

1. Accueil Prison Escape	2
1.1 Guide pour utilisation du plugin 3ds Max et pour la création des composants customs	2
1.1.1 1- Compilation du plugin	2
1.1.1.1 1.1 - Compilation du plugin	3
1.1.1.2 1.2 - Compilation des nodes custom	3
1.1.2 2 - Création de composants custom	4
1.1.3 3- Écriture de la logique du component en C++	6
1.1.4 4 - Utilisation du plugin	7
1.1.5 5 - Utilisation de 3ds Max	11
1.2 How-To Articles	14

Accueil Prison Escape

Page de Prison Escape

Guide pour utilisation du plugin

- 1- Compilation du plugin
- 2 - Création de composants custom
- 3- Écriture de la logique du component en C++
- 4 - Utilisation du plugin
- 5 - Utilisation de 3ds Max

Recent space activity



Mathieu Parent

- Accueil Prison Escape mis à jour mars 22, 2014 • [afficher les modifications](#)
-  4 - Utilisation du plugin mis à jour mars 22, 2014 • [afficher les modifications](#)
-  3- Écriture de la logique du component en C++ mis à jour mars 22, 2014 • [afficher les modifications](#)
-  1- Compilation du plugin mis à jour mars 22, 2014 • [afficher les modifications](#)
-  Guide pour utilisation du plugin 3ds Max et pour la création des composants customs mis à jour mars 22, 2014 • [afficher les modifications](#)

Space contributors

- Mathieu Parent (il y a 20 jours)

Guide pour utilisation du plugin 3ds Max et pour la création des composants customs

- 1- Compilation du plugin
- 2 - Création de composants custom
- 3- Écriture de la logique du component en C++
- 4 - Utilisation du plugin
- 5 - Utilisation de 3ds Max

1- Compilation du plugin

La compilation du plugin se fait en 2 étapes:

1. Compilation de la DLL utilisée par 3ds Max
2. Compilation de la DLL de nodes customs qui est utilisé par le plugin

Ces deux étapes sont nécessaires au bon fonctionnement du plugin.

1.1 - Compilation du plugin

Avant de commencer:

Il est nécessaire de manuellement créer le dossier: "C:\Temp\3dsMaxPlugins" qui va contenir les fichiers du plugin.

Compilation du plugin

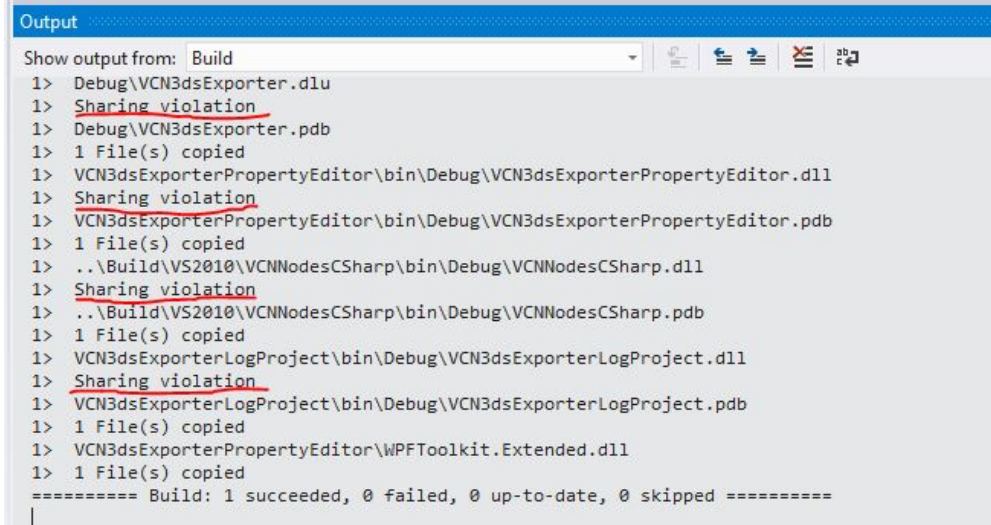
Le plugin est en fait représenté par une DLL (extension DLU). Les autres DLLs utilisées par le plugin doivent être dans le même dossier que de fichier DLU du plugin.

La solution du plugin est située à: "trunk\Vicuna\Exporter\VCN3dsExporter.sln".

Seul le mode Debug est supporté pour l'instant.

Lorsque la solution est générée, les fichiers nécessaires sont copiés dans le dossier utilisé pour stocker les DLLs du plugin.

NOTE: Si vous voulez rebuild la solution, il est nécessaire de fermer 3ds Max. Si vous voyez le message suivant:



```
Output
Show output from: Build
1> Debug\VCN3dsExporter.dlu
1> Sharing violation
1> Debug\VCN3dsExporter.pdb
1> 1 File(s) copied
1> VCN3dsExporterPropertyEditor\bin\Debug\VCN3dsExporterPropertyEditor.dll
1> Sharing violation
1> VCN3dsExporterPropertyEditor\bin\Debug\VCN3dsExporterPropertyEditor.pdb
1> 1 File(s) copied
1> ..\Build\VS2010\VCNNodesCSharp\bin\Debug\VCNNodesCSharp.dll
1> Sharing violation
1> ..\Build\VS2010\VCNNodesCSharp\bin\Debug\VCNNodesCSharp.pdb
1> 1 File(s) copied
1> VCN3dsExporterLogProject\bin\Debug\VCN3dsExporterLogProject.dll
1> Sharing violation
1> VCN3dsExporterLogProject\bin\Debug\VCN3dsExporterLogProject.pdb
1> 1 File(s) copied
1> VCN3dsExporterPropertyEditor\WPFToolkit.Extended.dll
1> 1 File(s) copied
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

Cela signifie que probablement que 3ds Max était encore ouvert et qu'il utilise les DLL que vous essayez de compiler. Il arrive aussi que Visual Studio bug et garde un lock dessus. Il faut alors le fermer et le réouvrir.

1.2 - Compilation des nodes custom

Compilation des nodes custom

Le but de l'éditeur est de permettre de créer rapidement et facilement de nouveaux types de composants qui seront par la suite utilisées dans l'engin.

C'est pour cela que les composants customs sont situées dans une solution et une DLL externe qui est chargée dynamiquement.

La solution qui contient ces nodes est située à: "trunk\Vicuna\ComponentGenerator\ComponentGenerator.sln".

Lorsque compilée, la solution copie les DLLs nécessaires dans le dossier du plugin de la même façon que la solution du plugin.

Contenu de la solution

Cette solution contient les composants customs (projet "VCNNodesCSharpCustom"), mais également le programme qui va générer automatiquement la représentation c++ des composants customs définis (projet "ComponentGenerator"). (Voir document sur la création de composants customs).

2 - Création de composants custom

Création de composants customs

Les nodes customs sont définies dans la solution "ComponentGenerator.sln" dans le projet "VCNNodesCSharpCustom" et dans le fichier "CustomComponents".

Plusieurs exemples de composants sont définis dans ce fichier et il est possible de s'en inspirer.

Écriture des composants

Règles à respecter

1. Les classes qui représentent des composants doivent dériver de la classe "BaseComponent"
2. Ils peuvent surcharger la représentation du component avec ceci:

Exemple de surcharge de nom d'affichage

```
public override string ComponentName { get { return "Physics Component"; } }
```

3. Seuls les champs public sont supportés (voir les attributs pour les valeurs sans accesseurs)
4. Les attributs suivants sont permis sur les champs:

Liste des Attributs pour els champs

```
[DefaultValue(0.0f)] // Définit la valeur par défaut du champ. Pour les champs de  
type non standards, il faut utiliser le constructeur par défaut (voir  
Testcomponent)  
[NoAccessors] // Ne fournira pas d'accesseurs dans le C++  
[Getter] // Seul un Getter sera défini en C++  
[Setter] // Seul un Setter sera défini en C++  
[VCN_Range(0.0f, 20.0f)] // Définit que la variable de type numérique peut avoir  
une valeur définie uniquement dans ce Range.
```

5. Il faut respecter les types de champs ci-dessous
6. Les méthodes ne sont pas acceptées et elles doivent être définies directement dans le .h et dans le .cpp

Type de champs disponibles

Seules les types suivants sont supportés dans l'éditeur:

1. int
2. uint
3. char
4. bool
5. string
6. float

7. double
8. Vector2
9. Vector3
10. Vector4
11. LuaTrigger
12. VCN_Color // Encore en développement
13. VCN_Node_Ref // Plannification future

Génération des composants

Afin de simplifier la tâche lors de la création de nouveaux composants, le processus a été automatisé au maximum et les fichiers .h et .cpp seront générés automatiquement afin de créer une représentation compatible avec ce qui est défini en C#.

Génération automatique

Le programme "component Generator" sert à créer une représentation C++ des composants définis dans le fichier "CustomComponents.cs" afin de les utiliser dans l'engin de jeu.

Ce programme prend en paramètre 2 variables:

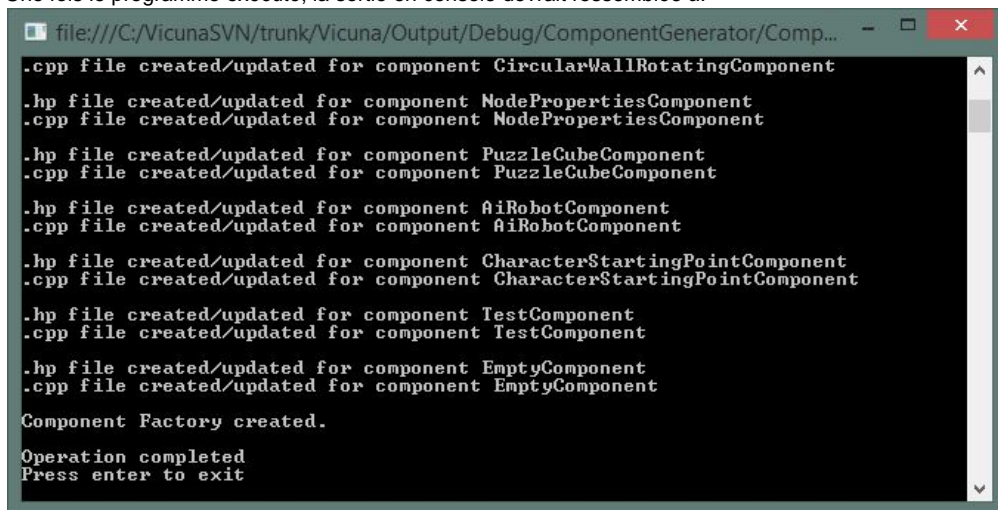
1. Path vers le dossier VCNNodes (trunk\Vicuna\Sources\VCNNodes). Ce path doit être en valeur absolue.
2. Path vers le dossier de builds de projets VS (trunk\Vicuna\Build\VS2010). Ce path doit être en valeur absolue.

Il est possible de mettre ces deux valeurs directement dans Visual Studio dans "Properties -> Debug -> Command Line Arguments" où le contenu devrait ressembler à:

Exemple de commande pour le ComponentGenerator

```
C:\VicunaSVN\trunk\Vicuna\Sources\VCNNodes C:\VicunaSVN\trunk\Vicuna\Build\VS2010
```

Une fois le programme exécuté, la sortie en console devrait ressembler à:



```
file:///C:/VicunaSVN/trunk/Vicuna/Output/Debug/ComponentGenerator/Comp...
.cpp file created/updated for component CircularWallRotatingComponent
.hp file created/updated for component NodePropertiesComponent
.cpp file created/updated for component NodePropertiesComponent
.hp file created/updated for component PuzzleCubeComponent
.cpp file created/updated for component PuzzleCubeComponent
.hp file created/updated for component AiRobotComponent
.cpp file created/updated for component AiRobotComponent
.hp file created/updated for component CharacterStartingPointComponent
.cpp file created/updated for component CharacterStartingPointComponent
.hp file created/updated for component TestComponent
.cpp file created/updated for component TestComponent
.hp file created/updated for component EmptyComponent
.cpp file created/updated for component EmptyComponent
Component Factory created.
Operation completed
Press enter to exit
```

De plus le programme va s'occuper d'aller modifier le fichier de projet Visual Studio VCNNodes afin d'ajouter les nouveaux types générés. C'est pour cela qu'il n'est jamais nécessaire d'ajouter les fichiers dans la solution "Vicuna.sln", **MAIS IL EST IMPORTANT DE LES AJOUTER SUR SVN!** Les fichiers générés seront situés dans "trunk\Vicuna\Sources\VCNNodes".

Lors de l'exécution du programme, un fichier "_backup" de chaque fichier qui va être modifié est créé afin d'éviter de perdre du code si le programme ne se comporte pas bien.

Code regions

Dans les fichiers générés par le ComponentGenerator, il y a des blocks définis par les tags:

Exemple de CodeRegion tag

```
/**CodeRegion
    // (votre code)
**EndCodeRegion
```

Si vous ajoutez du contenu à ces fichiers, ce sernier doit absolument être situé dans une des balises de ce type.

Vous ne devez pas ajouter ou enlever de balises

3- Écriture de la logique du component en C++

Écriture de la logique en C++

Lorsque générées, les classes C++ ne contiennent que les champs définis en C# et leurs accesseurs.

Méthodes à implémenter

Initialise	Méthode appelée lorsque le component est créé. À ce stade, la validité de la Node parent est garantie. Cependant, d'autres composants peuvent ne pas avoir été initialisés encore.
Update	Méthode appelée à chaque "tick" du jeu. le paramètre est le temps depuis le dernier "tick" en secondes.
SetAttribute	Méthode auto-générée. NE PAS MODIFIER
Copy	Méthode auto-générée. NE PAS MODIFIER
Constructeur par défaut	Dans le constructeur le contenu interne n'est pas garanti de rester le même. Les valeurs assignées dans le constructeur ne resteront probablement pas à l'exécution car elles seront écrasées par la méthode SetAttribute lors du chargement du fichier XML

Ajout de méthodes

Vous pouvez ajouter les méthodes que vous voulez tant que ces dernières sont définies et implémentées dans des Code Region tags

Règles à respecter

- Toujours ajouter votre code dans les "Code Region" tags
- Faites attention à vos includes. Le projet VCNNodes ne connaît pas tout les projets
- Pour accéder à la Node qui contient ce Component, vous pouvez utiliser GetOwner(), mais pas dans le constructeur.
- Pour accéder à un autre component sur la même Node, vous pouvez utiliser la méthode GetOwner()->GetComponent<typeDecomponentAChercher>(). ATTENTION! Le résultats sera NULL si la Node ne contient pas ce component.

4 - Utilisation du plugin

Version de 3ds Max requise

Afin de pouvoir utiliser le plugin dans 3ds Max, il vous faut absolument la version 2013 32 bits.

Configuration

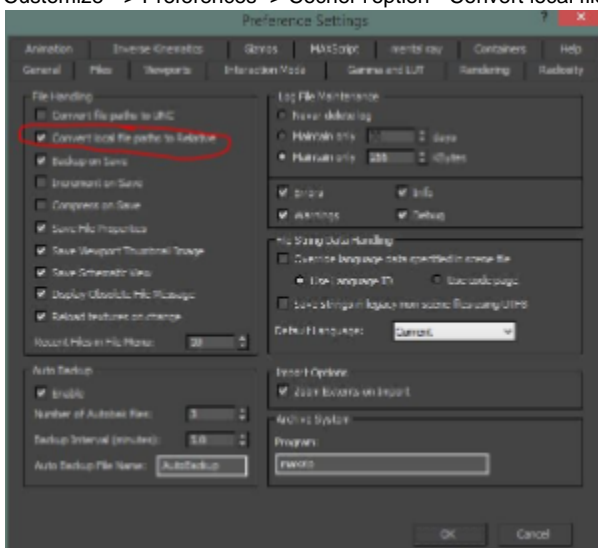
Une fois le plugin compilé, veuillez suivre les étapes suivantes afin de charger le plugin dans 3ds Max (fourni par le prof):
[Vicuna_Exporter_3dsmax_2012.pdf](#)

Dans 3ds Max

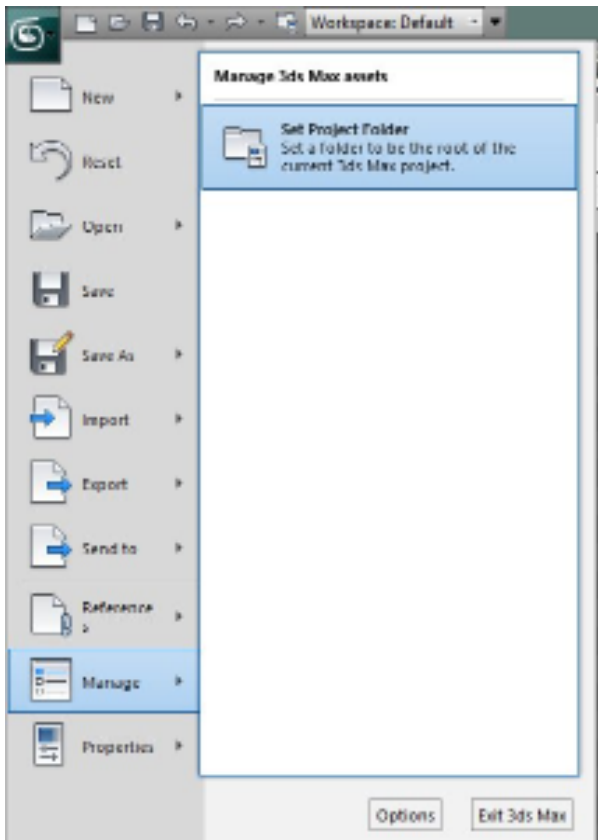
Une configuration hyper importante à effectuer est de mettre les paths des fichiers externes relatifs car sinon il y aura des problèmes de textures.

Il faut donc aller ici:

Customize --> Preferences --> Cocher l'option " Convert local file paths to Relative"



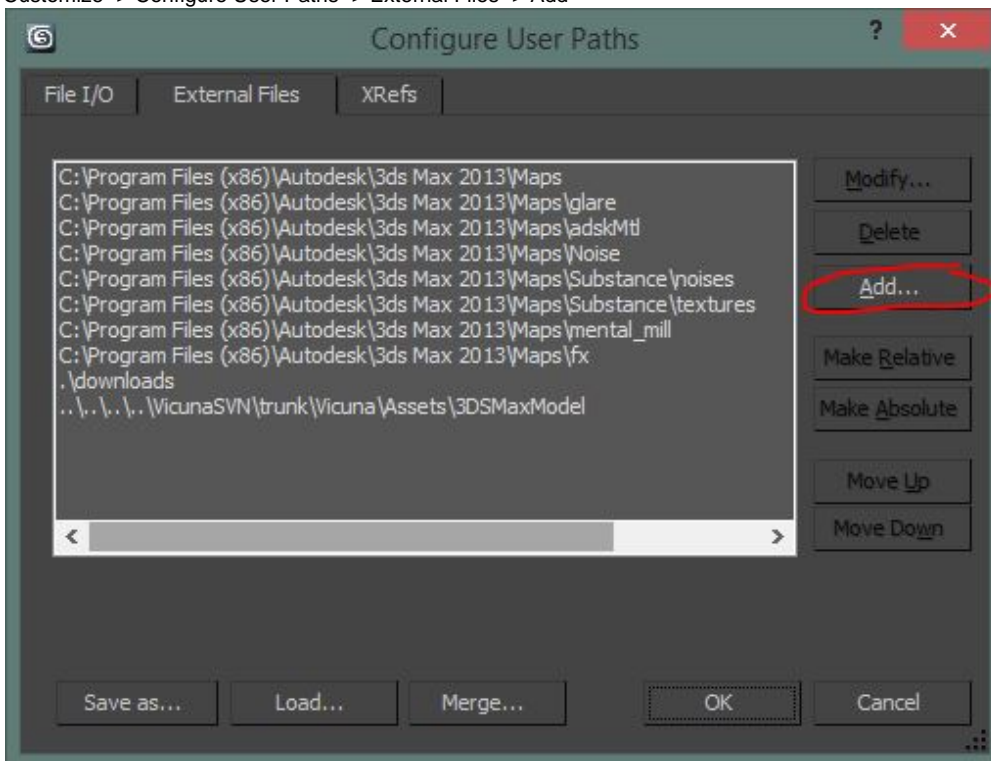
Ensuite, il faut aller ici:



et sélectionner le path "trunk\Vicuna\Assets\3DSMaxModel"

Ensuite, il faut aller ici:

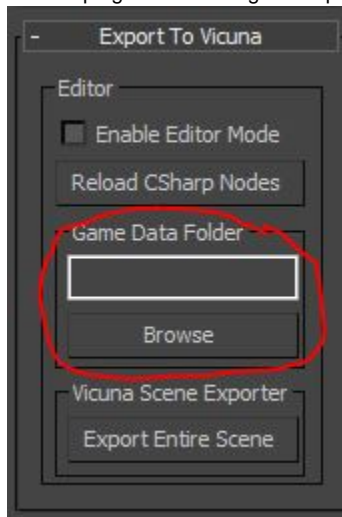
Customize -> Configure User Paths -> External Files -> Add



et aller chercher le dossier "trunk\Vicuna\Assets\3DSMaxModel".

Dans le plugin

Dans le plugin il faut configurer le path du dossier "trunk\Vicuna\Game\Data" ici:

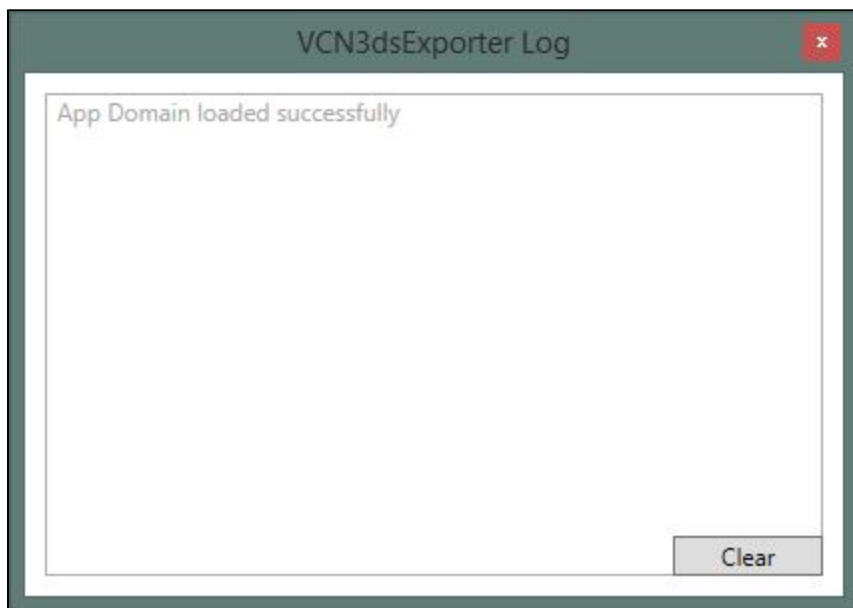


Cette préférence devrait être sauvegardée même après la fermeture du logiciel.

Utilisation du plugin

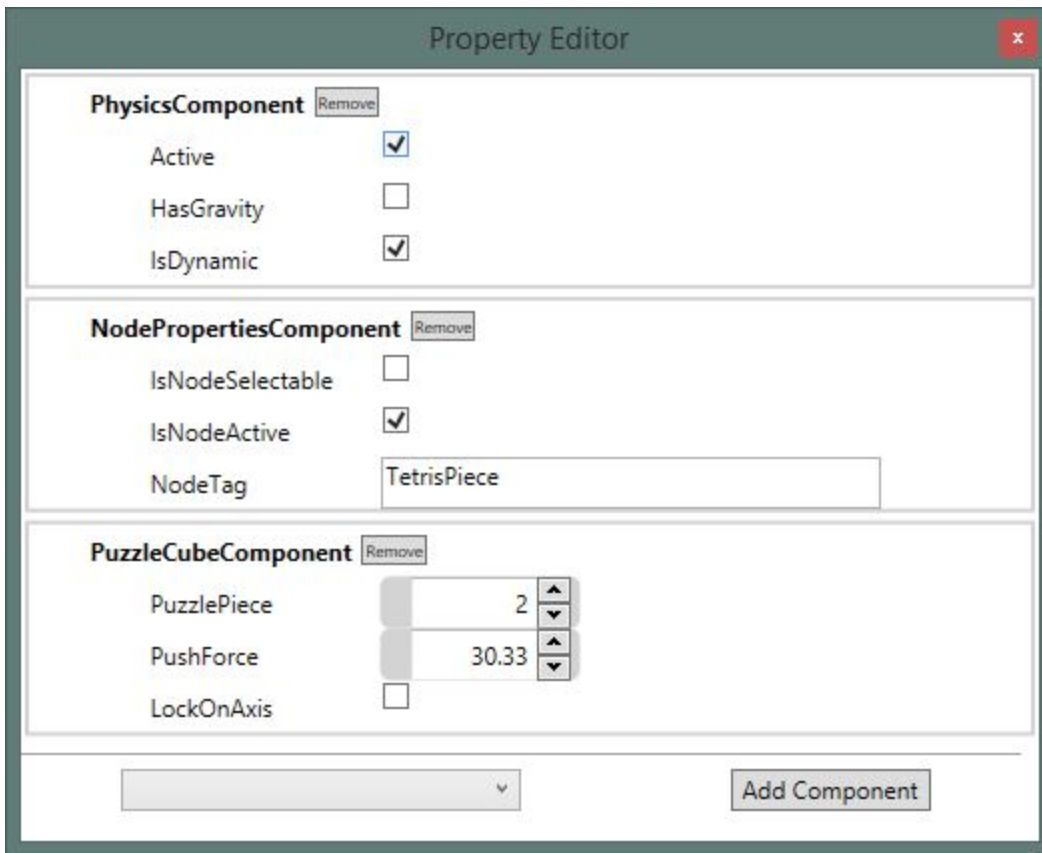
Fenêtres

Fenêtre de Log



La fenêtre de Log va afficher les informations nécessaires au debug et afficher les erreurs qui surviennent. Il est donc important d'y jeter un coup d'oeil fréquemment. Cette fenêtre ne peut pas être fermée.

Fenêtre de l'Éditeur



La fenêtre de l'éditeur sert à afficher et à modifier les composants de chaque Node.

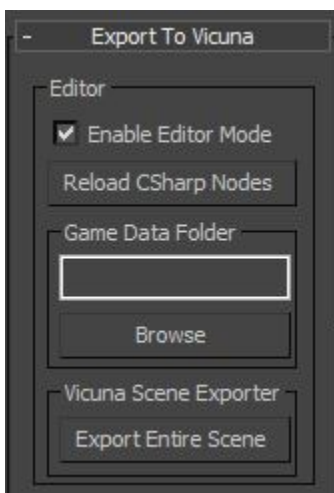
Lorsqu'une Node de la scène est sélectionnée (seulement une), le contenu de la fenêtre représente les composants de cette Node.

L'éditeur ne supporte pas l'édition de plusieurs objets en même temps.

Le Combobox en bas sert à ajouter les différents Components à la Node sélectionnée.

Une Node ne peut avoir qu'une fois un certain type de component d'un certain type. Par exemple, une Node ne pourrait pas avoir 2 component de type PhysicsComponent.

Rollup dans les Tools de 3ds Max



Il s'agit du point de départ du plugin.

Pour ouvrir la fenêtre de l'éditeur, il faut cocher le checkbox à côté de "Enable Editor Mode".

Le bouton "Export Entire Scene" est le bouton à utiliser afin d'exporter la scène en XML et de pouvoir la charger dans Vicuna.

Pendant l'export, deux progressbars s'affichent afin de montrer la progression de la tâche.

Une fois terminé, une fenêtre de Tortoise SVN devrait ouvrir avec les fichiers qui ont été créés par l'exporter et qui doivent être Add sur SVN. Si la liste contient des fichiers, il est fortement recommandé de les "Add" en appuyant sur "OK".

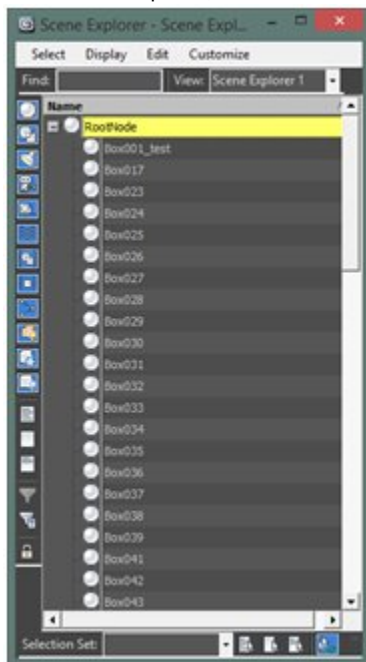
5 - Utilisation de 3ds Max

Utilisation de 3ds Max

- Appuyez sur "P" pour avoir la vue perspective
- Click middle mouse + drag pour pan
- Alt + Click middle mouse pour orbit
- "M" pour le material pannel
- Tool -> New Scene Explorer pour avoir une vue hiérarchique des nodes de la scène
- Alt + W pour maximize un viewport

Utilisation dans le cadre du projet

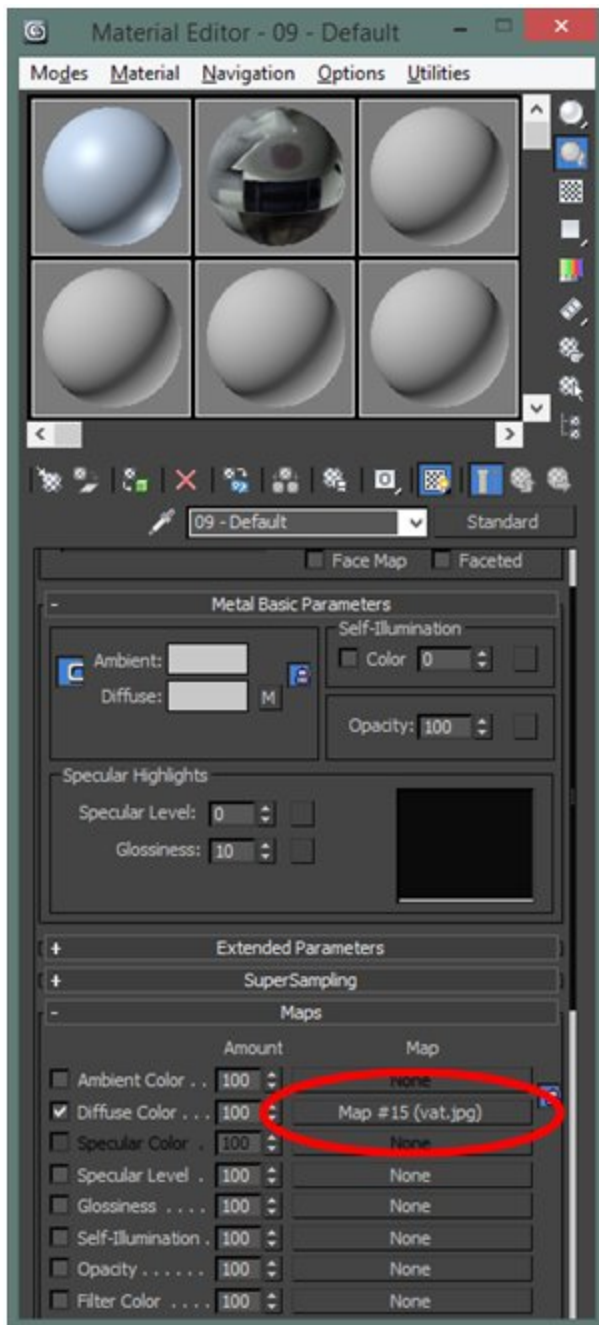
- Pour utiliser l'exporter, il faut absolument mettre votre scène dans une "node parent" comme ça:



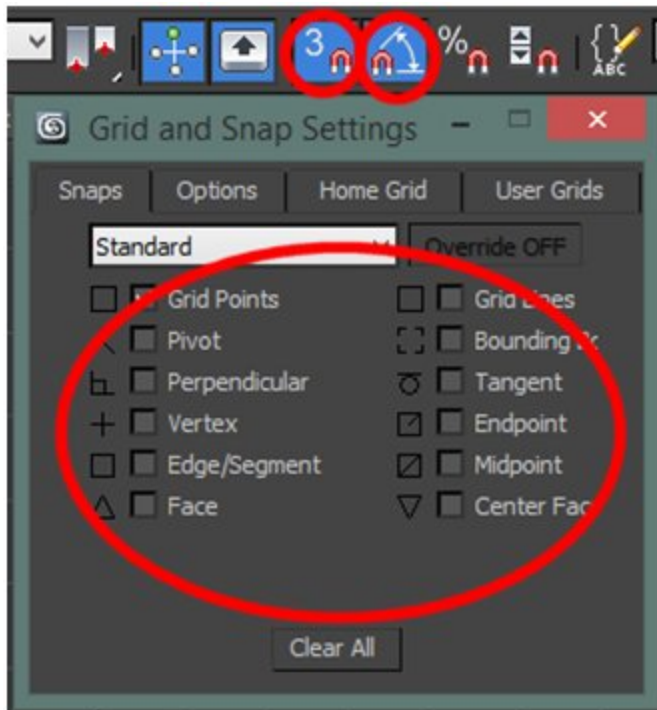
- Avant de pouvoir export, il faut mettre le path du dossier "Data" du projet (voir la section Description du Toolbar).
- Ne JAMAIS mettre deux surfaces une sur l'autre, sinon ça va faire ça:



- Mettre ses références de textures en path relatifs. Voir: http://www.maxforums.org/threads/absolute_paths_texture_maps/0001.aspx
- Les textures ne sont pas "attachées" dans le fichier .max de la scène. Il faut donc les ajouter manuellement sur SVN.
- L'exporter ne supporte qu'une texture alors les autres seront ignorées. Il faut la mettre dans la "slot" Diffuse comme ça:

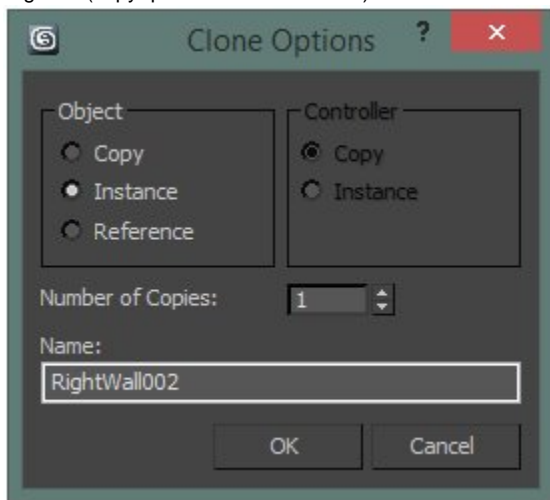


- Il faut utiliser les formats suivant pour les textures: jpg, png (si transparence). Les autres formats ne sont pas supportés
- Utilisez le "grid" de 3ds Max pour positionner des objets (click droit sur l'icône pour configurer):



Mettez les options qui vont le mieux pour vous

- Pour convertir un objet de base en polygone et en éditer les sommets et arêtes, sélectionnez l'objet et faites un click droit dessus (dans le viewport) et faites Convert To -> Editable Poly
- Une fois l'objet converti en Editable Poly, il est possible de passer au mode pour modifier ses vertex en appuyant sur "1", ses arêtes en appuyant sur "2", ses bordures en appuyant sur "3" et ses faces en appuyant sur "4".
- ***** IMPORTANT** Vous pouvez faire des "instances" de nodes que vous avez déjà. Pour le faire, faites Shift + Déplacer une node avec le gizmo (copy+paste fonctionne aussi). Ce menu devrait ouvrir:



Sélectionnez l'option "Instance" pour créer un instance.

L'avantage d'utiliser une instance est que si la node de départ ou l'instance est modifiée, tout les nodes qui se référencent de cette façon le seront. Il ne sera alors pas nécessaire de modifier toutes les toilettes par exemple. En en modifiant une, les autres vont aussi être changées. Il y a aussi le fait que ça va prendre moins de place en mémoire puisque chaque objet n'y sera qu'une fois et sera utilisé pour rendre toutes les toilettes à des positions différentes.

How-To Articles

Ajouter un article de conseil

Title	Creator	Modified
Guide pour utilisation du plugin 3ds Max et pour la création des composants customs	Mathieu Parent	mars 22, 2014