# SOMMAIRE




**2D toolkit classes :**
http://www.unikronsoftware.com/2dtoolkit/docs/1.92/html/classtk2d_animated_sprite.html

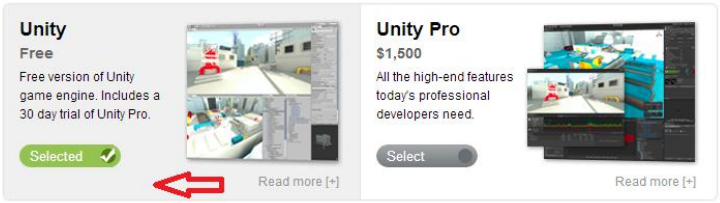
# Développement du client Unity3D

Unity3D est le logiciel qui a été choisi pour développer le client du jeu Akhiris. Il sert surtout aux développeurs, mais nous disposons aussi d'une extension 2D qui peut être utile aux pixel-artistes et animateurs.

## Téléchargement

Tout d'abord, téléchargez-le sur le site : https://store.unity3d.com/

Il existe deux versions de Unity3D. La version Free (gratuite) est celle qui nous intéresse ici. Elle ne possède pas toutes les fonctionnalités de la version PRO, mais suffira amplement pour compiler le projet. Attention, l'installateur pèse tout de même plus de 650Mo !




## Installation

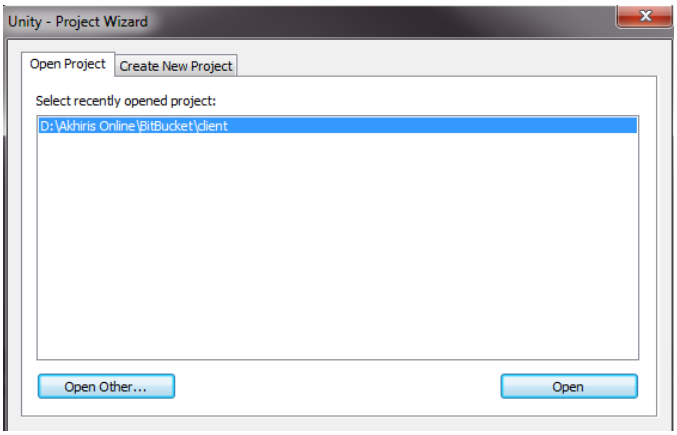
Une fois le téléchargement terminé, installez le logiciel où vous le souhaitez, en suivant les indications de l'installateur.

## Importation du projet Akhiris Online

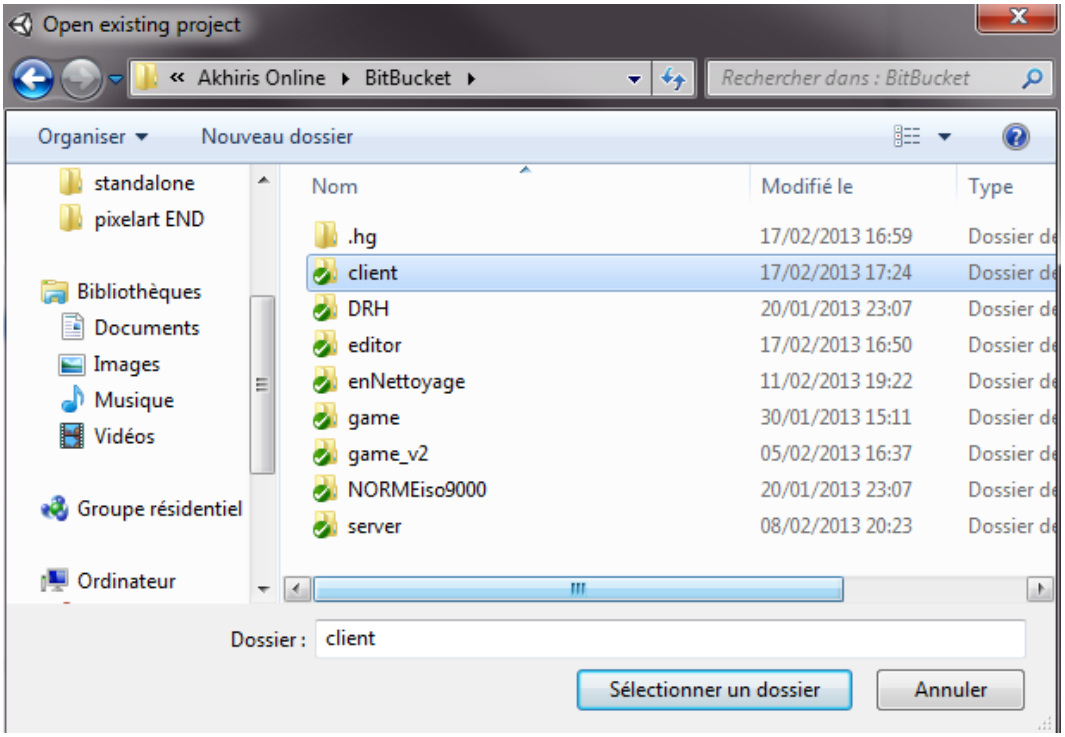
On va maintenant importer le projet du jeu, à partir du dépôt officiel sur Mercurial.

Si vous avez déjà le dépôt Mercurial sur votre ordinateur local, faites juste un « hg pull ».
Sinon, téléchargez une copie du dépôt ici : https://bitbucket.org/OlivierL/akhiris/get/default.zip

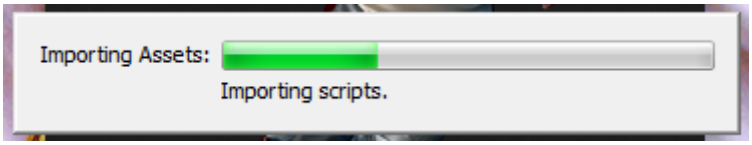
Il ne vous reste plus qu'à ouvrir Unity3D, une fenêtre devrait s'ouvrir, vous permettant de choisir un projet à ouvrir.



Cliquez sur « Open Other » en bas à gauche, et allez dans le dépôt. Il vous suffira de sélectionner le dossier « client », puis de cliquer sur le bouton « Sélectionner un dossier »

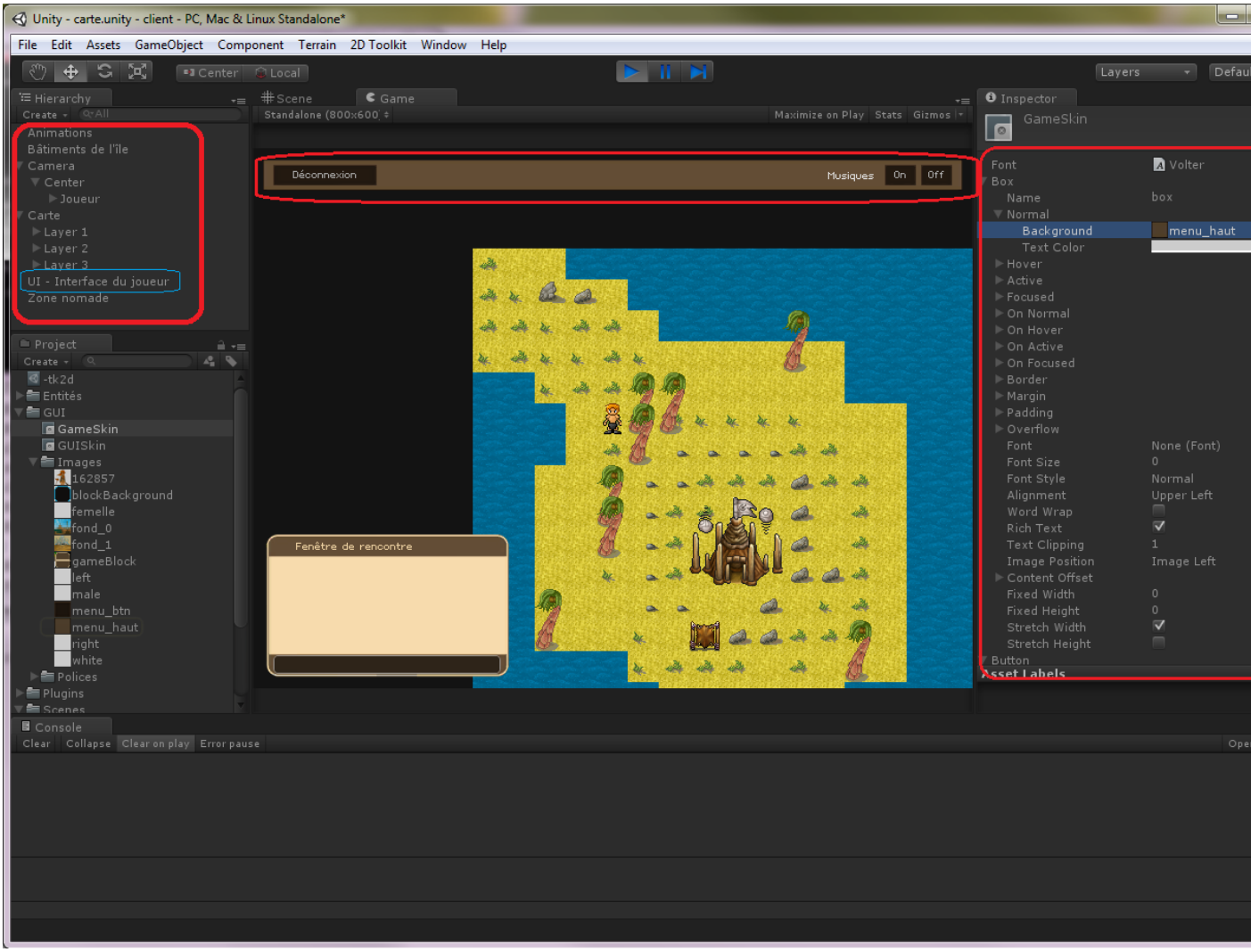



Unity3D devrait se mettre au travail et importer tout le projet !



# Les scènes principales du projet

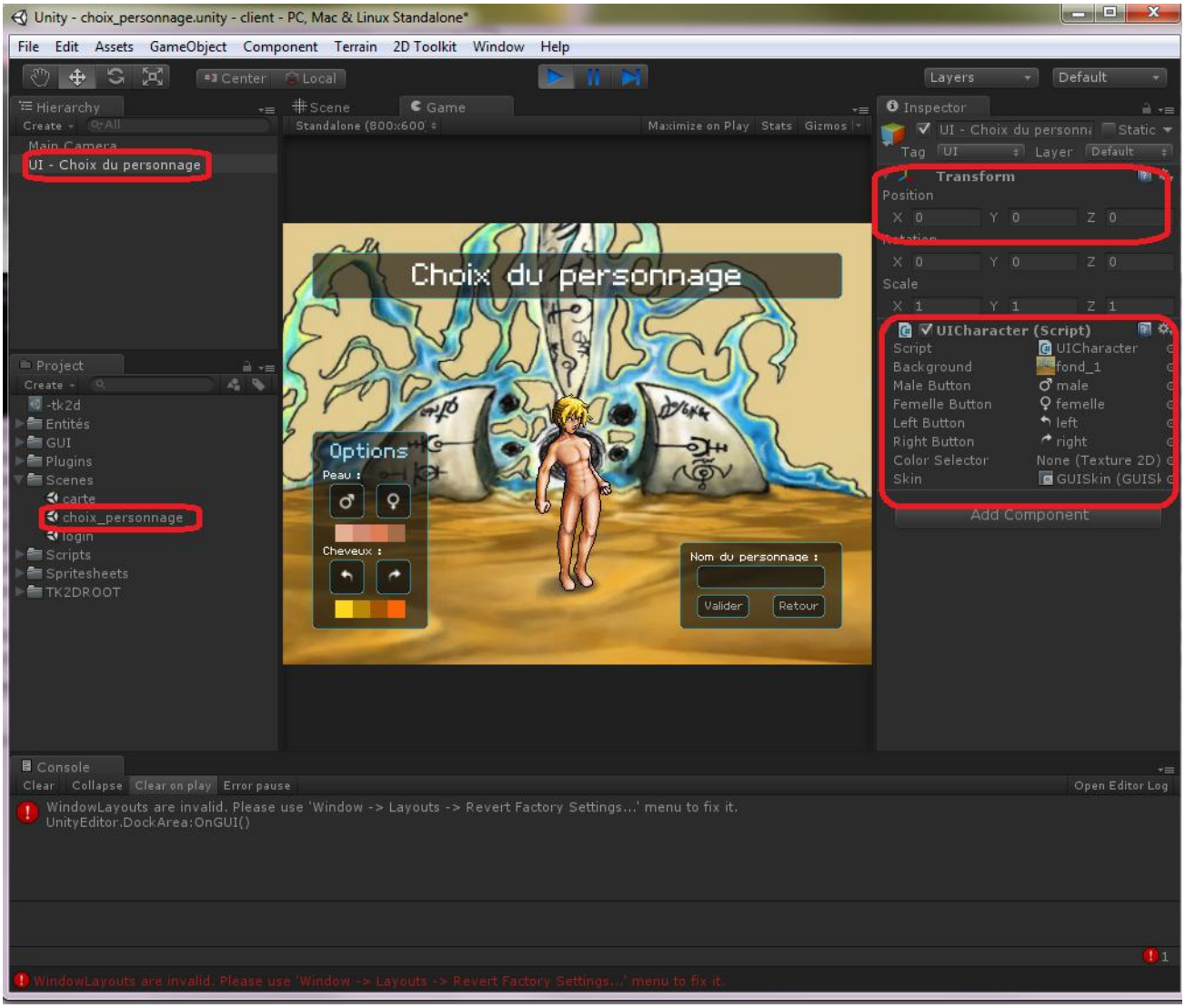
## Scène de la carte principale :




A gauche en rouge : la liste des entités affichées

- Animations : les packs packagées à partir des spritesheets
- Camera : vue principale, on peut varier le niveau de zoom, les résolutions d'écran et la perspective (et joueur centré en permanence, car enfant de la caméra)
- Carte : contient les ressources chargées dans http://akhirisonline.com/game_v2/data/cartes/[nom_delacarte].xml du dossier donc : ici
    - Couches de transparence « Layer 1 / Layer 2 / Layer 3 » pour les sprites
    - PNJ et autres entités de jeu (sprites complexes définis dans l'éditeur ou Unity3D, préfabs)
- UI – Interface du joueur : définit l'interface de l'IHM navigation. L'interface peut être définie par un script, mais la plupart des modifications se font dans le thème graphique : GameSkin (à droite de l'écran, et onglet projet en bas à gauche) ca ressemble à des paramètres CSS, selon le style voulu (on peut en définir autant qu'on veut et faire des tests en direct).

## Scène du choix du personnage :




Même disposition ici.

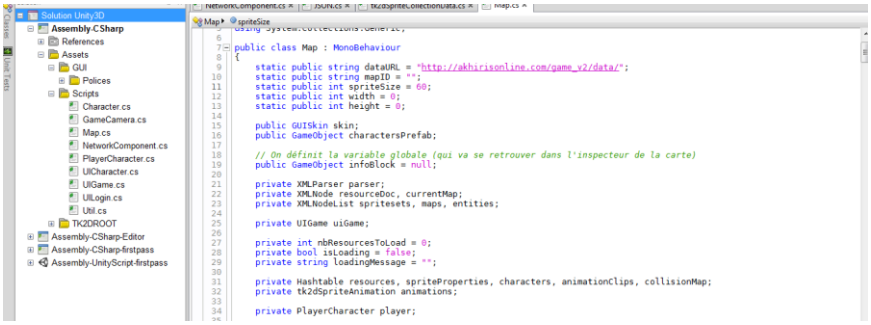
On délègue l'interface graphique au script UI – Choix du personnage, mais on demande un certain nombre de paramètres pour le background et les différents boutons / styles (à droite)

Le placement de chaque élément graphique (bouton, titre, cadre) et le chargement des données du skin est effectué de façon dynamique dans le script UICharacter.cs de sorte que développeurs et infographistes s'y retrouvent.

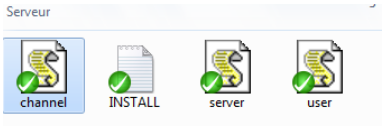
Les scènes sont stockées dans le même dossier réservé.

# « Middleware »

Chaque classe herite de MonoBehaviou : enfin presque toutes.
La classe Monobehaviour




Le client se connecte au serveur.
(adresse ip du serveur hébergeur)




Ces fichiers ont été lance sur le serveur qui stock 2 tables : users et permet en JSON (transfert XML) est la MAP et les ENTITES sont stocké en ftp(XML donc arbre ) .Le « middleware «  permet d'effectuer les comportements sur la carte par ces 2 biées.

Exemple de code  pour WebService : http://www.isima.fr/~lacomme/pagewebservice/webservice/
" le webservice » c'est  ( *entre Editeur // ftp-> // Serveur*)

Puis  Dialogue entre Client et Serveur par: **Socket**
( *Serveur  //<-SOCHET-> //Client*) "
Au départ : on instancie la session (transfert l'id client … style **REST** ) . Mais après tout passe par le socket
Le socket TCP est un protocole d'échange réseau. HTTP tout comme FTP : est construit a partir des sockets TCP.
Pour Akhiris, Client utilise les sockets TCP pour les échanges rapides d'info avec le serveur, et HTTP pour le transfert de ressources XML / PNG Presque tous les échanges réseau passent par des sockets.

### 1_ JSON et un concept différent de l AJAX

- JSON c'est une structure de données (ca permet de sauvegarder et récupérer facilement une information/donnée) , au meme titre que le XML indexe les données

- l'AJAX c'est un protocole d'accès (au meme titre que les sockets, d'ailleurs c basé sur des sockets) qui permet d'envoyer et recevoir de l'info entre le navigateur web et le serveur web

Cela permet notamment de faire du PHP interactif un genre de tchat par navigateur entre le navigateur et le serveur ou de récupérer des infos meme une fois que la page web a terminé de se charger en gros, tu fais en javascript :

AppelAJAX("http://akhirisonline.com/obtenirHeure.php") et ca te retourne le contenu écrit par le fichier php *ici ca donnera : 15h14*

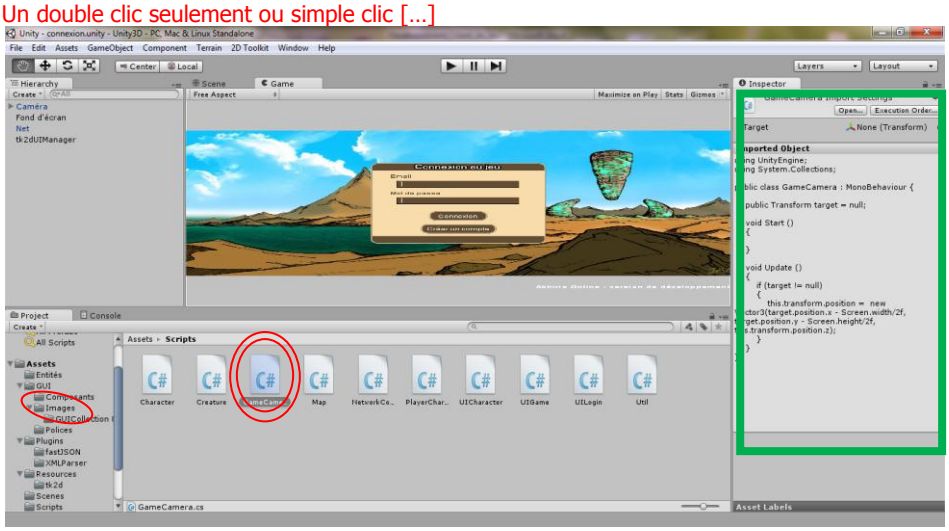
# CODER

- On peu mettre l'éditeur de code a que l'on préfère :

MENU//EDIT//PREFERENCES//External Tools ( MonoDevelopp, NPP )

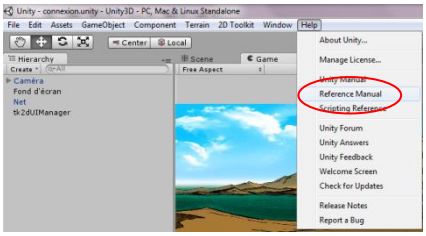
- Accéder à l'éditeur de code **(voir P20(pasfait) pour le rangement sur le treeview )**

Un double clic seulement ou simple clic [...]

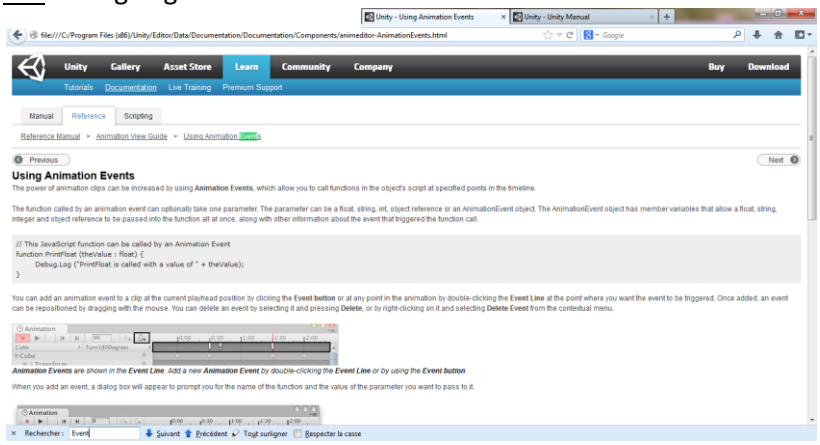


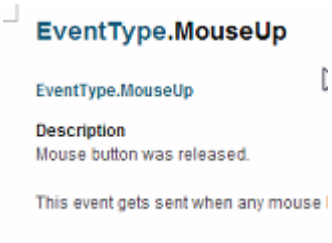
Visualiser

- Accéder à l'aide



Ex : Debug.Log

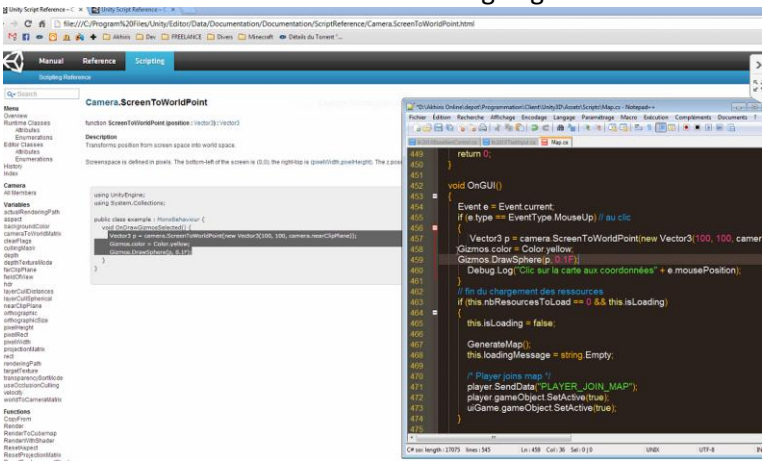




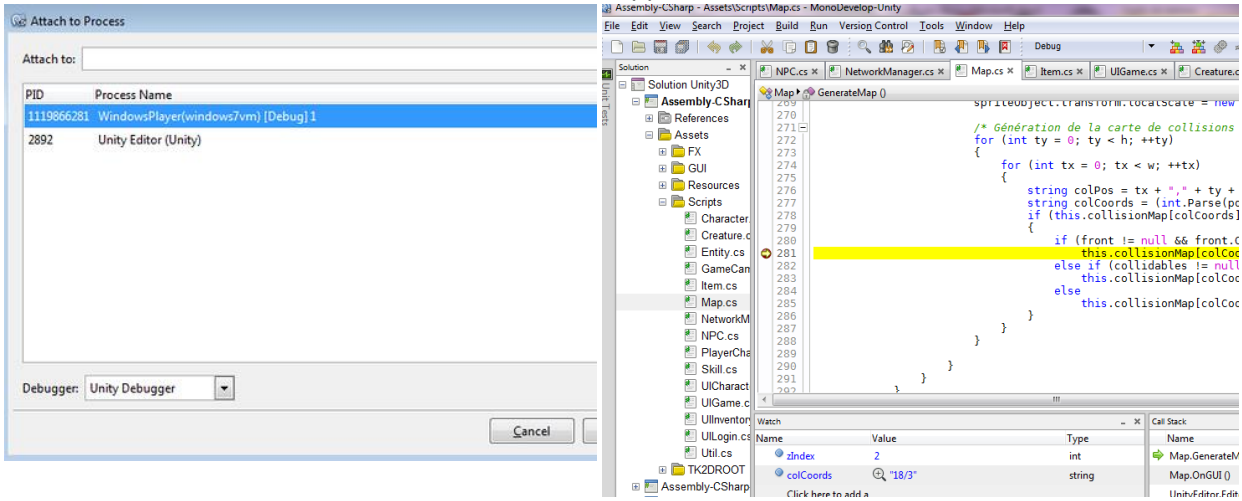
-Tester //Debugger

Tester :


DEBUGGER avec Debug.Log



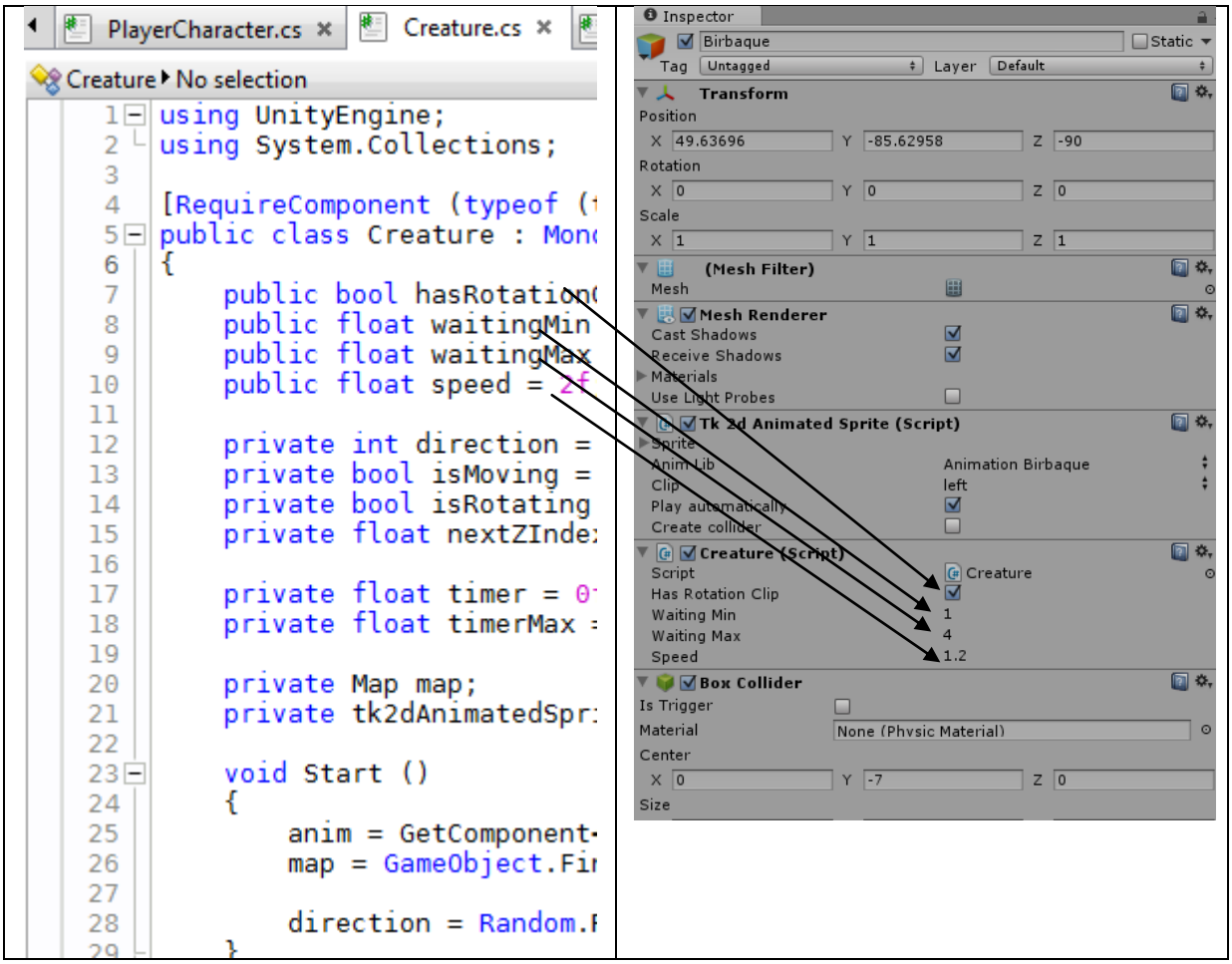
DEBUGGER avec Mono Developpe//Run//Attach Process



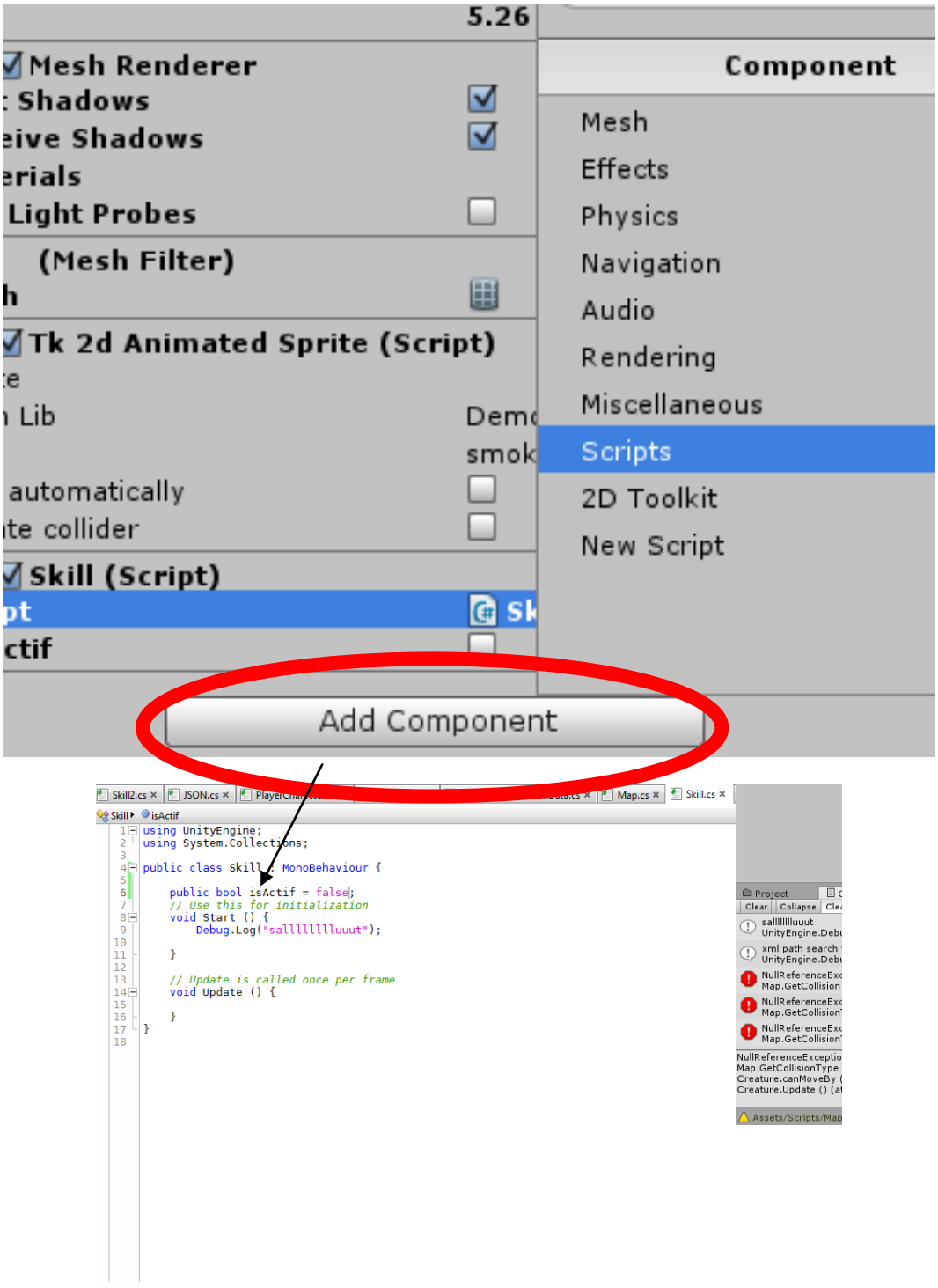

Pour avoir les point d 'arrêt

**BON A SAVOIR**

Une propriété de script "publique"
Elle peut être modifiée sur l'éditeur Unity
Ex :



# Commencer à PROGRAMMER



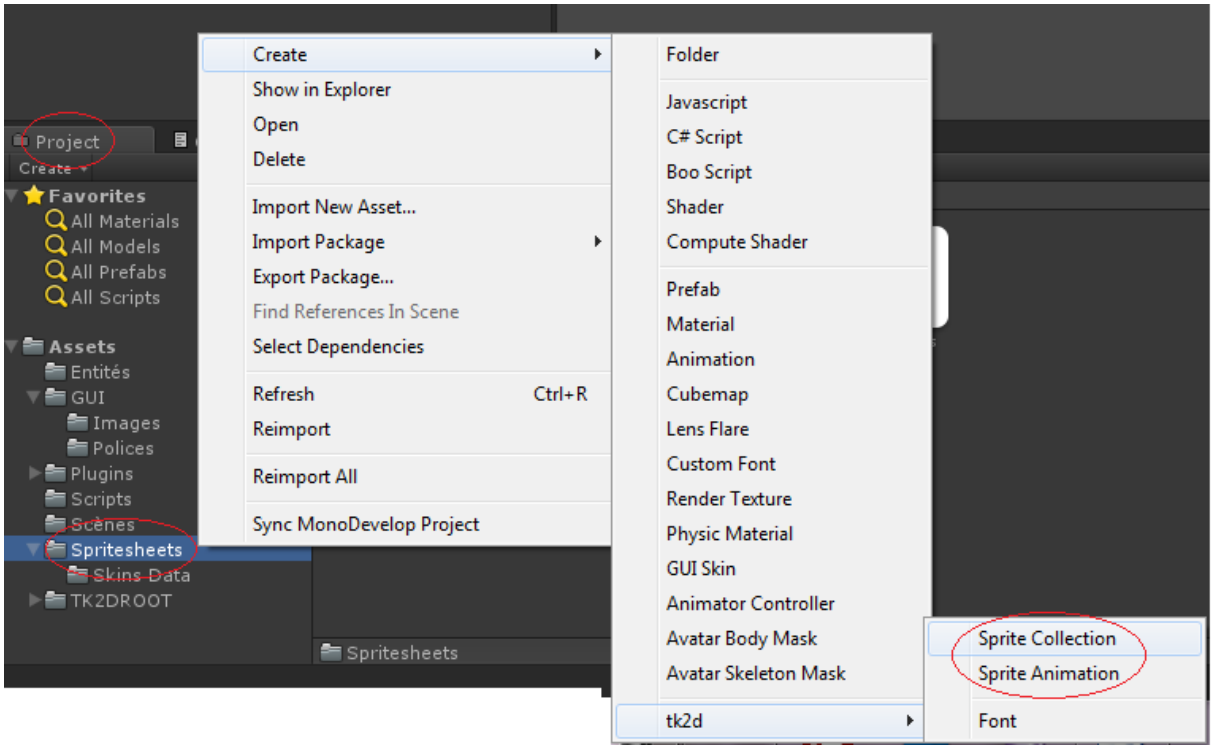

**unity3d LIRE les .psd**

# L'extension 2D Toolkit

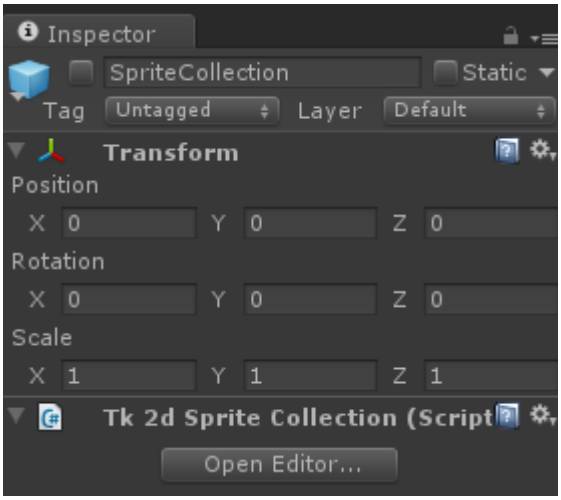
Le projet Akhiris dispose de l'extension 2D Toolkit pour le développement en 2D.
Nous allons voir ici comment créer un sprite animé.

Tout d'abord, nous devons ajouter l'image à animer au projet. Il vous suffit de la copier dans le dossier *client/Assets/Spritesheets*

Une fois l'image référencée, créons une collection de sprites (« *Sprite Collection* »)

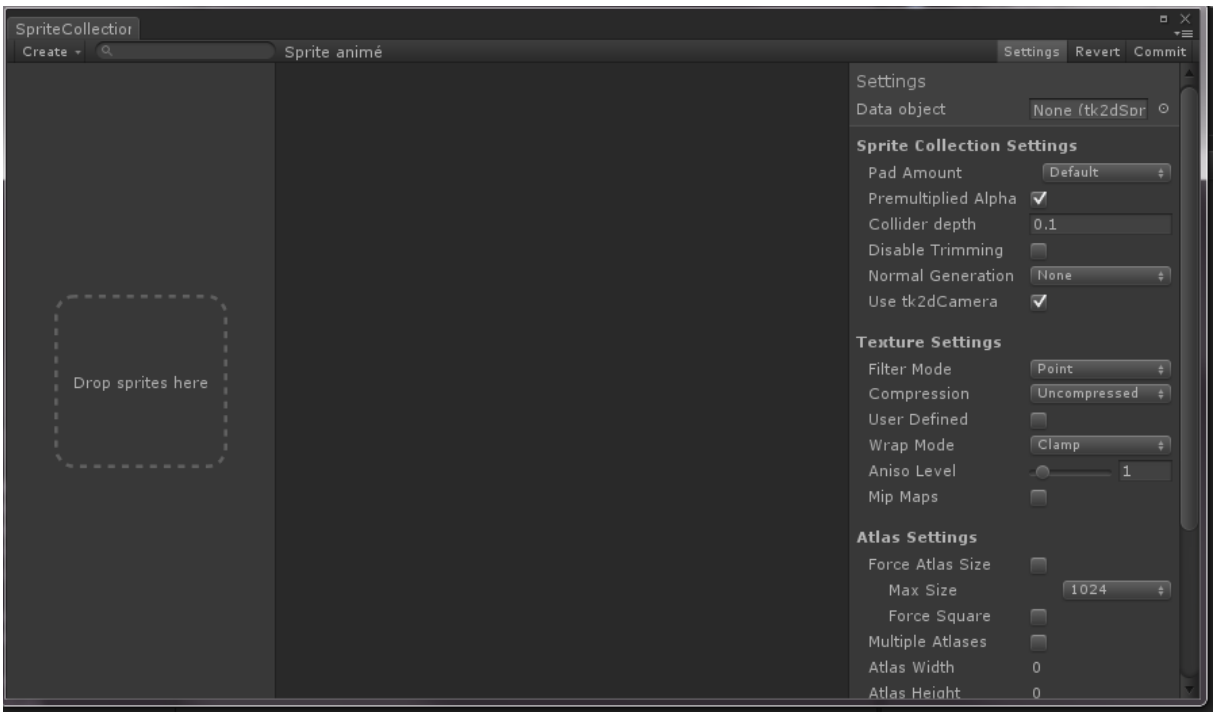



L' « *Inspector* » nous permet de voir les propriétés du composant créé, en l'occurrence notre collection. Renommez le champ texte en haut, puis cliquez sur *« Open Editor… »*

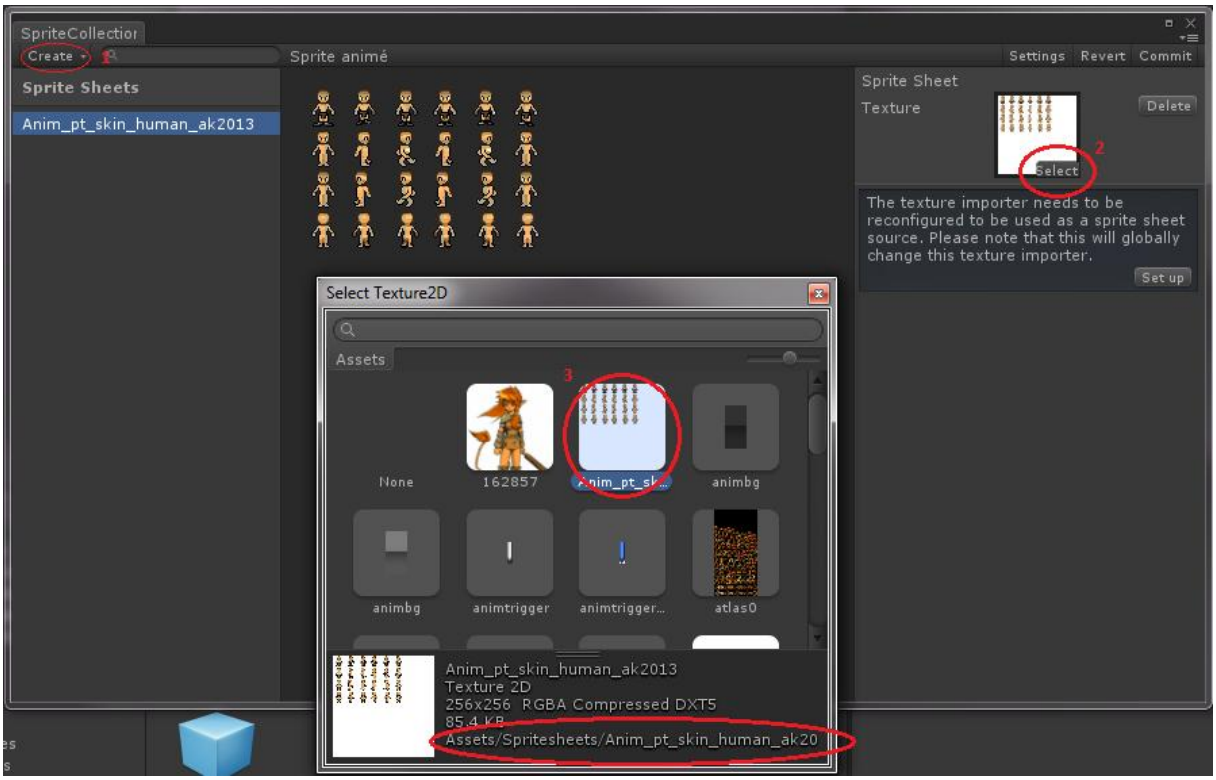


L’éditeur de spritesheet ci-dessous devrait s’ouvrir.

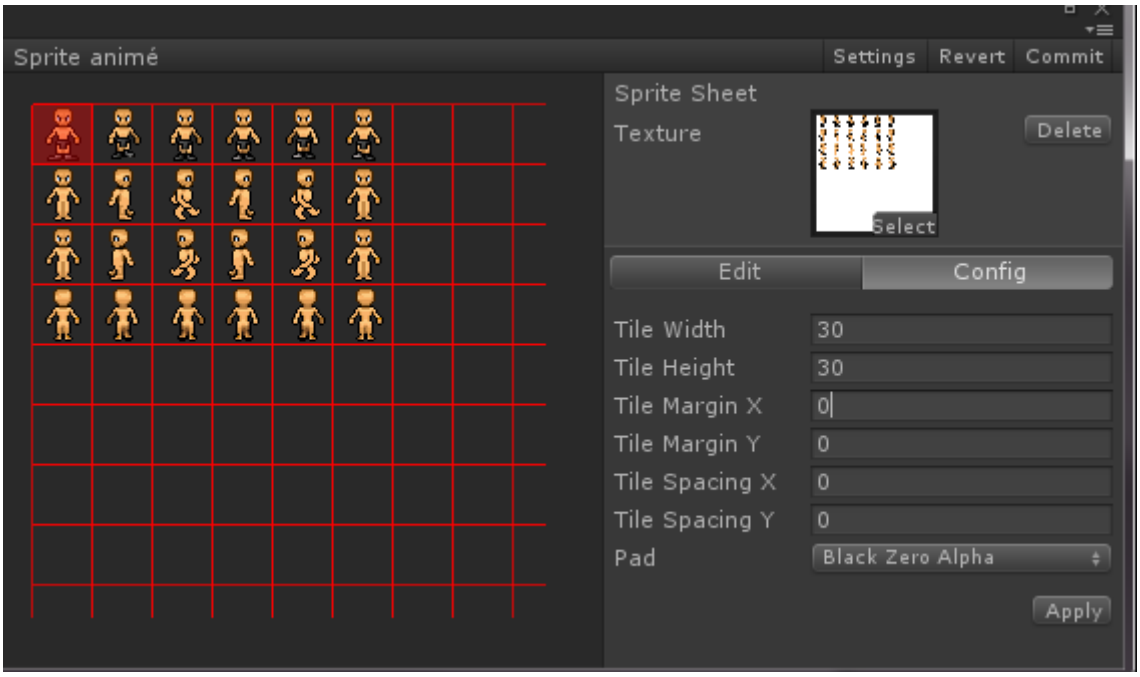
Vérifiez que vous avez les paramètres définis ainsi dans l’onglet « Settings »




Ajoutons maintenant notre spritesheet en suivant la procédure ci-dessous.
Bouton « Create > Spritesheet »,  bouton « Select Texture », et cliquez sur le spritesheet souhaité.

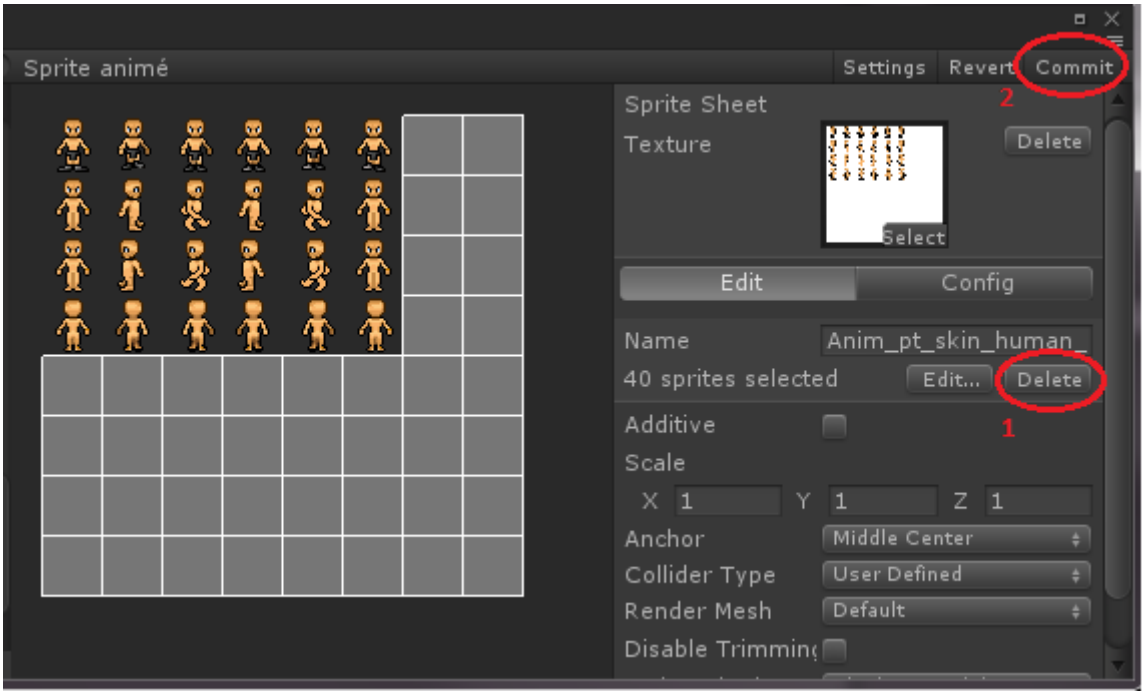


Ok, il nous reste plus qu'à découper l'image (ici en morceaux de 30*30) et le spritesheet sera défini une fois pour toutes. Cliquez sur « set up » à droite, si vous avez le message « The texture importer needs to be reconfigured …. ». Puis rentrez les paramètres que vous souhaitez :



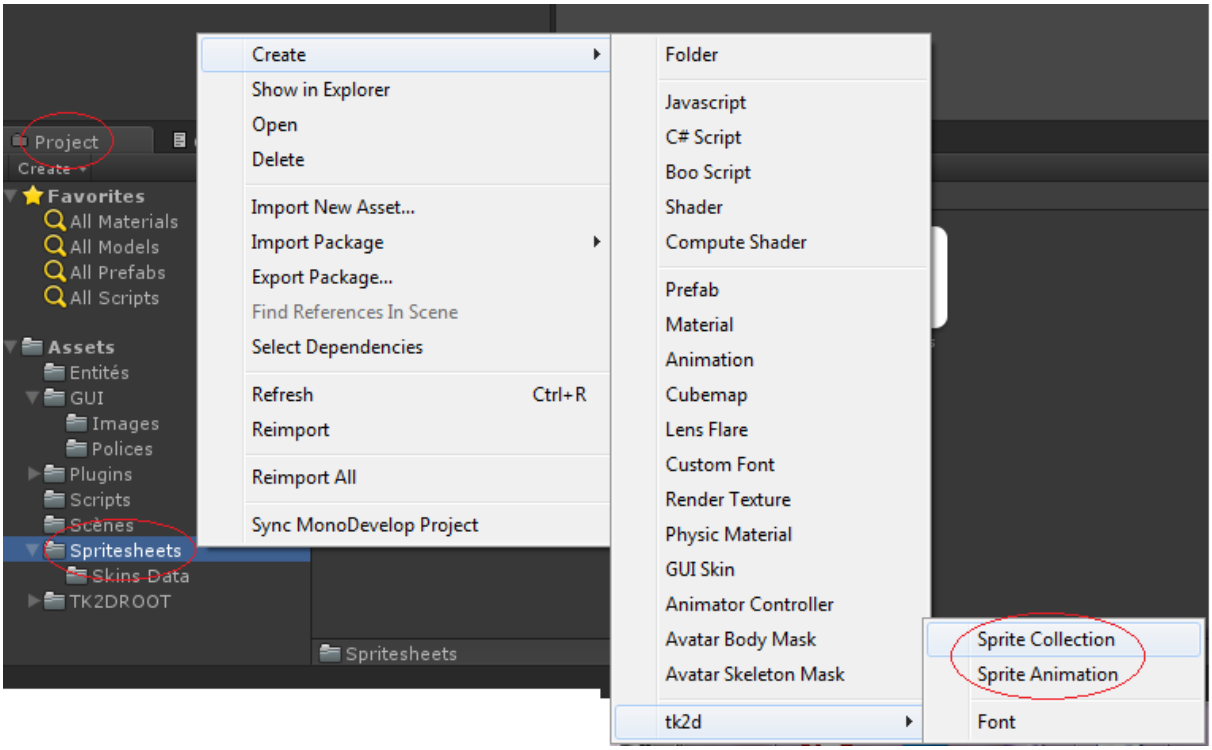

**Black Zero Alpha** est l'option préférée pour les animations de personnages. Cliquez sur « ***Apply*** » pour que le spritesheet se découpe en « sprites » uniques de 30*30.

Supprimons les parties vides en traçant un rectangle de sélection, comme indiqué (1)



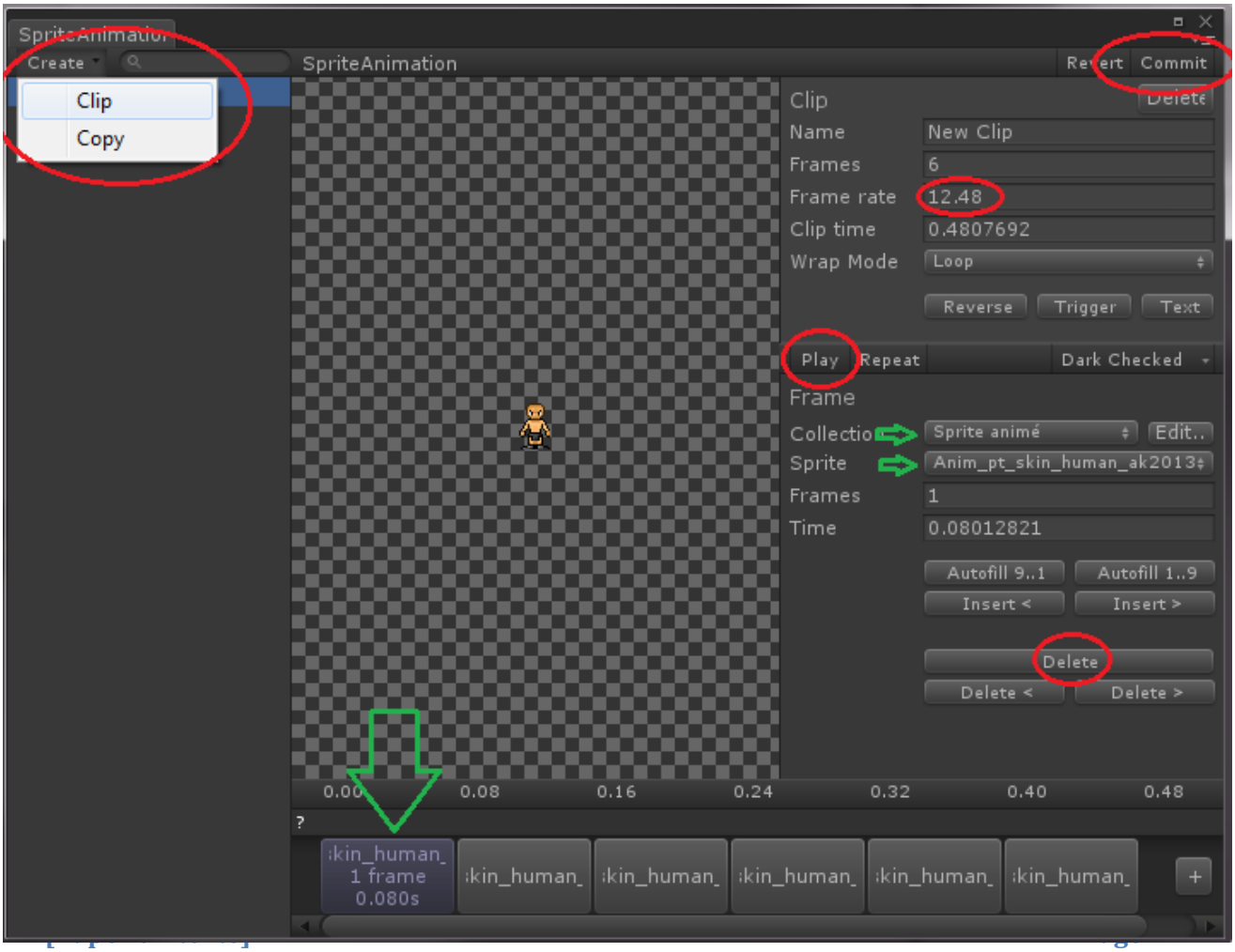

Cliquez ensuite sur Commit (2), cela sauvegardera les changements. Vous pouvez maintenant fermer l'éditeur de spritesheet.

## Dernière étape : la création de l'animation




Faites un clic droit sur le dossier Spritesheets, mais cette fois, sélectionnez « *Sprite Animation* » au lieu de Sprite Collection. Dans *l'Inspector*, cliquez à nouveau sur Open Editor.

Voilà comment se présente l'éditeur d'animations :



Pour construire l'animation, cliquez d'abord sur Create (en haut à gauche) , puis Clip.

Vous allez devoir maintenant ajouter les images frame par frame à l'animation. Pour cela , sélectionnez la collection de sprites, et le sprite 30*30 souhaité (petites flèches vertes)
Cela créera votre première frame d'animation (grosse flèche verte).

Vous pouvez en ajouter d'autres automatiquement, avec « Autofill » ou manuellement en cliquant sur le bouton [+] à côté de votre frame.

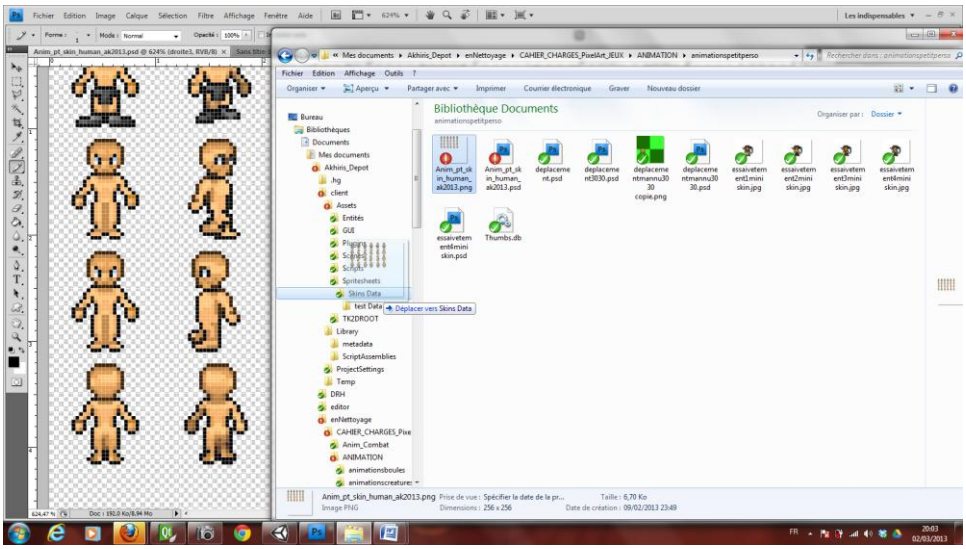
Une fois que vous êtes satisfait, lancez l'animation avec le bouton **« Play »**
Vous pouvez également modifier la vitesse de l'animation dans le champ Frame Rate (frames par secondes)

Si l'animation est calée, vous pouvez sauvegarder avec le bouton « Commit » en haut à droite.

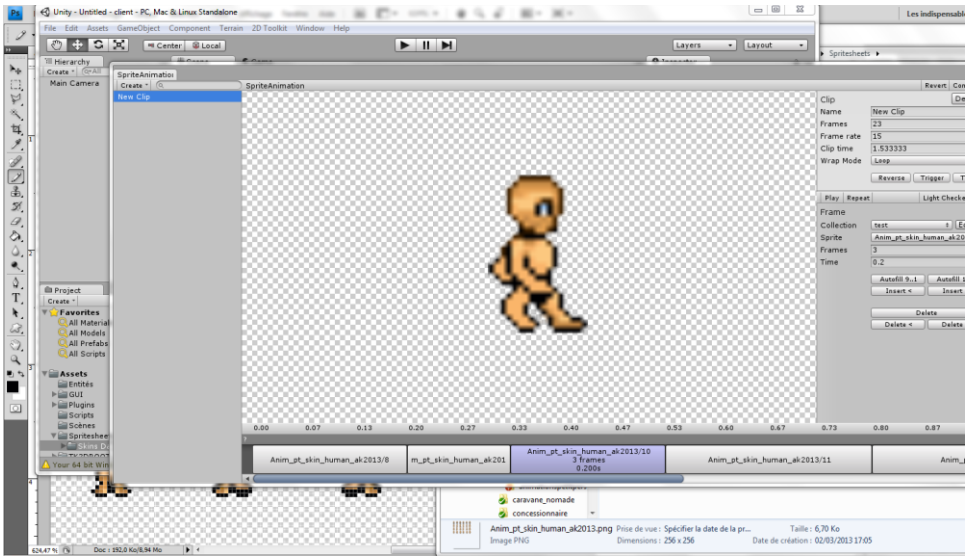
Pour obtenir des informations supplémentaires sur l'utilisation de l'outil 2D Toolkit, il existe une documentation assez fournie sur le site officiel, pour animateurs et développeurs :
http://unikronsoftware.com/2dtoolkit/doc/ (en anglais)

# Début de Guide pour Pixel Artiste

## Actualiser une image de Photoshop vers Unity3D




Penser juste à garder une fenêtre avec l'arborescence Windows pour actualiser l'image sur Unity lors de modifications.



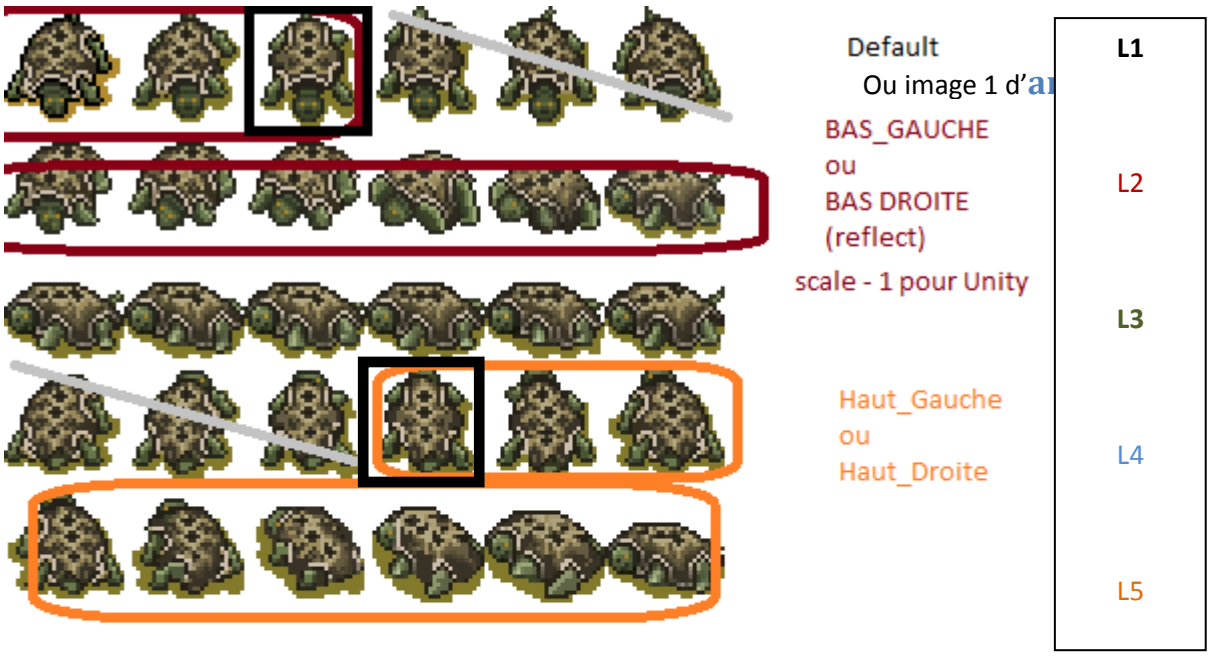
Unity actualise rapidement l’image si vous avez un doute (F5) pour rafraîchir.

# Norme de Spritesheet
# (LES CLIPS Creatures)

LA DUREE DES ANIMATIONS DE MEME TYPE DOIVENT ETRE EGALES ENTRE ELLES

Exemple :Cas concret : (chaque cas est différents il en vient de l'analyser)
Ici le bibarque est lent

*Coté graphisme on a:* **les couleurs sont pour la prog ou infographie**




je crois que cela ne sert pas

Ici nous avons 2 types de durée d'animation :

- LES ROTATIONS
- LES DEPLACEMENTS EN HORIZONTAUX ET VERTICAUX

*Coté programmation on a les déplacements d* ***animations*** *:*

- LES DEPLACEMENTS EN HORIZONTAUX ET VERTICAUX


- LES ROTATIONS

Rotate_
de**BAS**_à_Gauche **L1**/2+ L2

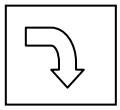
Rotate_
de**HAUT**_àDroite L4/2 + L5

Rotate_
de**BAS**_à_Droite (Scale= -1 de **L1**/2+L2)

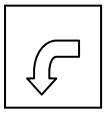
Rotate_
de**HAUT**_àGauche (Scale = -1 de L4/2 + L5 )

On par de l'**animation** image 1 ou « par default »

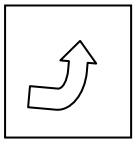
Rotate_
deDroite_à_**BAS** (Scale :inverse de :**L1**/2+ L2)



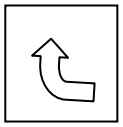
Rotate_
deGauche_à_**BAS** (inverse de :**L1**/2+ L2)



Rotate_
deDroite_à_**HAUT**(Scale :inverse de : de L4/2 + L5)



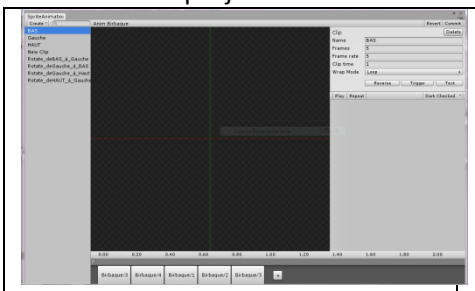
Rotate_
deGauche_à_**HAUT**(inverse de : de L4/2 + L5)



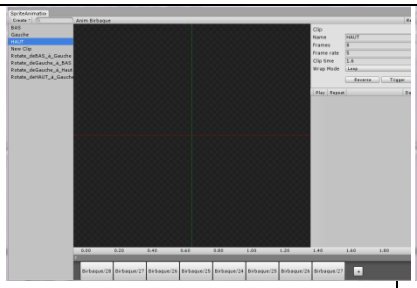
On arrive à l'**animation** image 1 ou « par default »

# Procédure d'importation MONO_Développeur du Projet sur le dépôt

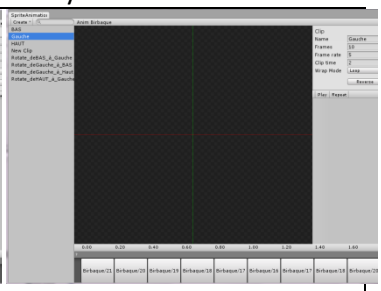
Définir les clips de l animations tel que vu précédemment.
Faire dans un projet en local « Bac à sable » et tester les animations sous Unity :



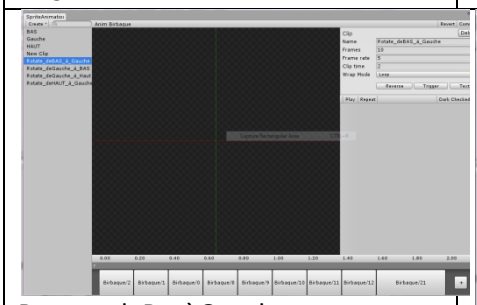
BAS



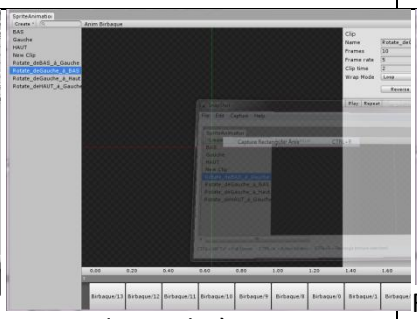
HAUT



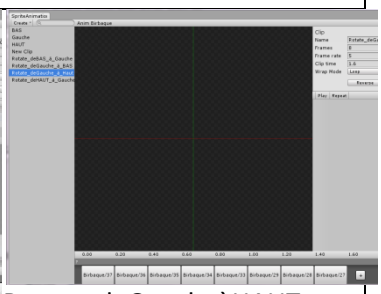
Gauche



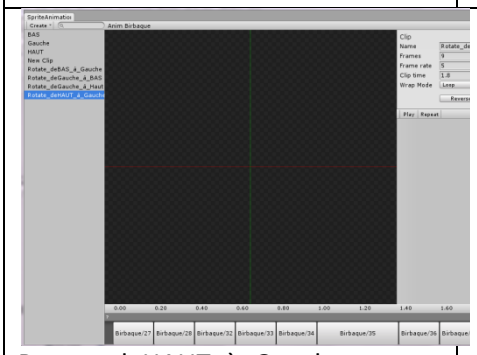
Rotate_deBasàGauche



Rotate_deGaucheàBAS



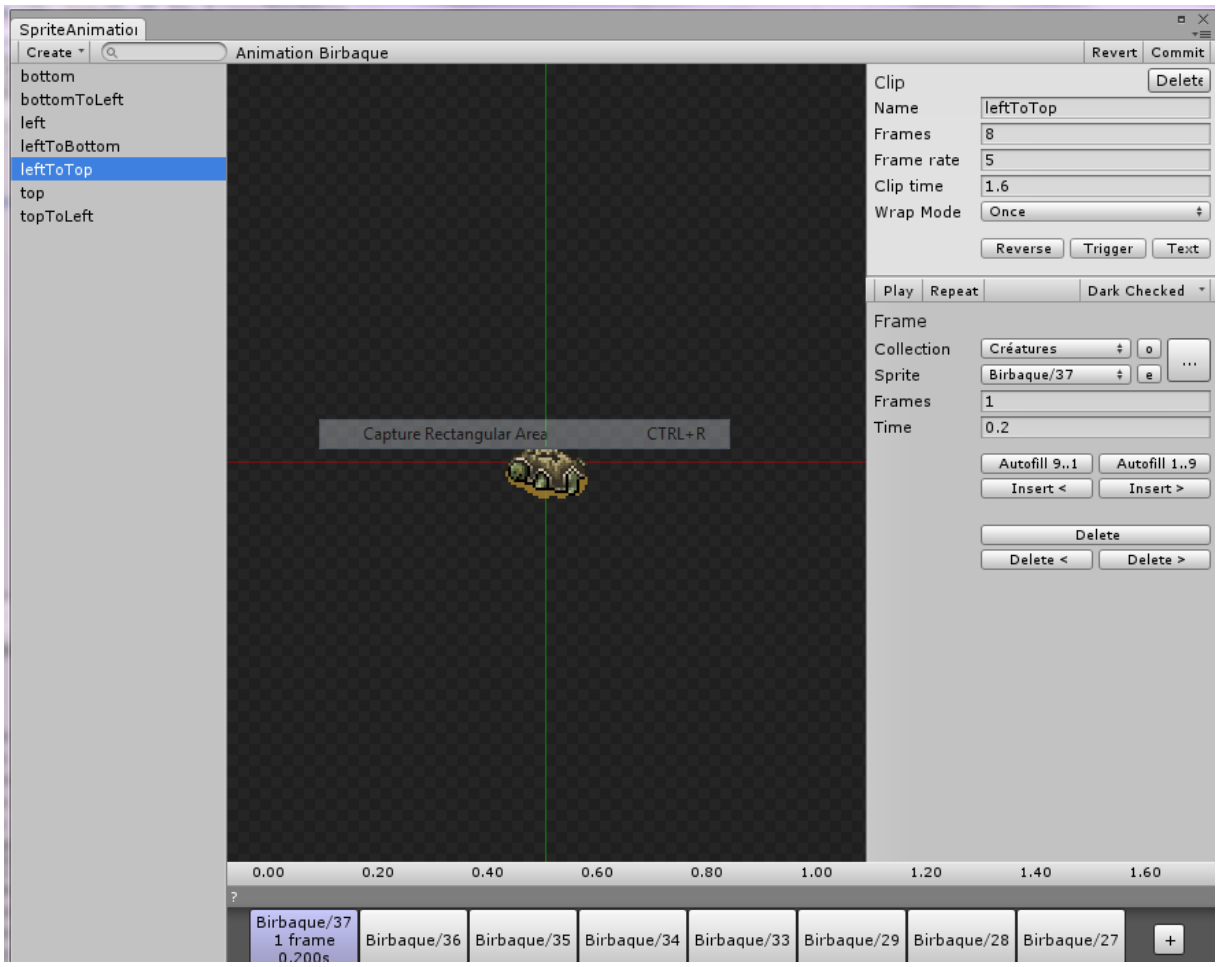
Rotate_deGaucheàHAUT



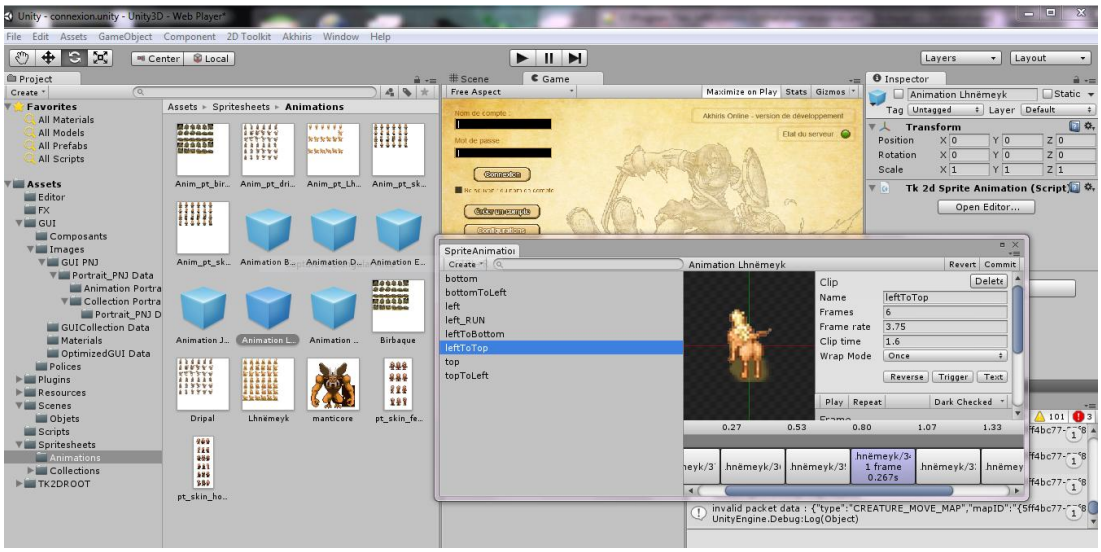
Rotate_deHAUT_à_Gauche

-Quand tous est testé et vous semble valide pour réaliser tous les mouvement d'actions possible ici du « birbaque « .
-Refaire les mêmes clips directement sur le projet sur votre dossier du Dépôt en un minimum de temps et le pousser sur le dépôt. ( en s'appuyant des screens ou de l'autre projet en local )

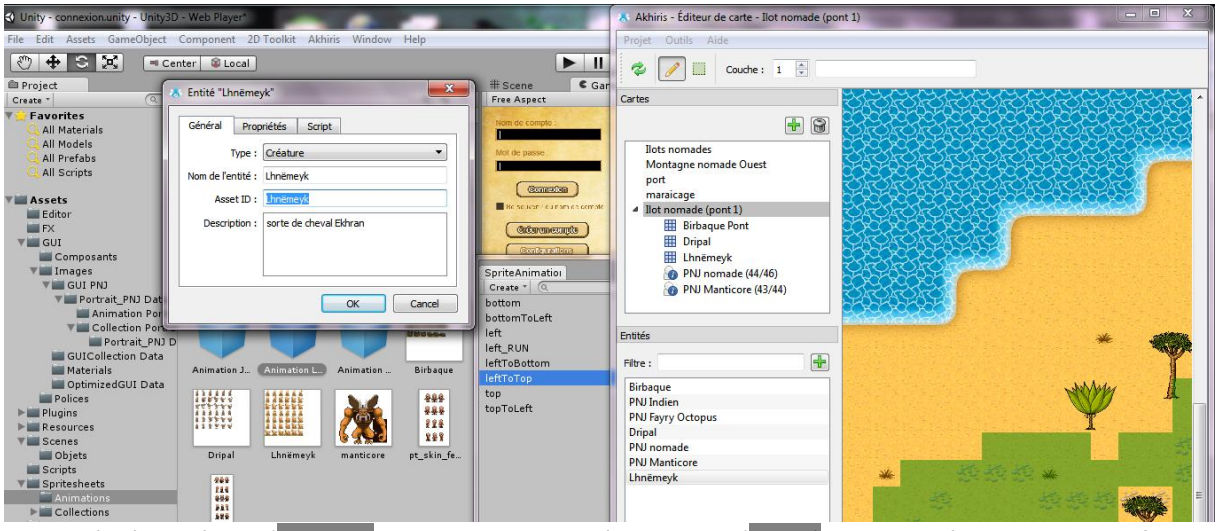
# Au final on a : 7 clips minimum



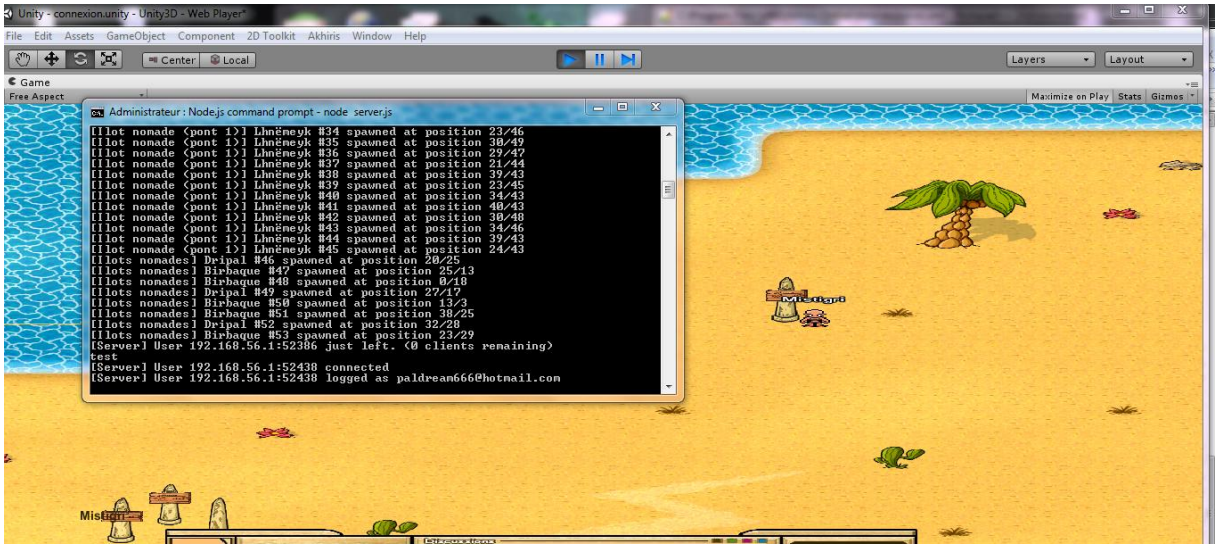
# Particules sur Création d'une Entité suite à la création des Clips de déplacement de celle-ci.



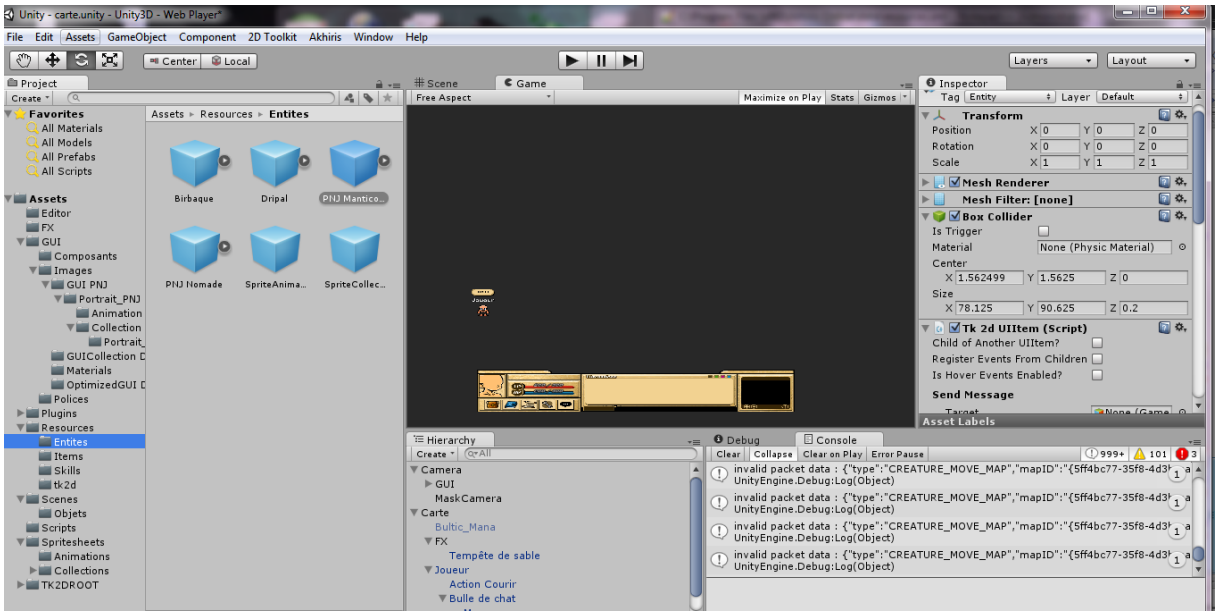
On commence par analyser si les clips de SpriteAnimation sont complet pour le déplacement

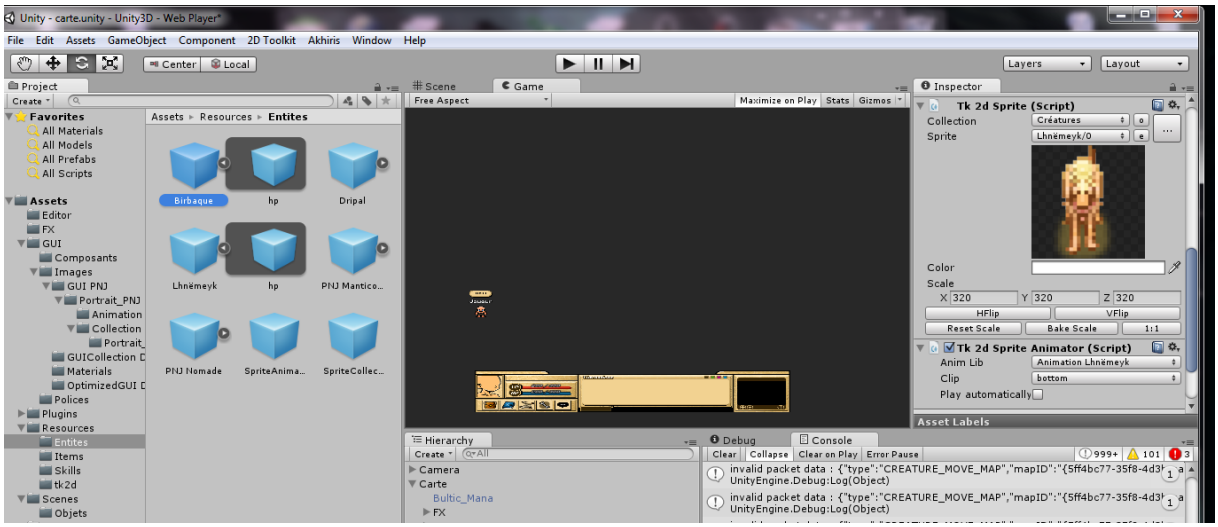


On crée l'entité : ici l'Asset ID est le nom de la créature : de l'entité dans la Scéne du projet. Il faut aussi créer la zone de spawn et éditer le nombre de créatures invoqué dans cette zone.



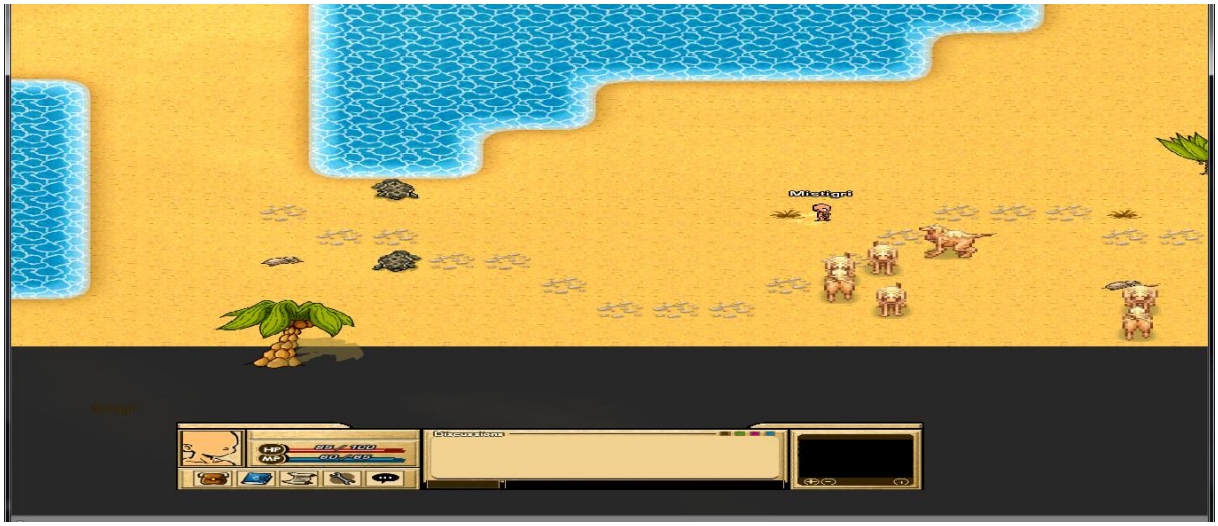

On vérifie sous node.js à son démarrage que le serveur lance bien les entités.




On peut glisser depose l'Animation Lhnëmeyk dans la Scéne de la carte ; puis glisser déposer

l’animation dans les entités ; puis glisser déposer l’entité Birbaque dans l animation Lhnëmeyk pour remplacer ces propriétés et modifier les champ Anim Lib ( les deux noms )

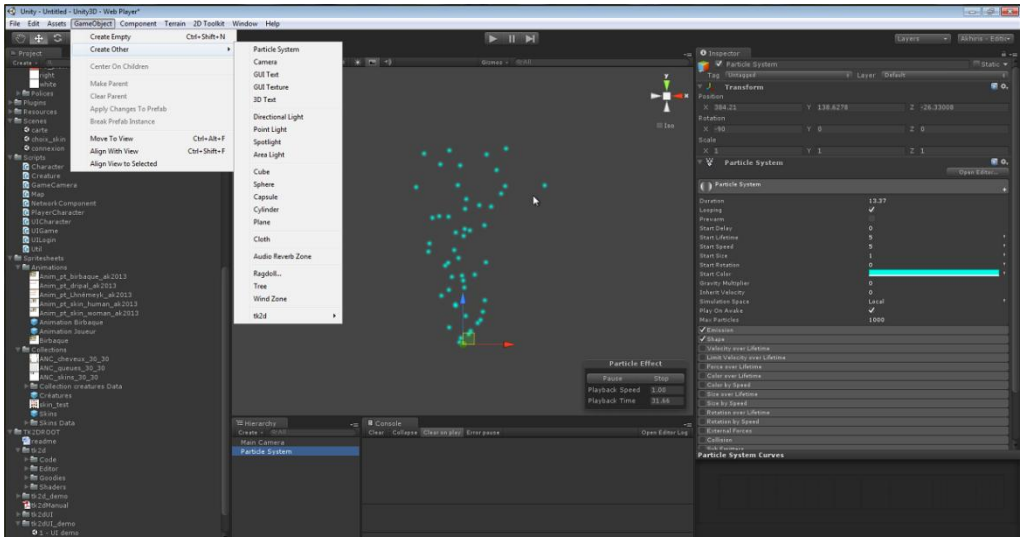


On obtiens enfin l’animation de base du Lhnëmeyk avec l’ancienne configuration du Birbaque. ( En souhaitant qu’on a respecter la « **Norme** de déplacement du Birbarque vu au dessus »

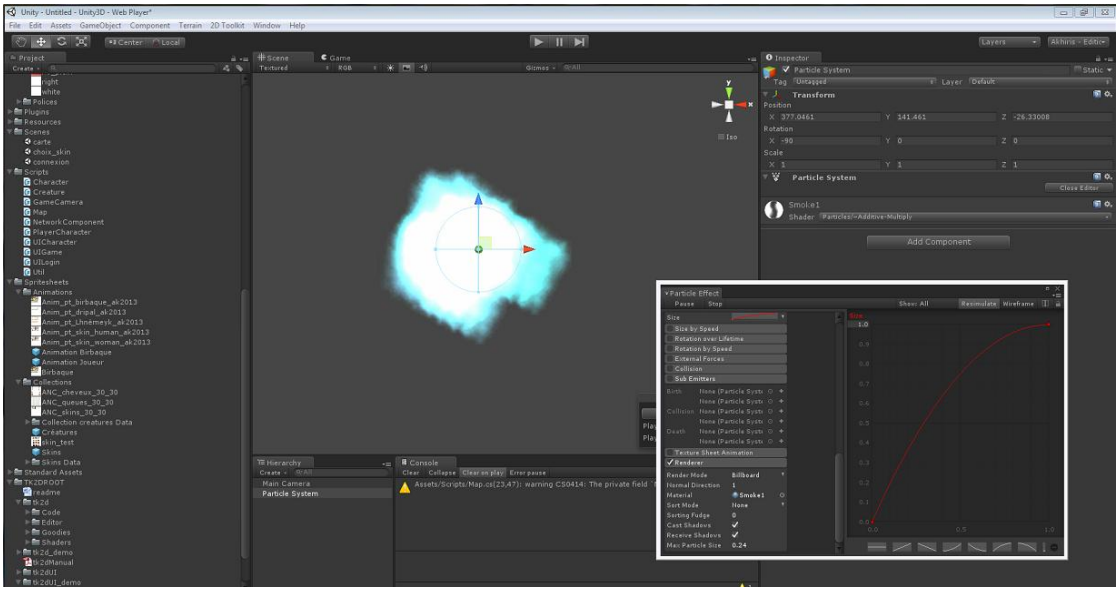
# Particules sur Unity3D

## Pour Les Skills et effets très dynamique

Pour réalisé certaine fonctionnalités de jeux nous employons des outils qui permettent de simuler les effets de la vie réel ou de les amplifier.



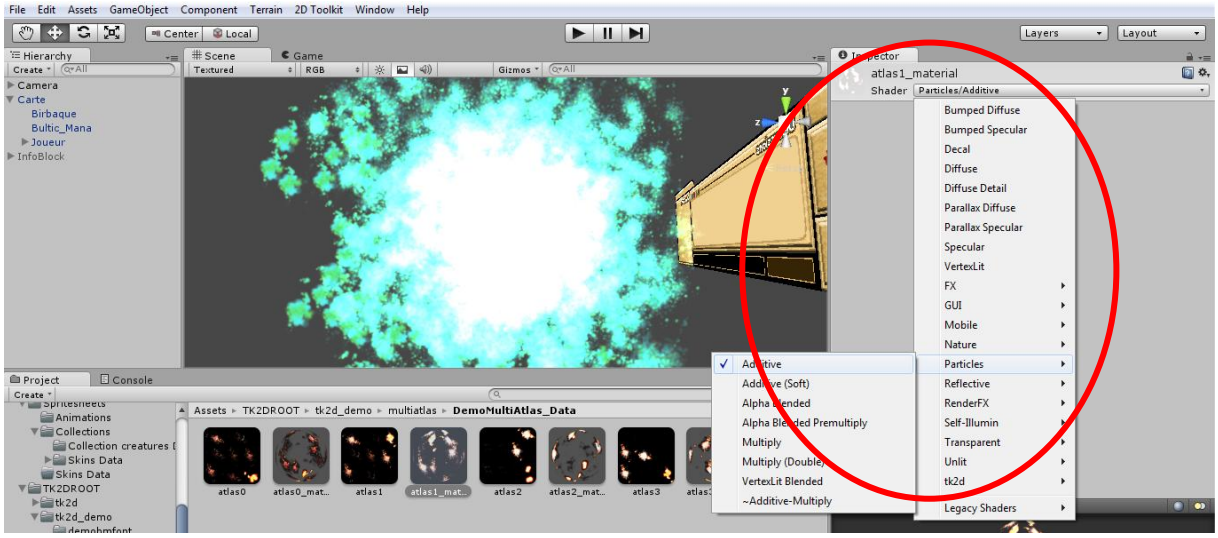
**Accéder a cet outil**



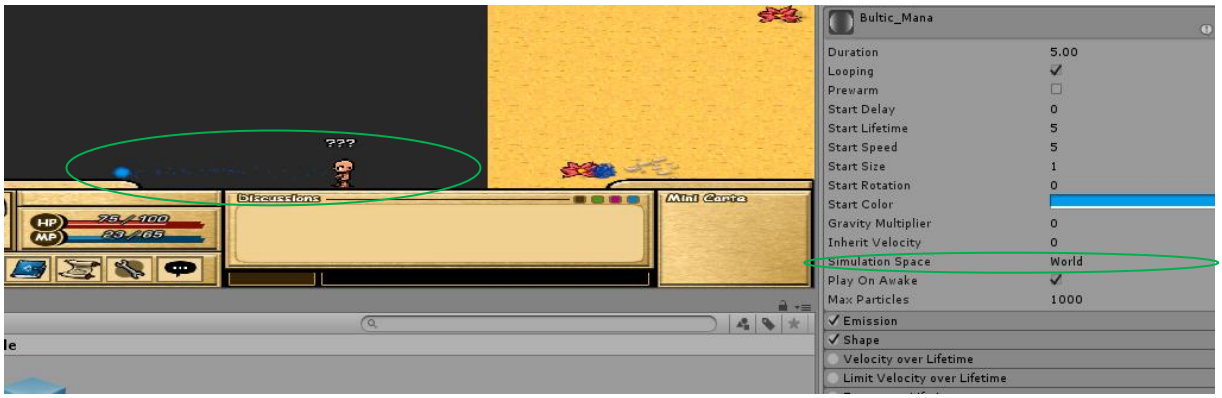
**maxSizeEffect**

# Effet sur Particles de fusion ou de Filtre

Les effets layer style photoshop



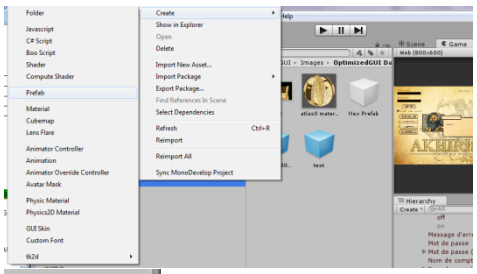

L'effet trainé de poussière de la particule



# Création d'un Fenêtre

Les éléments utiles à savoir :

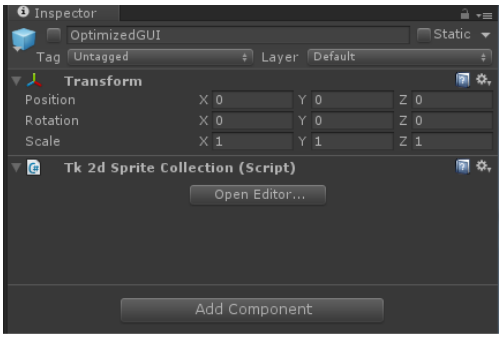
Pour créer un Gui :

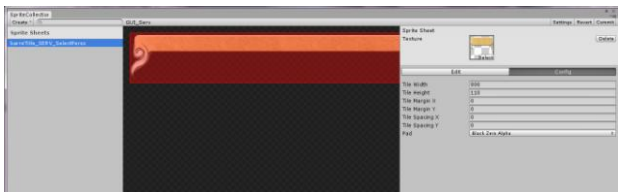


Create Prefab

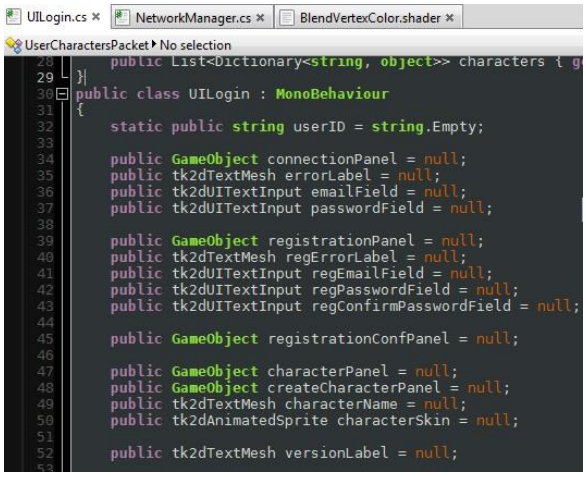

Glisser un Objet pour le faire devenir Optimized GUI par exemple ( un spritesheet )

Pour editer un GUI : ouvre OptimizedGUI (dans Assets/GUI/Images)



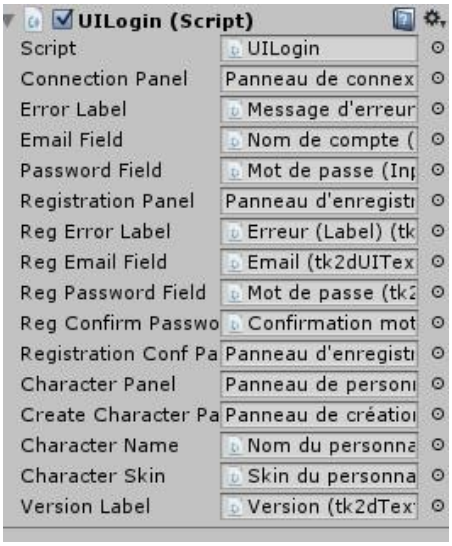



Pour créer un sprite : Faire attention à la taille du sprite qui doit être < à la taille de l'image lors de l'edition de celui-ci SINON pas de bouton Apply !




Il faut noter un grand interêt prochain pour les GameObject :

S'est eux qui contrôle les éléments graphiques de l'interface. Ex : visible ou non




# Rangement Treeview

(prochainement )

# Remerciements

**NB :** Si vous ne comprenez pas un des points du tutoriel, merci de me le signaler

Robin Ruaux < robin.ruaux@gmail.com >
Développeur pour Akhiris Online

Louet Olivier < louet.olivier@gmail.com >
Pixel Artiste pour Akhiris Online