset Store].

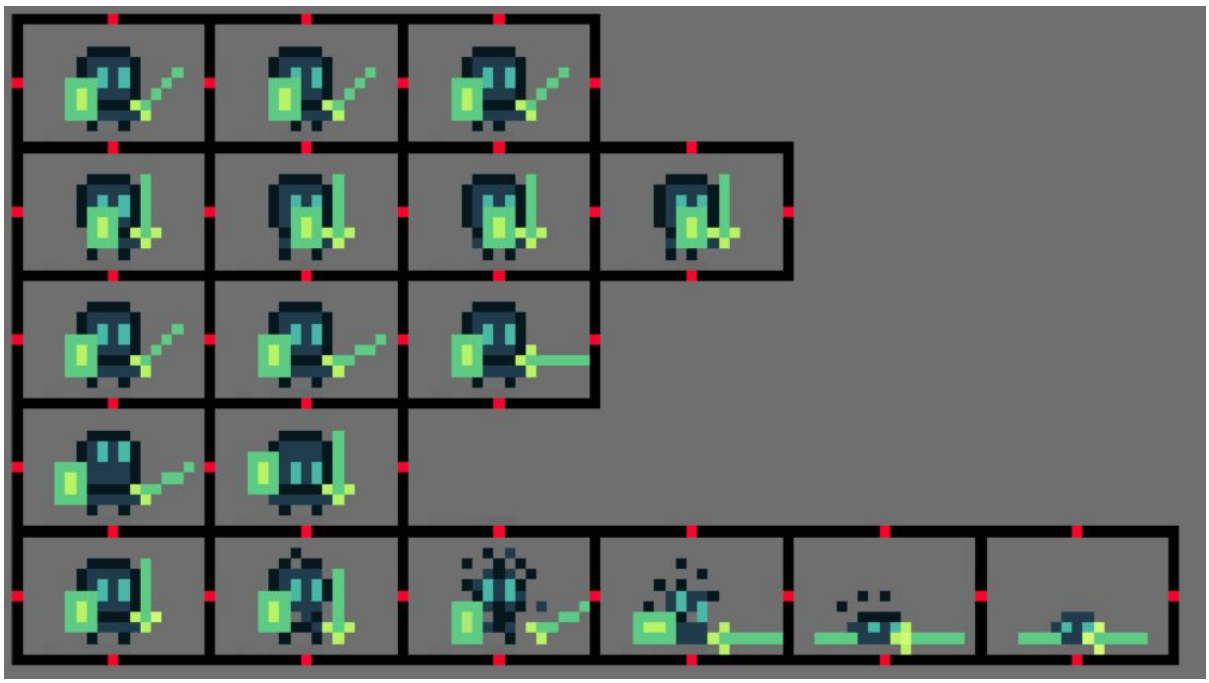
Tu auras ensuite besoin de leur appliquer une texture, relis le [tutoriel TextureMaterials].

Enfin, tu auras besoin de créer une SpriteSheet (utilisée pour créer les animations de ton personnage), lis le [tutoriel SpriteSheet]!

2.1 - Créer une Sprite Sheet 2D

1. Dans un projet 2D

Dans un projet 2D il existe principalement deux manières d'animer un objet.
La première est d'utiliser une "Sprite Sheet" : un document qui contient toutes les images décrivant un mouvement.
Par exemple une "Sprite Sheet" de personnage et une "Sprite Sheet" de petits éléments (goutte d'eau, flamme, coeur) :



Le principe est simple : on va faire défiler les images tout au long de l'animation.

Trouver des images sur Internet

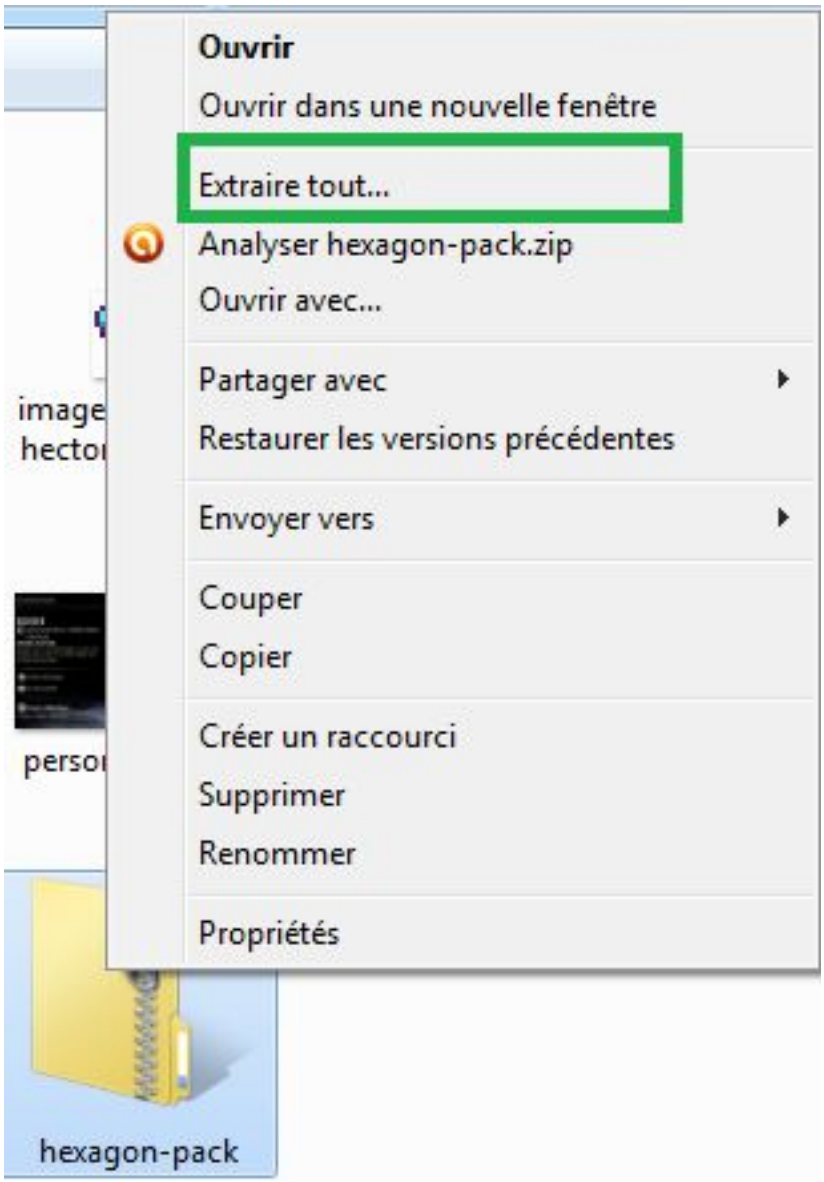
Tu peux télécharger n'importe quelle image que tu trouves sur Internet, en faisant attention à la licence : prends uniquement des images sous licence CC (Creative Commons) et cite l'auteur ainsi que le nom de l'image dans les remerciements.

Il existe de nombreux sites d'images libres (des "banques d'images"). Par exemple :

- http://opengameart.org/art-search-advanced?keys=&field_art_type_tid[]=9 (on peut aussi trouver des sons sur ce site)
- http://www.gameart2d.com/freebies.html
- http://kenney.nl/assets
- http://www.dumbmanex.com/bynd_freestuff.html

- http://blogoscoped.com/archive/2006-08-08-n51.html
- etc.

Souvent, on télécharge des archives (.zip) depuis ces sites. Dans ces cas là, il faut les “extraire” : après avoir téléchargé une archive, fais un clic-droit dessus puis sélectionne “extraire tout” :



J’ai une “Sprite Sheet” comment je l’applique à mon personnage ?

- En tout premier lieu il va vous falloir une animation “Idle” : c’est l’animation qui sera jouée quand notre personnage ne fera aucune autre action.

- Ensuite il nous faudra des action tel que “Jump” pour sauter, “Move” pour avancer, etc
- Commencez par importer votre “Sprite Sheet” dans Unity

- Sélectionnez votre “Sprite Sheet” et allez dans l’inspecteur.

Par défaut votre “Texture Type” est à “Texture” et non à “Sprite (2D and UI)”, pensez à le changer !

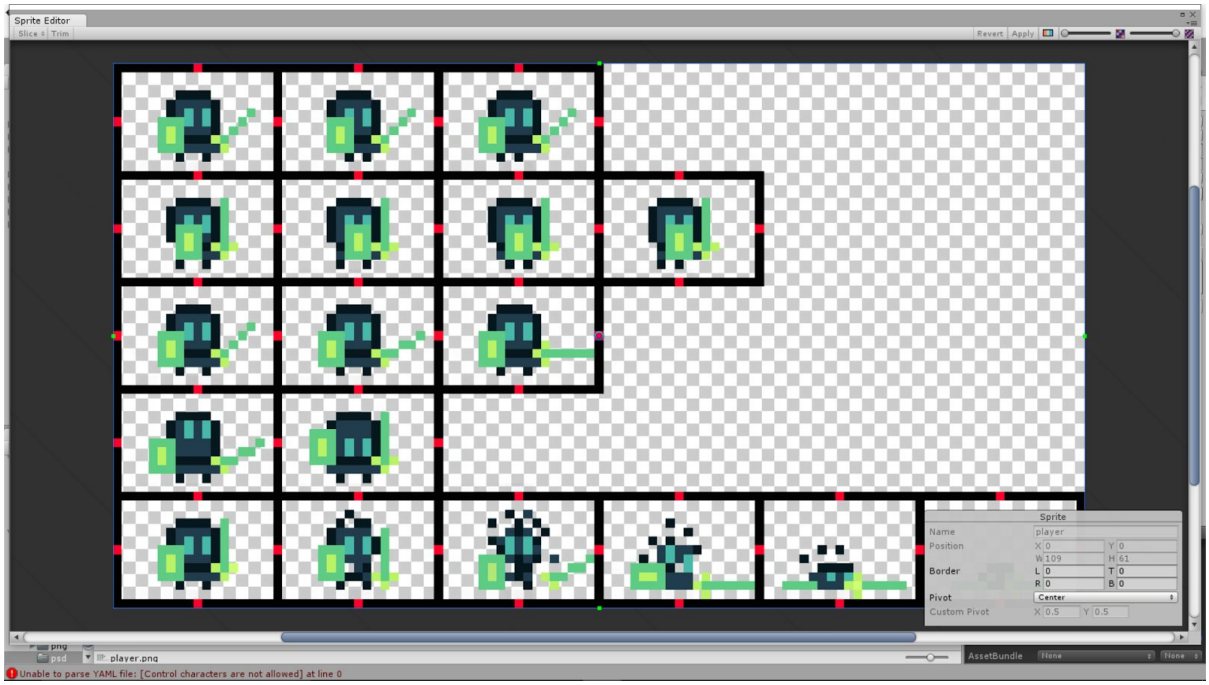
Le “Sprite Mode” est par défaut à “Single” : votre personnage n’aura qu’une seule image. Il faut penser à changer “Single” pour “Multiple” sur votre “Sprite Sheet” sinon vous ne pourrez pas avoir d’animations.

Cliquez sur

/!\ Si vous faites du Pixel Art, décochez **“Generate Mip Map”** passez le **“Filter Mode”** à **“Point (no filter)”** et mettez le “Format” à **“Truecolor”**.

Maintenant que les préparatifs sont terminés nous allons découper cette “Sprite Sheet”

- Cliquez sur



Vous voilà dans l’éditeur de découpe d’Unity. Nous allons sélectionner les différentes zones correspondants à nos images pour isoler les morceaux de notre future animation.

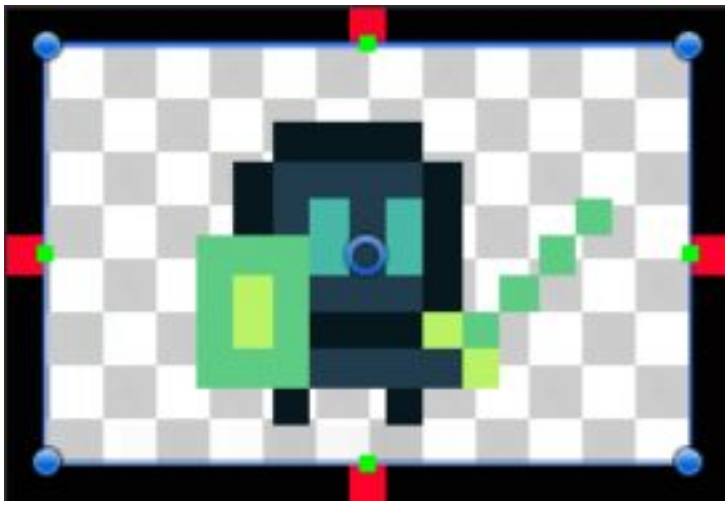
Tips: Lors du dessin de ma Sprite Sheet j’ai créé des cadres noirs avec un centre rouge pour faciliter la découpe dans Unity.

Pour découper une zone il suffit de cliquer sur le coin en haut a gauche de la partie à découper puis d’étirer le rectangle jusqu’au coin en bas a droite.

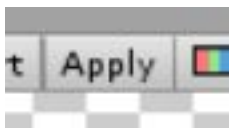
/!\ toutes vos découpes doivent être de la **même taille** et le cercle (ici entre les yeux), qui est le **point de pivot**, doit être au même endroit pour **toutes les découpes**. Cela évite les décalages lors des rotations.

Avant de passer à la suite vérifiez ces points :

- Ma Sprite Sheet est découpée en morceaux de même taille, et avec un point de pivot placé logiquement par rapport au dessin.



- Pas de découpes qui se superposent.
- Pas de découpes inutiles qui traînent.

Quand tout est bon cliquez sur 

et c’est fait !

3 - Anime tes personnages

Objectif

Ton univers est maintenant peuplé d'une multitude de personnages, grands ou petits, gentils ou méchant, courageux, peureux, violet fluo ou vert pâle, …. Il est maintenant venu l'heure de leur donner vie avec des animations !!

Ce que tu vas apprendre

- Créer une animation
- Faire des transitions entre tes animations
- Décider quand ces animations se déclenchent

Méthodologie

Pour commencer, regarde le [Tuto Animations] pour découvrir comment créer une animation sur Unity. Prends bien le temps de construire tes animations avant de passer à la suite !! Il est possible que les personnages que tu as pu télécharger sur l'Asset Store possèdent déjà leurs animations, vérifie bien avant de te lancer dans la démarche de faire les tiennes !!

Quand tes animations sont formées, tu trouveras dans le [Tuto Animator] le moyen de créer les liens de transitions entre elles. Par exemple : de l'animation de ton personnage qui marche tu peux passer à l'animation de ton personnage qui tombe ou à celle de ton personnage qui saute….

Si tu as le courage de fouiller le déclenchement de tes animations à partir du code C#, nous te conseillons d'aller lire le [Tuto Les Scripts] et [Tuto Les Inputs en C#]. Une fois que tu es à l'aise avec ces aspects là, va voir le [Tuto Animations et C#] pour décider quelles actions de ton jeu vont déclencher tes animations.

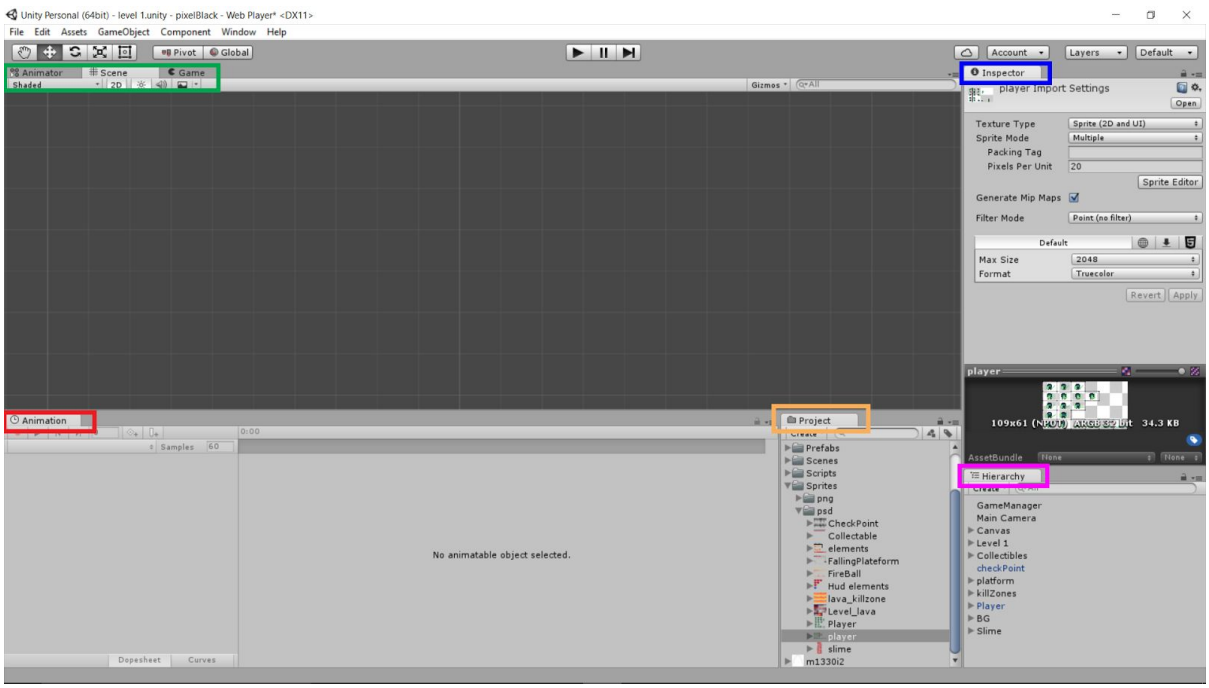
3.1 - Les Animations

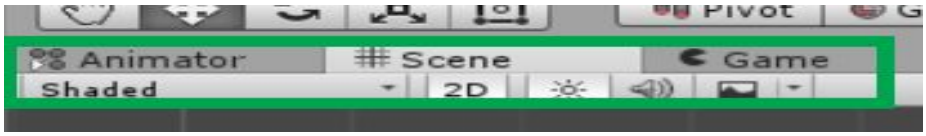
- Pré-Requis: Tutoriel sur les Sprite Sheet

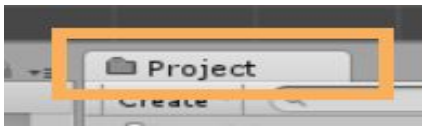
  Commençons par arranger notre interface (il suffit de glisser-déposer les éléments) :

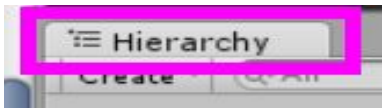
- Dans “Window” ouvrez :

- Puis ouvrez :







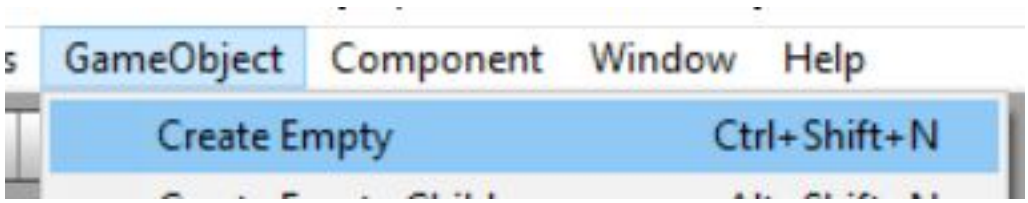


<u>Pour animer notre personnage nous allons procéder en 3 étapes:</u>

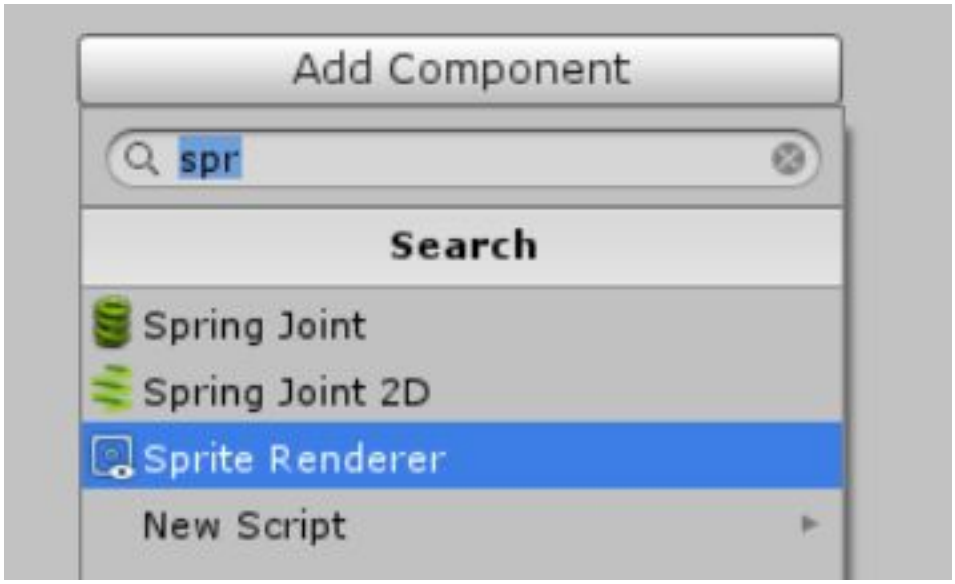
1. la création des animations.
2. la gestion des transitions entre les animations.
3. le déclenchement par code des animations.

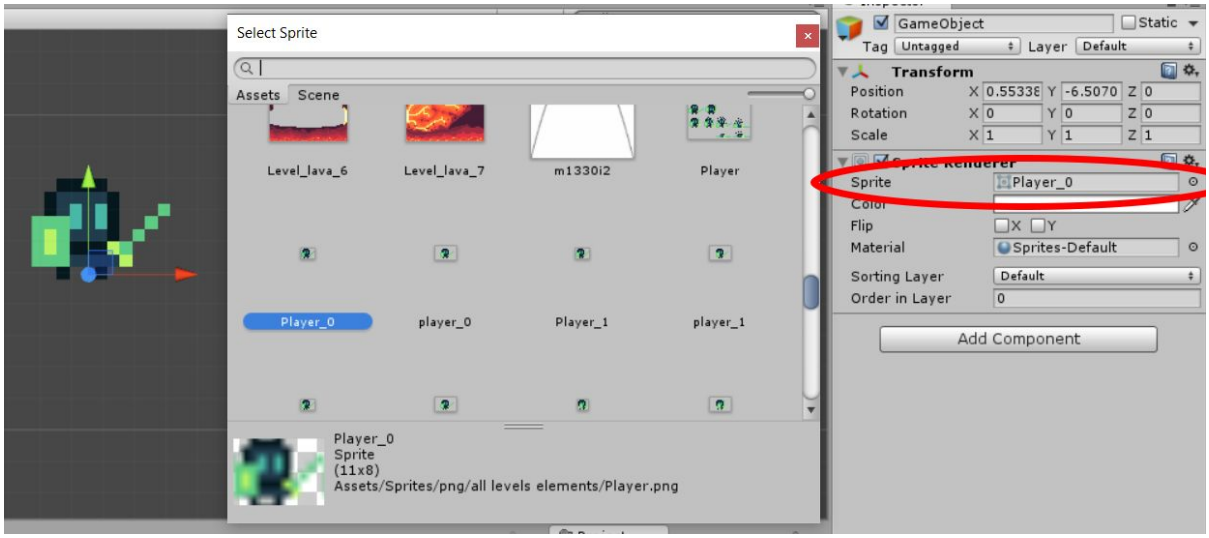
Comment créer mon animation?

Tout d'abord commençons par créer un nouveau "GameObject" vide :



Ajoutons lui un sprite de base (une image). On va prendre la première image de notre SpriteSheet qu'on a créé avant.





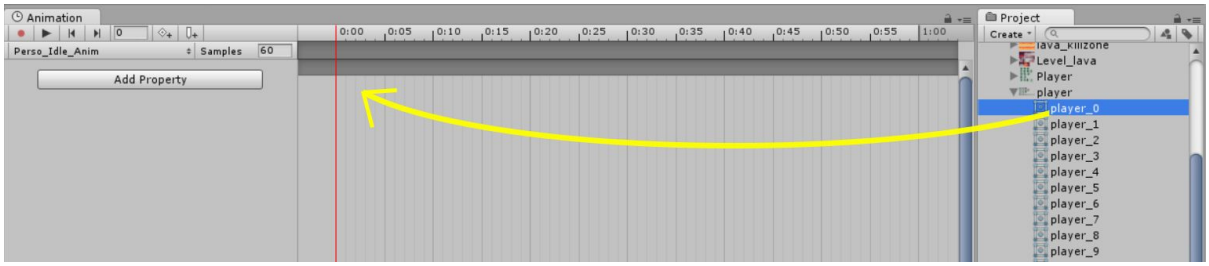
Commençons par créer la plus simple des animations dans le cas présent: Idle (ou quand le perso bouge pas).

En ayant le début de personnage sélectionné cliquez dans “Animation” sur :

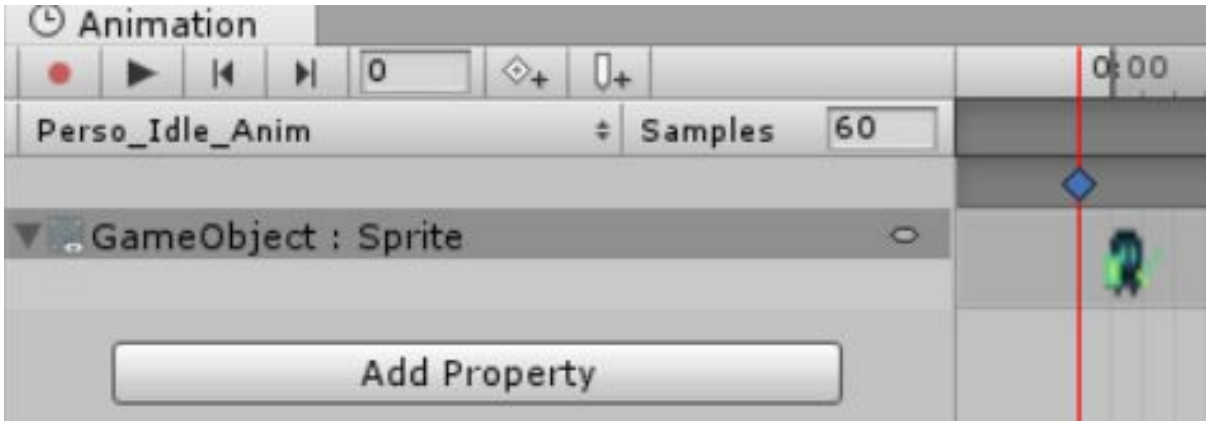
Enregistrez sous le nom Perso_Idle_Anim dans le dossier Animation de votre projet.

Sur la “time line”(frise du temps) qui s’affiche vous allez disposer les sprites dans l’ordre dans lequel vous voulez qu’ils s’affichent.

Ici on veut que notre personnage ne bouge pas. On va donc lui mettre une seule image celle de sa pose Idle (player_0 pour moi).

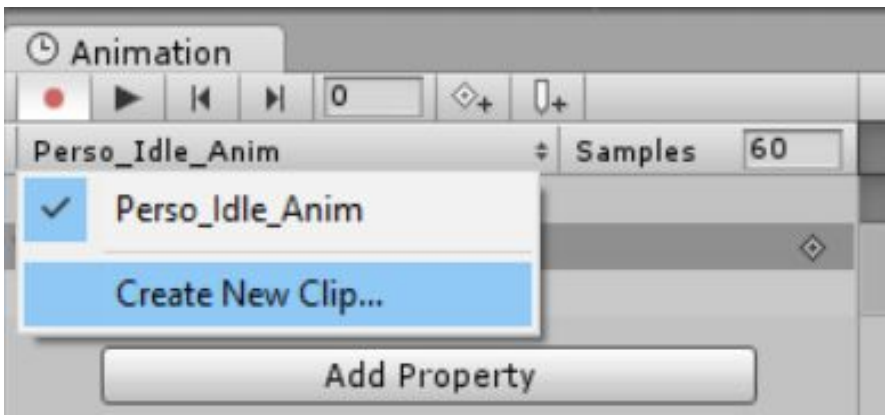


Ce qui donne ceci :

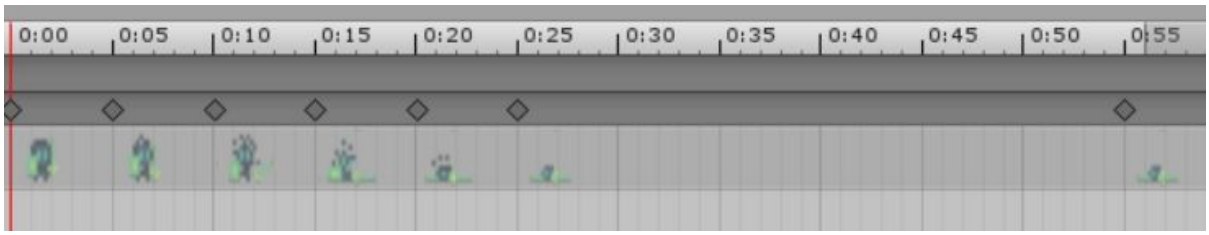


Pour les autres animations le début est le même, on ne changera que le nombre, l'ordre et le temps entre les différentes images de l'animation.

Pour créer une nouvelle animation :



- Par exemple, l'animation de mort :

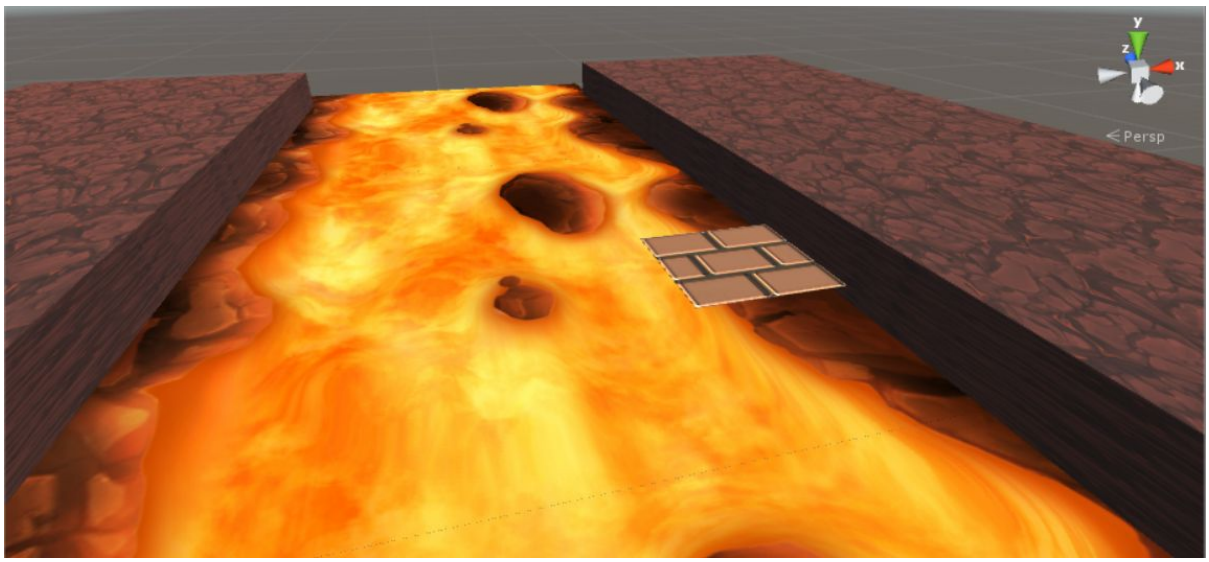


Voici la liste des animation pour ce personnage.

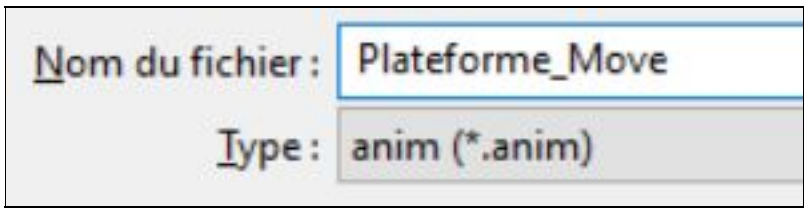
- Ne rien faire ("idle")
- Marche ("walk")
- Saut ("jump")
- Tomber ("fall")
- Lever le bouclier ("shield")
- Marcher en levant le bouclier ("walk_shield")
- Attaque ("attack")
- Mort ("death")

Si vous êtes en 3D et que vous ne voulez pas changer de texture

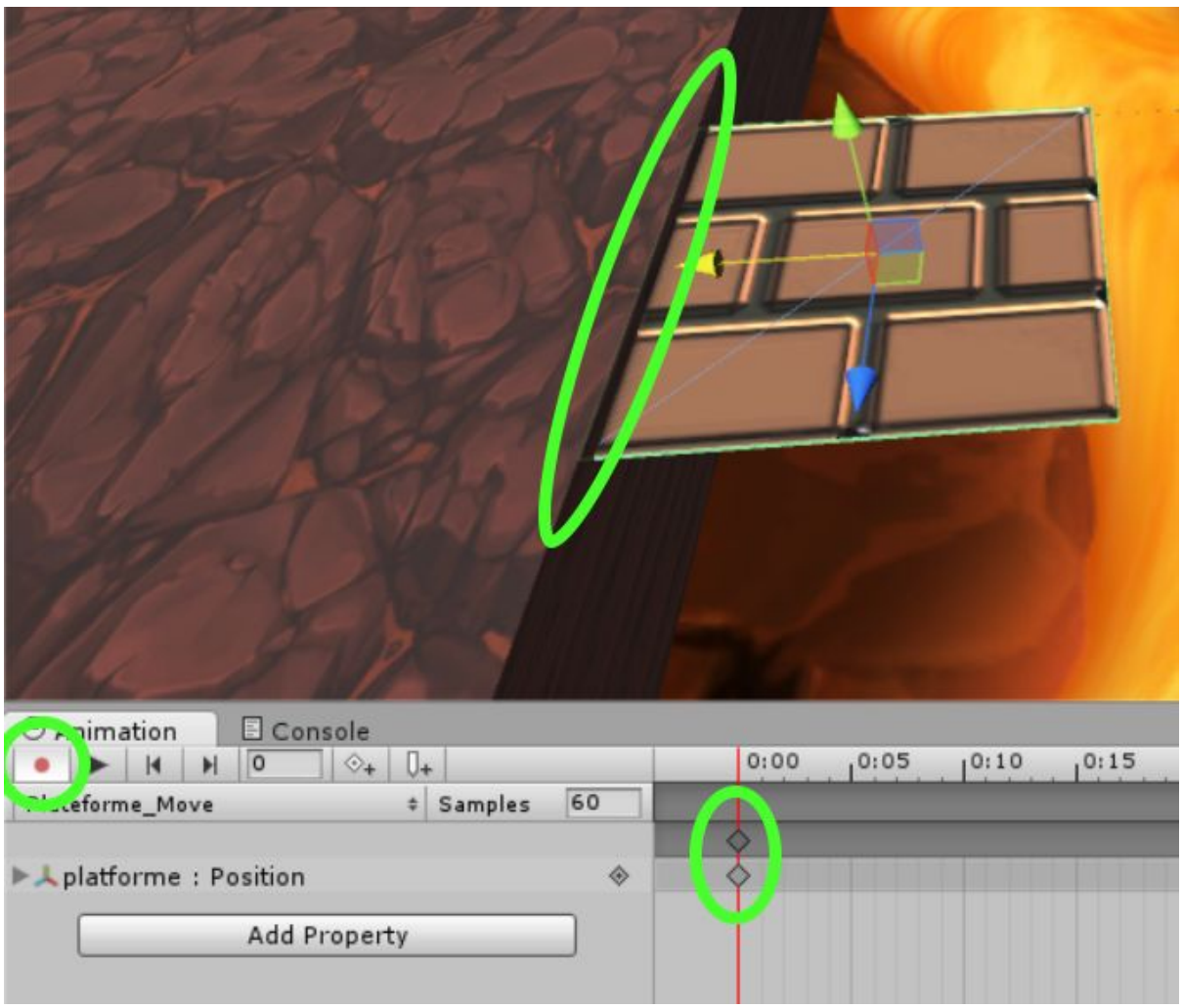
Vous n'êtes pas obligés de changer de texture à chaque étape de l'animation, il est aussi possible de simplement changer la position ou la taille.
Par exemple, si je veux bouger cette plateforme au dessus de la rivière de lave :



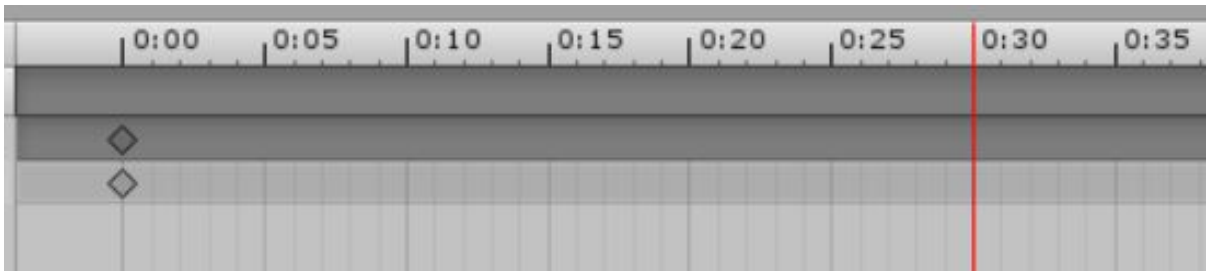
Commençons par créer une animation :



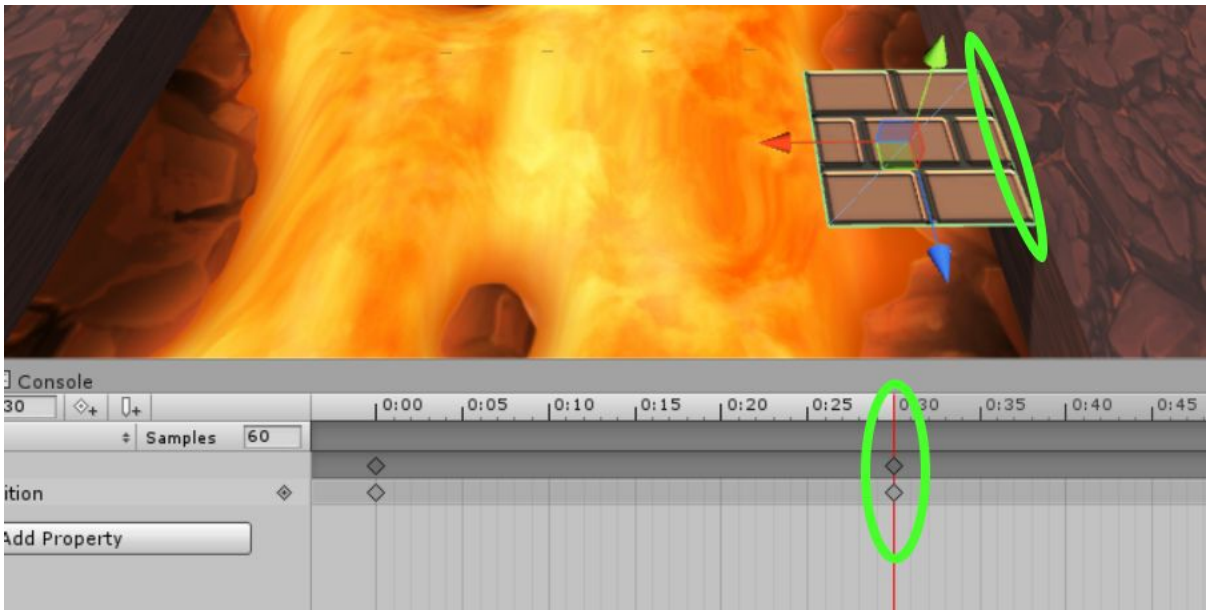
1. On vérifie dans le bloc animation que le point rouge est allumé (ce qui veut dire que tout ce que l'on fait sera enregistré)
2. On colle la plateforme contre le bord cela à pour effet de créer une première "key frame" de position. C'est la première position de notre plateforme.



3. On positionne la barre rouge un peu plus loin dans la "Time-Line" de l'animation



4.
On déplace la plateforme jusqu'à l'autre rive pour créer notre deuxième "key-frame" de position.



5. On décale une dernière fois la barre rouge d'une distance égale à celle de l'étape précédente (sinon notre plateforme passera plus de temps à revenir de la position d'arrivée à la position de départ que de temps à aller du départ à l'arrivée)., puis on fait revenir la plateforme à l'exacte position qu'elle avait au départ.
Pour cela cliquez sur la première "key-frame", notez les coordonnées dans le Transform de la plateforme puis revenez sur votre dernière position et remplacez les valeurs par celles que vous venez de noter.

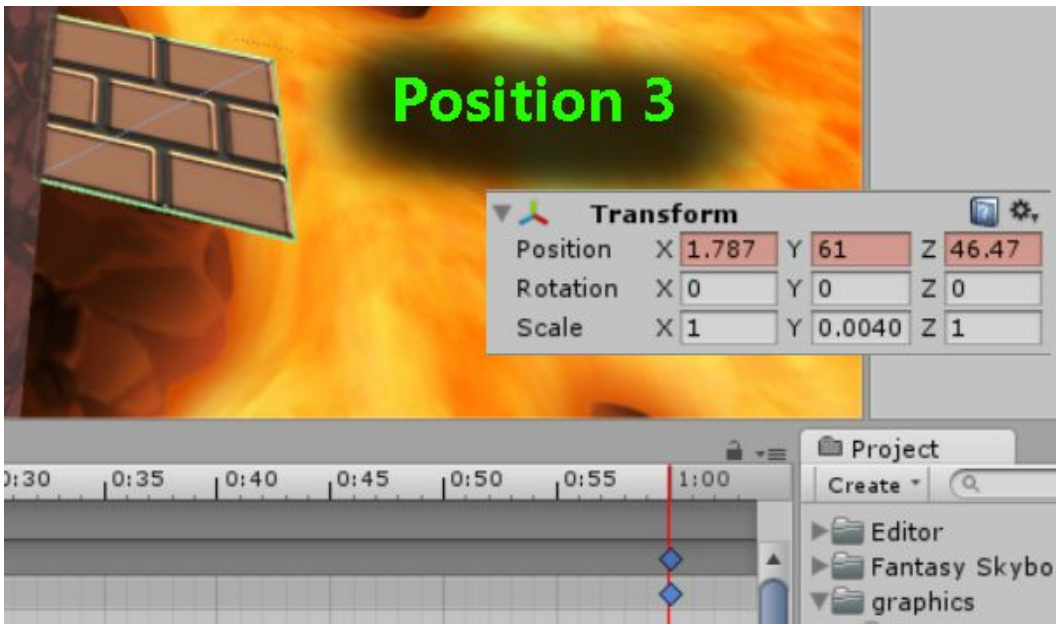
Quand vous avez vos 3 positions:

la première correspondant au “Point de départ”

la seconde correspondant au “point d’arrivée”

la dernière correspondant au “Point de départ”

Vous pouvez quitter le mode



animation en cliquant sur le point rouge 

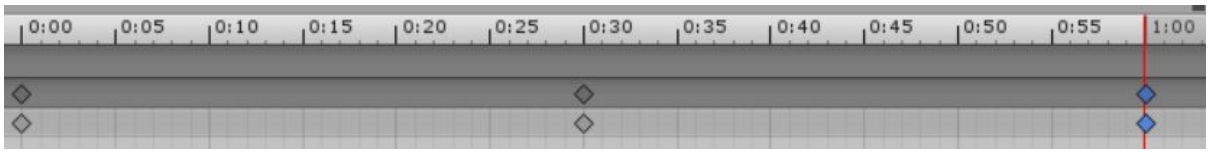
Appuyez sur PLAY pour voir votre animation.

Mon animation est beaucoup trop rapide !!!

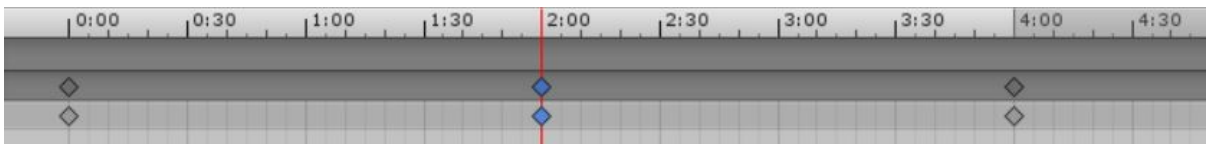
La “time-line” par défaut étant de 1 seconde (60 frames) votre animation peut être trop rapide pour le rendu que vous souhaitiez.

Sur la “Time-line” en maintenant ctrl et en manipulant la molette vous pouvez zoomer ou dézoomer pour accélérer / ralentir votre animation. Vous pouvez ensuite déplacer vos “key-frames” pour modifier le temps de votre animation.

AVANT: 1 seconde d’animation



APRES: 4 secondes d’animation

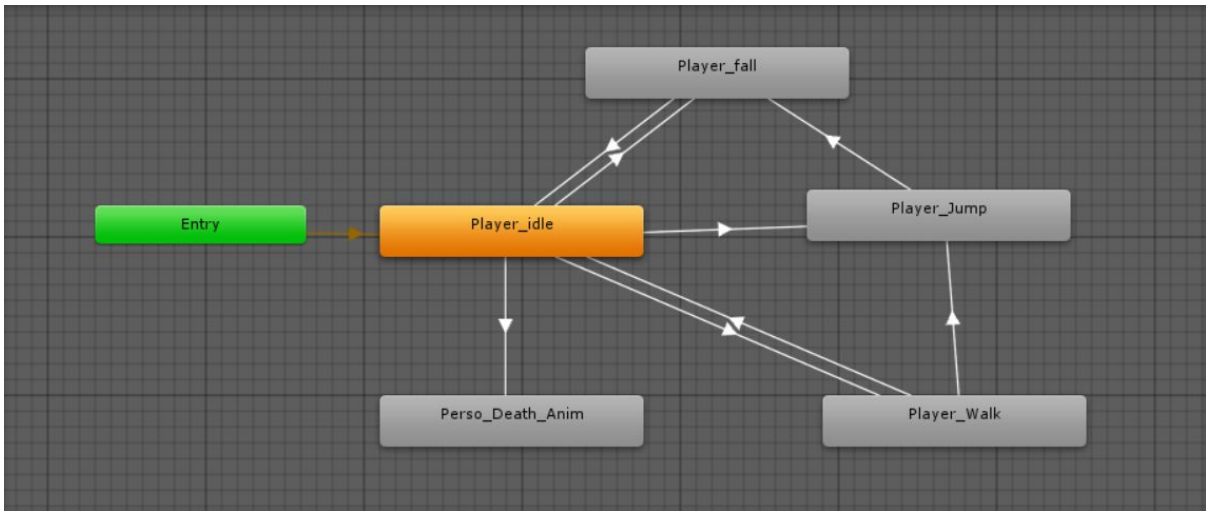


3.2 - Les Animator

- Pré-Requis: Tutoriel Animations

Comment je gère les transitions entre mes Animations ?

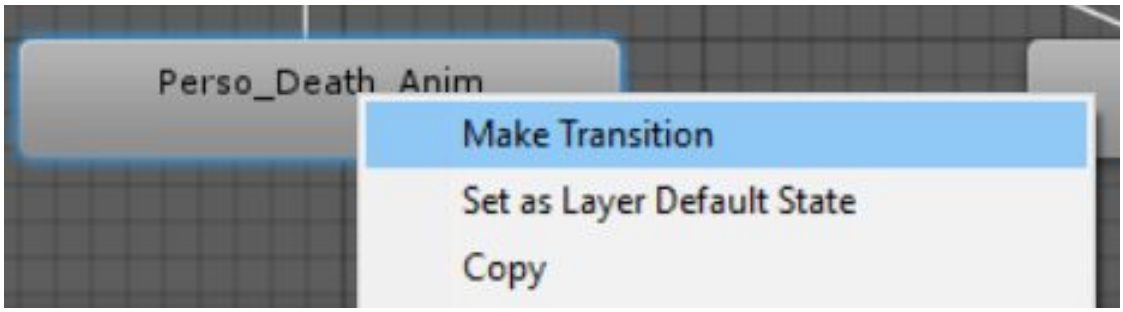
On va utiliser “l’Animator”, dont voici l’interface, avec le schéma des différentes animations :



La première animation que vous avez créé (“Idle”) sera reliée au block vert “Entry” par une flèche orange : c’est la première animation qui sera jouée lors de l’apparition de notre Player.

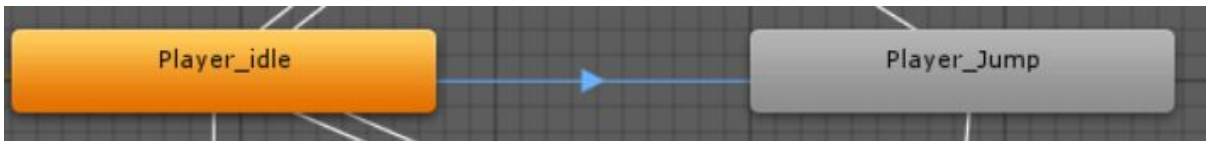
Pour ce qui est des autres animations, elles ne sont pas (encore) reliées entre elles. Chaque flèche correspond à une transition. Par exemple depuis Idle on peut passer à Walk, Jump, Fall, Death. Mais on ne peut pas passer de Jump à Idle (il faut d’abord passer par “Fall” quand on saute).

Pour créer un lien entre deux animations effectuez un clic droit sur l’origine puis cliquez sur l’arrivée.

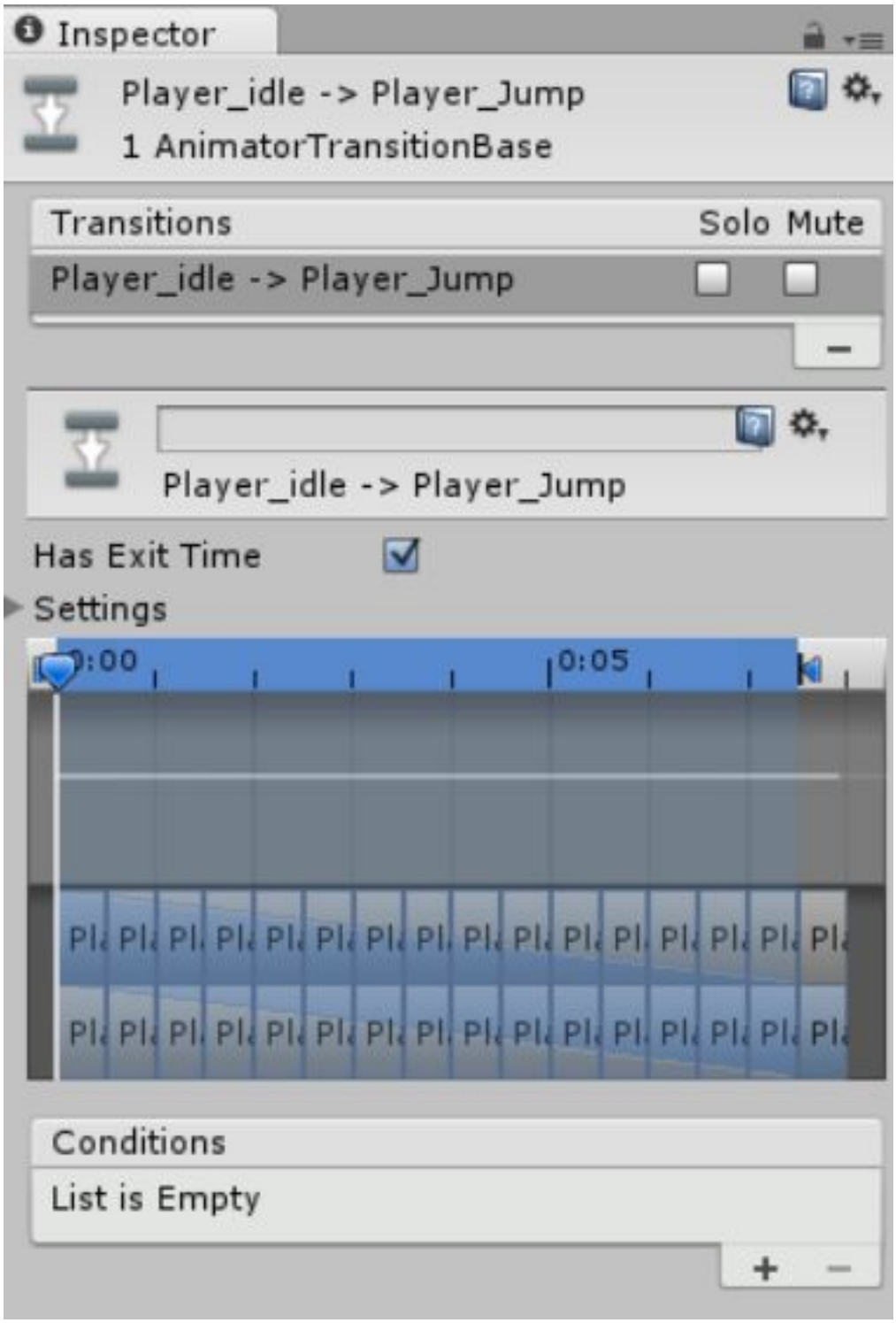


Si on veut changer le “Default State”, l’animation par défaut qui sera joué au début du personnage (reliée par la flèche orange), il suffit de faire clic droit > “Set as Layer Default State”.

Il faut maintenant ajouter un peu de logique car sinon il est impossible d’utiliser nos transitions (et donc de changer d’animation).
Cliquez sur la flèche de transition :



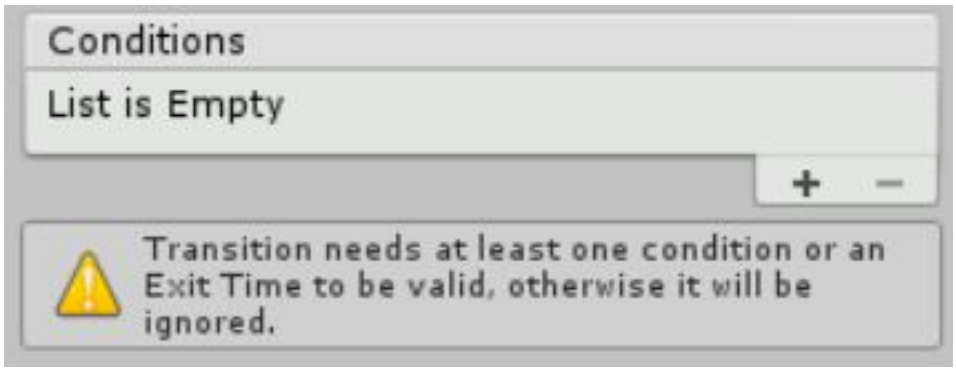
Dans l’inspecteur les détails de cette transition vont s’afficher:



Premier paramètre “Has Exit Time” permet de choisir si la transition peut se faire à n’importe quel moment (tant que le déclencheur est activé) ou si la transition ne peut se faire qu’à la fin de l’animation précédente.

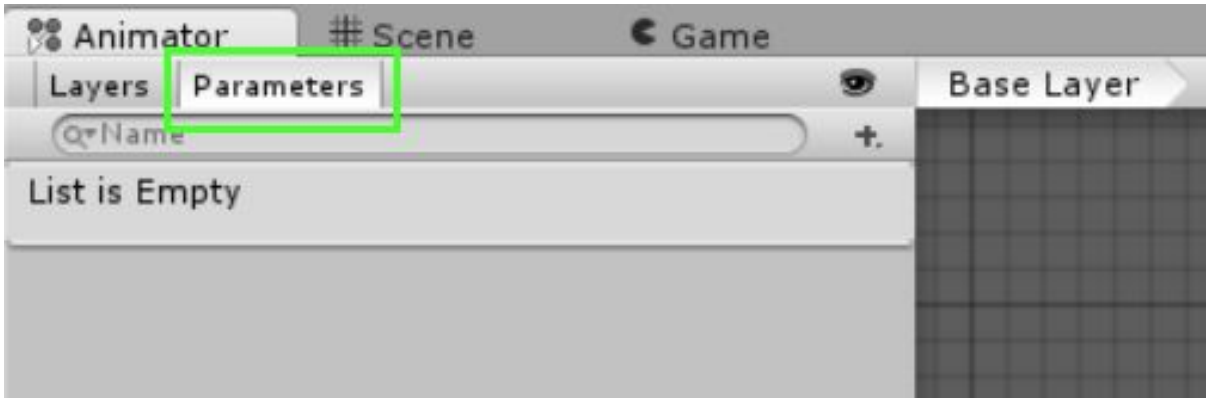
Je décide de le décocher car mes transition de saut, de chute et de course ne doivent pas bloquer la fluidité du gameplay de mon jeu.

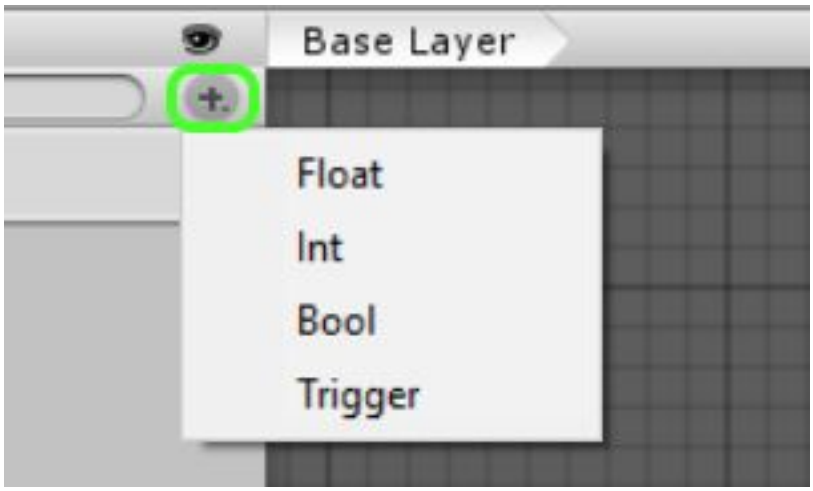
En décochant ce paramètre vous verrez un petit message d’alerte plus bas :



En effet notre transition n’a plus aucune condition pour s'effectuer. Pour le moment on ne lui a pas attribué d'évènements et on vient de lui retirer “Has exit time”.

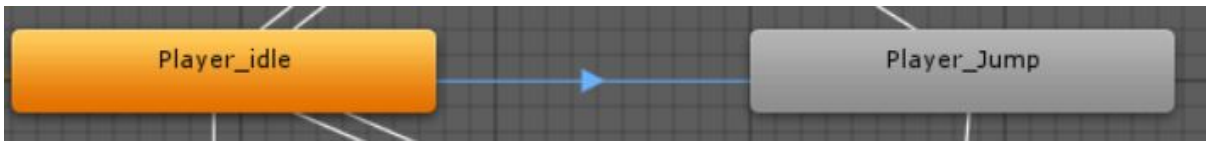
Nous allons donc rajouter des paramètres (qui nous permettrons de déclencher nos animations)



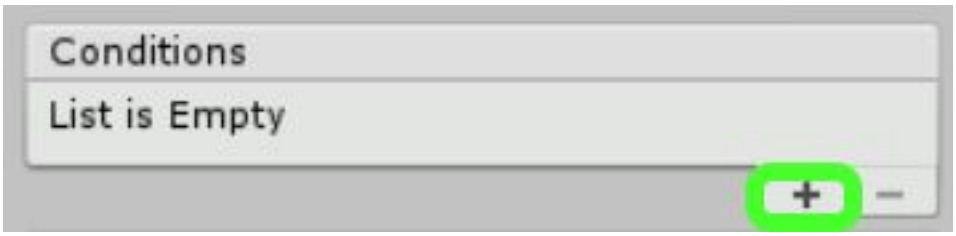


Ces paramètres seront modifiés par votre code. Par exemple vous pouvez ajouter un booléen (type de variable prenant pour valeur uniquement vrai ou faux) pour demander si vous êtes en train d'avancer, ou bien ajouter un Float pour vérifier si votre mouvement (souvent appelé vélocité) sur l'axe Y est positif (je saute) ou négatif (je tombe).
Créez par exemple un booléen "isJumping" qui prendra "true" si on saute et "false" si on saute pas.

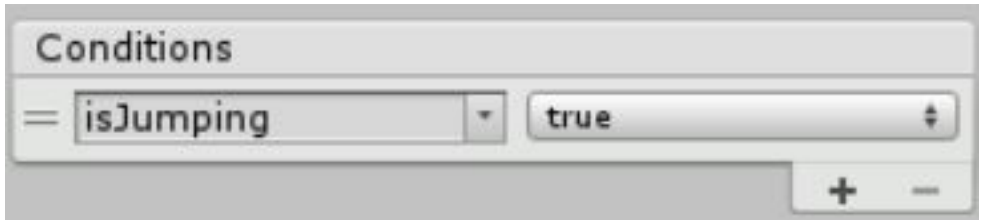
Retournons sur la flèche transitionnelle entre "Idle" et "Jump" :



Nous pouvons désormais rajouter une condition d'exécution :



Bien évidement la condition naturelle pour cette action est :



Certaines transitions demanderont plus de conditions et de paramètres que celle ci. Par exemple si je me protège avec mon bouclier et que je décide de taper en même temps, si j'ai une animation qui correspond a cette action je vais devoir la déclencher uniquement si les deux conditions (se protéger et attaquer) sont vérifiées en même temps

3.2 - Les scripts dans unity

Jusqu'à maintenant nous avons utilisé l'éditeur graphique d'Unity. Même si celui-ci est très puissant, il a ses limites et nous allons donc voir comment créer des scripts, indispensable dans n'importe quel projet.

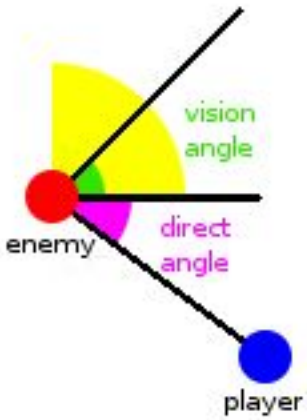
Quel langage est utilisé dans unity ?

En faisant un clic droit dans votre projet et en sélectionnant "Create" vous aurez la possibilité de choisir entre deux langages : 

Le plus commun et le plus documenté sur internet est le C#. Nous allons donc nous concentrer exclusivement sur ce langage lors de ce tutoriel.
Note : on prononce "C#" "C-sharp" et non "C-dièse", ne soyez pas étonnés si vous l'entendez dans une vidéo !

À quoi sert un script ?

Un script permet de donner des instructions précise à un objet en utilisant plusieurs paramètres. Par exemple un script pour les déplacements du personnage ou un script pour déplacer la caméra en fonction des déplacements du joueur.



Quelques règles importantes

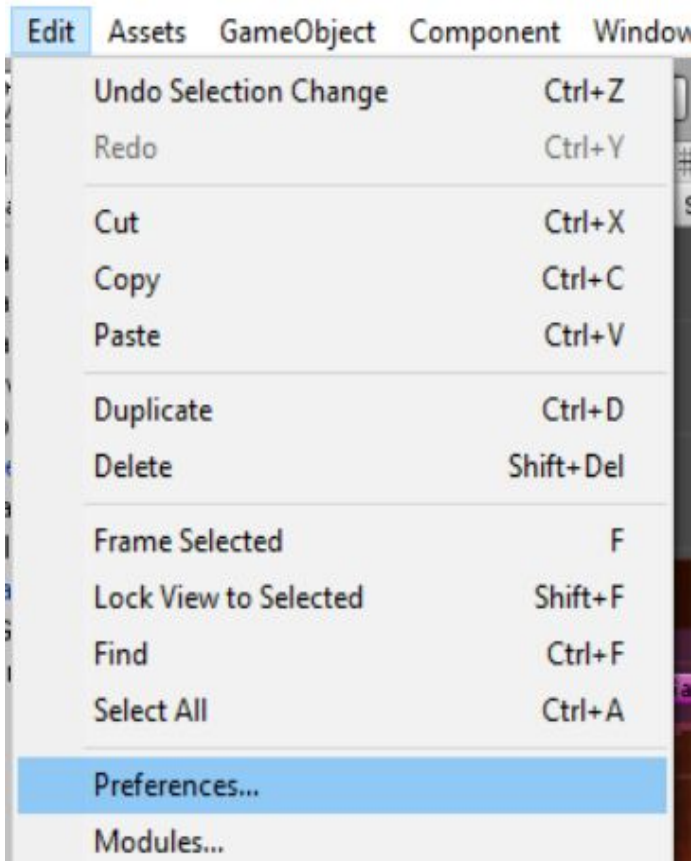
Il est important d'organiser un minimum nos scripts, car on peut facilement s'y perdre (essayez de retrouver une feuille dans une boîte remplie de feuilles …). On va donc séparer les scripts et créer un script par objet de notre code. On aura donc un script pour le joueur, un script pour le bonus (une pièce par exemple), un script pour chaque type d'ennemi.

À l'intérieur même de nos scripts, il est important de rester clair, pour cela quelques règles de base :

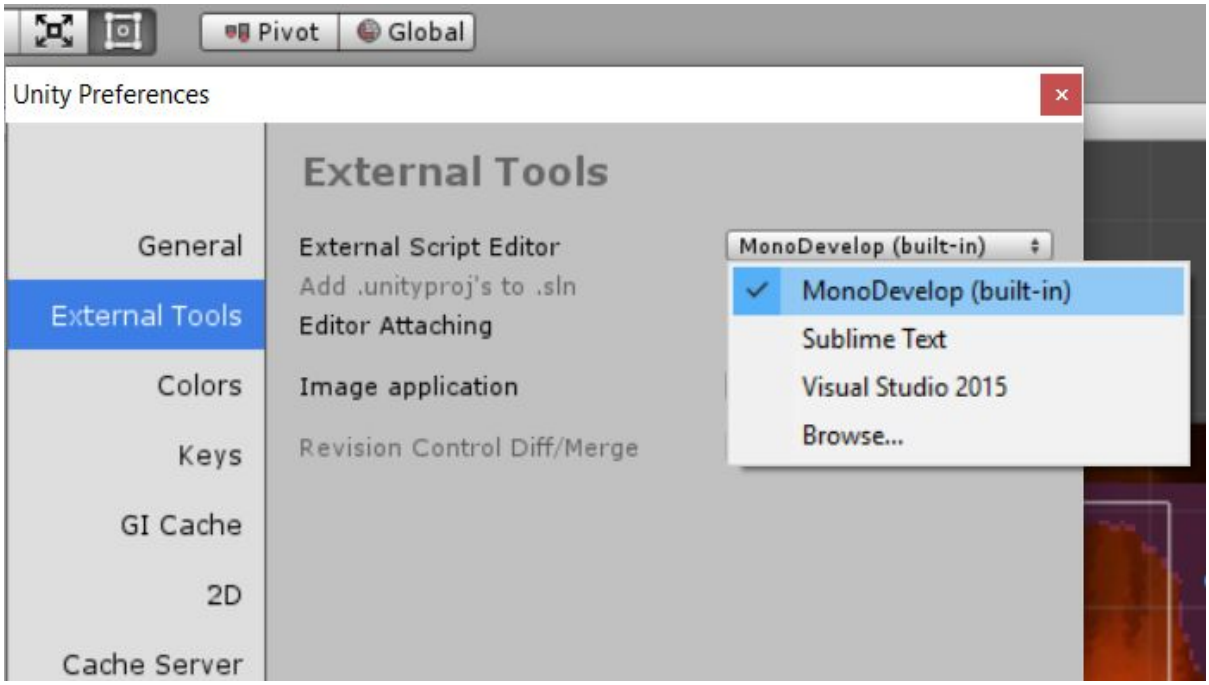
- Les noms de variables doivent être clairs. Pas de "a", "b" et "c", mais plutôt "pointsDeVie", "attaque", "maCamera" …
- Rajoutez des commentaires ! Ils seront très importants quand vous relirez votre code.
- Sautez régulièrement des lignes pour "aérer" votre code

Démarrage sur Unity

Il est possible d'utiliser beaucoup d'éditeurs différents avec Unity. Nous allons utiliser "MonoDevelop" (un éditeur présent avec Unity)". Allez dans :



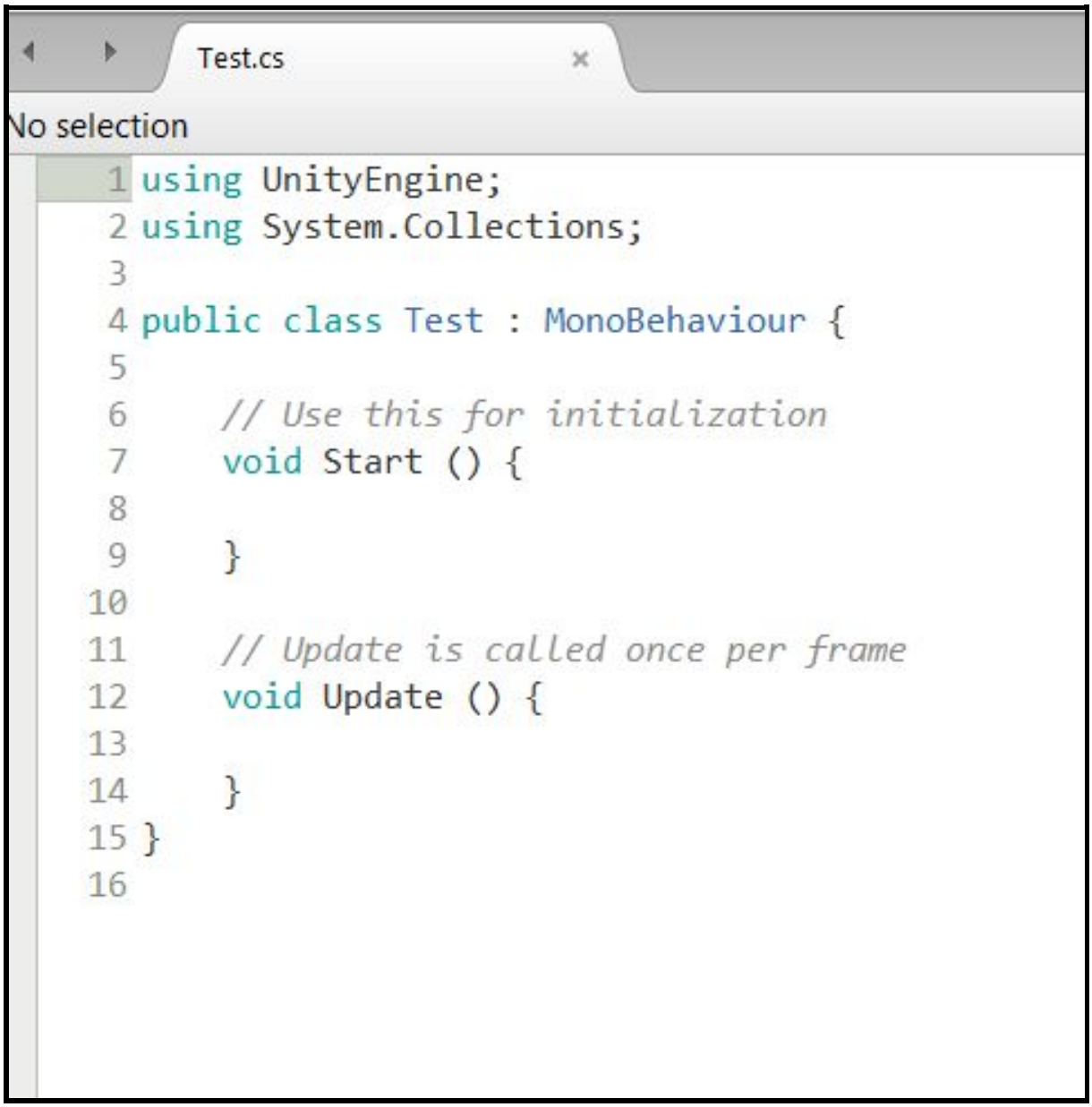
Et choisissez MonoDevelop comme "External Script Editor"



Créez votre premier script C# intitulé “Test” (clic droit dans la partie project, et “create C# script”)

(On met toujours une majuscule au nom des scripts).

Ouvrez le par un double-clic.



Expliquons dans l'ordre les différents éléments présent dans le fichier de base.

Lignes 1 et 2 Permet d'importer des "packages" de code ; un peu comme les "packages" d'assets, il y en a certains présents dans Unity par défaut, et on peut en récupérer sur Internet si besoin.

Ligne 4 Il faut remettre ici le nom du fichier (toujours avec une majuscule, donc Test) ; la syntaxe (manière d'écrire les choses) est importante à respecter, sinon Unity ne comprendra pas.

Ligne 6 et 11 ce sont des commentaires, mettre un // dans une ligne permet à Unity d'ignorer ce qu'il y a après. C'est très pratique pour indiquer à quoi sert telle ligne ou bout de code !

Ligne 7, il s'agit de la fonction Start ;elle est très utile car c'est ici qu'on initialise les valeurs de nos variables.

Ligne 12, une autre fonction très utile, la fonction Update. Elle est appelée une fois par image("frame" en anglais). C'est là que se passera la majorité de notre code (les évènements qui se passeront en boucle, comme les déplacements).

Les variables

Avant de rentrer vraiment dans le code, il est important de comprendre ce que c'est qu'une variable. Une variable est une boîte qui peut contenir une valeur. La valeur peut être un nombre entier(54), un nombre à virgule (12,34), un booléen (valeur vrai ou faux), un texte ("coucou"), ou encore des objets Unity (un gameObject par exemple).

Comme on peut le constater, chaque variable a aussi un type : ce que la boîte contient. En C#, il est indispensable de préciser le type d'une variable avant de l'utiliser, et on ne peut pas changer son type : si la variable "pointDeVie" est un nombre entier et contient 50, elle ne peut pas devenir du texte et contenir "coucou".

Il existe de nombreux types de variables, et nous n'allons pas tout utiliser. Regardons d'abord comment utiliser les types dit "de base" (qu'on peut retrouver dans à peu près tous les langages de programmation) :

Important : On met toujours une minuscule au début des nom de variables et une majuscule pour séparer les mots qui la composent.

```
public int monEntier; //pour les nombres entiers positifs ou négatifs

public float monFloattant; //pour les nombres à virgules positifs ou négatifs

public bool monBooléen; //pour les booléens (true = vrai / false = faux)

public string monTexte; //stock un texte

// Use this for initialization
void Start () {
    monEntier = -12;
    monFloattant = 12.5f; /*on met f pour faire comprendre
                            à l'ordinateur qu'il s'agit bien d'un float */
    monBooléen = true;
    monTexte = "hello world";
}
```

Nous avons aussi des types propres à Unity. Tous les composants d'un objet qu'on peut trouver dans l'inspecteur ont un type en C#.

- Déclaration:

```
public GameObject monGameObject; // permet de stocker l'ensemble des propriétés d'un objet.

public Transform monTransform; // permet de stocker le composant Transform d'un gameObject

public Animator monAnimator; // stock l'Animator d'un Objet => permet de gérer les animations

public Vector3 monVector3; // stock 3 valeurs sous la forme d'une position dans l'espace x,y,z

public Vector2 monVector2; // stock 2 valeurs sous la forme d'une position dans un plan x,y

public Camera maCamera; // stock un Objet de type Camera
```

- Initialisation:

```
monGameObject = new GameObject (); //créé un nouveau gameObject
monGameObject = gameObject; //gameObject permet d'accéder au gameObject sur lequel est ce script

monTransform = transform; //transform permet d'accéder au Transform du gameObject du script

monAnimator = GetComponent<Animator> (); // récupère l'Animator du gameObject

monVector3 = transform.position; //récupère la position de l'objet

monVector2 = new Vector2 (transform.position.x, transform.position.y); //récupère la position 2D de l'objet

maCamera = Camera.main; // récupère la mainCamera de ta scène
maCamera = new Camera (); // créé une nouvelle Camera
```

Les fonctions

On utilise des fonctions pour interagir avec Unity, que ça soit pour lui donner des informations (changer l'animation d'un personnage), ou pour en récupérer (savoir si l'utilisateur a appuyé sur la touche droite par exemple). Start() et Update() qu'on a vu un peu plus tôt sont aussi des fonctions ; de manière générale une fonction c'est juste plusieurs lignes de code.

Une fonction peut éventuellement renvoyer une valeur : c'est le cas quand on demande à Unity si l'utilisateur a appuyé sur la touche droite. Certaines fonctions n'en renvoient pas, par exemple, changer l'animation d'un personnage. Cette valeur est comme une variable, et elle a donc un type.

Une fonction prend (quasiment) tout le temps des arguments. Les arguments sont des variables qu'on va donner à la fonction pour qu'elle puisse fonctionner. Par exemple, à une fonction "bouger", on va lui donner le nombre de pas dont on doit avancer.

Un exemple d'utilisation de fonction, avec et sans retour de valeur :

```
Debug.Log ("l'Imagine cup junior c'est super !"); // Cette fonction va afficher dans la console
                                                  //  le texte que tu lui as passé en paramètre

GameObject maCamera = GameObject.Find ("Camera"); // Cette fonction cherche dans ta hiérarchie le nom du GameObject que
                                                  // tu lui as passé en paramètre, puis te retourne ce GameObject.
```

3.4 - Inputs C#

- Pré-Requis: Tutoriel sur les Scripts
  On va gérer comment interagir avec l'utilisateur, c'est à dire savoir si le joueur a appuyé sur une touche de son clavier ou l'écran tactile d'une tablette ou déplacé la souris.

  On va procéder de la même manière pour tout : on va utiliser des fonctions qui vont nous renvoyer un booléen : vrai si l'utilisateur est en train d'interagir avec nous d'une certaine manière. On va tester directement le résultat avec un "if" (si en anglais).

  Le clavier

  On a ici trois fonctions, qui prennent toutes un paramètre, la touche qu'on veut tester. Les plus utilisées sont Space (touche espace), Enter (entrée), A-Z (touches des lettres), RightArrow/LeftArrow/DownArrow/UpArrow (flèche droite - gauche - bas - haut). Tu peux trouver une liste complète des touches en anglais ici.

```
KeyCode.A;
KeyCode.B; //etc
KeyCode.Space;
/*
  les keycode correspondent à toutes les
  touches clavier
*/
```

```
void Update () {

    if (Input.GetKey (KeyCode.RightArrow))
    {
        //le code sera exécuté à chaque fois
        //qu'on appuie sur la flèche en question
    }
    if (Input.GetKeyDown (KeyCode.RightArrow))
    {
        //le code sera exécuté au moment
        //où on appuie sur la flèche en question
    }
    if (Input.GetKeyUp (KeyCode.RightArrow))
    {
        //le code sera exécuté au moment
        //où on relâche la flèche en question.
    }
}
```

La souris

Cela fonctionne sensiblement de la même manière et on a les mêmes fonctions. Elles prennent un paramètre, le bouton de la souris qu'on veut tester. 0 correspond au bouton gauche, 1 au bouton droit, 2 à la molette.

```
void Update() {

    if (Input.GetMouseButton (0))
    {
        //le code sera exécuté à chaque fois
        //qu'on appuie sur le bouton en question
    }

    if (Input.GetMouseButtonDown (0))
    {
        //le code sera exécuté au moment
        //où on appuie sur le bouton  en question
    }

    if (Input.GetMouseButtonUp (0))
    {
        //le code sera exécuté au moment
        //où on relâche le bouton en question
    }
}
```

L'écran tactile (tablette / ordinateur)

Cette fois ci cela fonctionne un peu différemment : vu qu'on peut avoir plusieurs "touch" simultanés (si on appuie avec 2 doigts à un endroit différent), Unity nous renvoie une liste de touch. Pour accéder à chaque touch individuellement, on va utiliser une syntaxe un peu différente avec un foreach :

```
Input.touches; //Liste d'objet de type Touch
```

```
void Update () {
    foreach (Touch touch in Input.touches) {
        /*
            tout code placé ici sera joué pour
            chaque touch sur la surface du portable
            ou de la tablette.
            5 doigts sur une tablette appelera 5 fois
            le code ici.
        */
    }
}
```

On utilise une méthode un peu différente : au lieu d'appeler une fonction pour savoir si on appuie (ou pas), on va tester si la "phase" (l'état) du "touch" est égale à soit :

```
void Update () {

    foreach (Touch touch in Input.touches)
    {
        if (touch.phase == TouchPhase.Began)
        {
            //le code s'exécute tant qu'on touche l'écran (comme GetKey)
        }
        if (touch.phase == TouchPhase.Ended)
        {
            //le code s'exécute une seule fois quand on arrête d'appuyer (comme GetKeyUp)
        }
        if (touch.phase == TouchPhase.Stationary)
        {
            //le code s'exécute tant qu'un doigt est appuyé sur l'écran sans bouger (comme GetKey)
        }
        if (touch.phase == TouchPhase.Moved)
        {
            //le code s'exécute tant qu'un doigt est appuyé sur l'écran en bougeant (comme GetKey)
        }
        if (touch.phase == TouchPhase.Canceled)
        {
            //le code s'exécute quand le touch est annulé, ce qui peut arriver quand il y a
            //trop de doigts en même temps ou le doigt sort de l'écran en bougeant
        }

    }
}
```

Les axes

Les axes nous permettent de savoir où se trouve la souris dans la fenêtre. La fonction GetAxis prend le nom en paramètre et renvoie un nombre (à virgules) entre -1 et 1. Il en existe 2 :

```
void Update () {

    if (Input.GetAxis ("Mouse X"))
    {
        /*renvoie -1 si la souris est à gauche de la fenêtre,
        1 si elle est à droite, et entre -1 et 1 sinon.
        Par exemple, si on est à la moitié du chemin entre le
        milieu (0) et la droite de la fenêtre (1), elle renverra 0.5*/
    }

    if (Input.GetAxis ("Mouse Y"))
    {
        /*renvoie -1 si la souris est en haut de la fenêtre,
        1 si elle est en haut, et entre -1 et 1 sinon.
        Par exemple, si on est à la moitié du chemin entre le
        milieu (0) et le bas de la fenêtre (-1), elle renverra -0.5*/
    }
}
```

3.5 - Les Animations avec du C#

- Pré-Requis: Tutoriels sur Animations, Animator, les Scripts

Je veux lancer l'animation de mon personnage quand je le fais marcher.
Lors du tuto "Animator", nous avons vu comment ajouter des paramètres (bool, int, float, etc…) pour déclencher les animations selon certaines conditions.
Nous allons ici utiliser un paramètres de type bool : enDéplacement, qui vaut vrai si on est en train d'appuyer sur la flèche droite OU gauche, et faux si aucune n'est appuyée.

Voyons comment en C# je peux agir sur ces paramètres.
Pour celà il va nous falloir :

- une variable de type "Animator" pour accéder au composant du même nom.
- un bout de code qui va tester si on appuie sur la touche flèche droite (comme on a vu dans le tuto sur les Input) et change les valeurs du paramètre dans l'Animator.

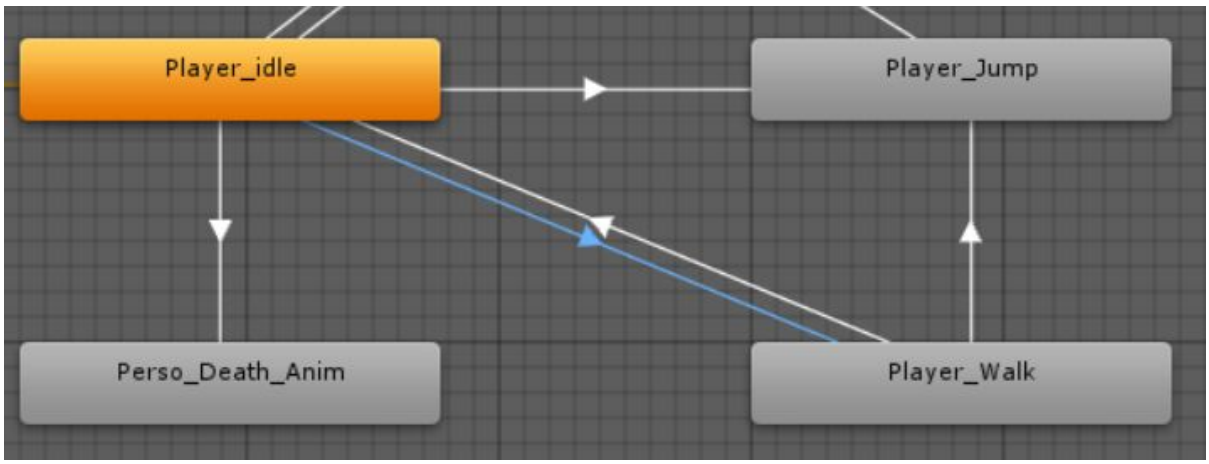
```
public Animator monAnimator;

void Start()
{
    monAnimator = GetComponent<Animator> ();
}

void Update() {

    // On test si la touche droite ou gauche est appuyée
    if (Input.GetKey (KeyCode.RightArrow ) || Input.GetKey (KeyCode.LeftArrow ))
    {
        //mettre la propriété "enDéplacement" de l'animator à vrai
        monAnimator.SetBool("enDéplacement", true);
    }
    else
    {
        monAnimator.SetBool("enDéplacement", false);
    }
}
```

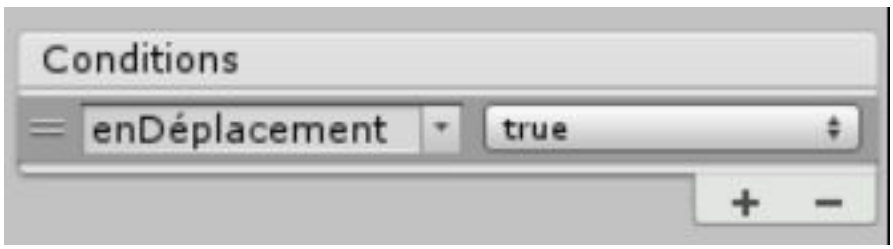
Ici, on utilise deux nouvelles choses :

- Le "||" ; c'est un "OU". Ainsi, on mettra "enDéplacement" à true si on appuie sur la touche droite OU sur la touche gauche. Il existe aussi le "&&" pour le ET (vérifie que les deux conditions sont vraies)

- Le “else” : le code qui suit dans les crochets s’exécute uniquement si la condition du if n’est pas vérifiée. Ici, le code s’exécutera si on n’appuie pas sur la touche gauche et qu’on n’appuie pas sur la touche droite.

Dans l’animator, il suffit de tester si “EnDéplacement” vaut vrai :





Pour chaque type de paramètre il existe un “Set….()”
En voilà trois exemples:

```
anim.SetInteger ("nom du param", valeur);
anim.SetBool ("nom du param", valeur);
anim.SetFloat ("nom du param", valeur);
```