Tutoriel sur la création de textures et matériaux pour Source SDK

Par Christian Clavet

Sommaire

Introduction

Ce tutoriel à pour but de vous guider à travers le Source SDK à fin que vous puissiez réaliser vos propres textures et matériaux.

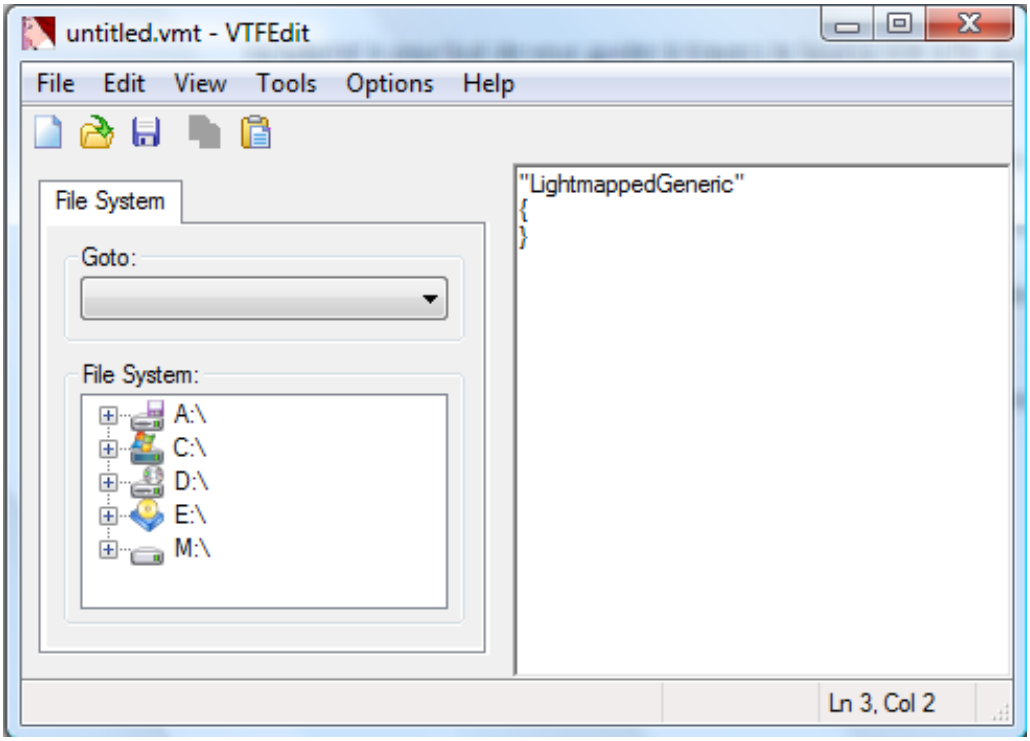
Source SDK utilise 2 types de fichiers principaux lors de la création de matériaux et de textures :

Fichiers VMT (Valve Material Type).
Est un fichier texte décrivant le matériel utilisé

Fichiers VTF (Valve Texture Format).
Est un format spécial pour contenir une image de texture et autres utilisé par l'engin et décrit dans le fichier de matériel VMT.

Outil recommandé pour la création de matériaux et textures pour source SDK :

VTFEdit : http://nemesis.thewavelength.net/index.php?c=178



Cet outil est bien adapté pour la création de matériaux et de textures pour Source SDK. Presque toutes les options offertes par Source SDK sont supportés.

Création d'une texture à l'aide de VTFEdit

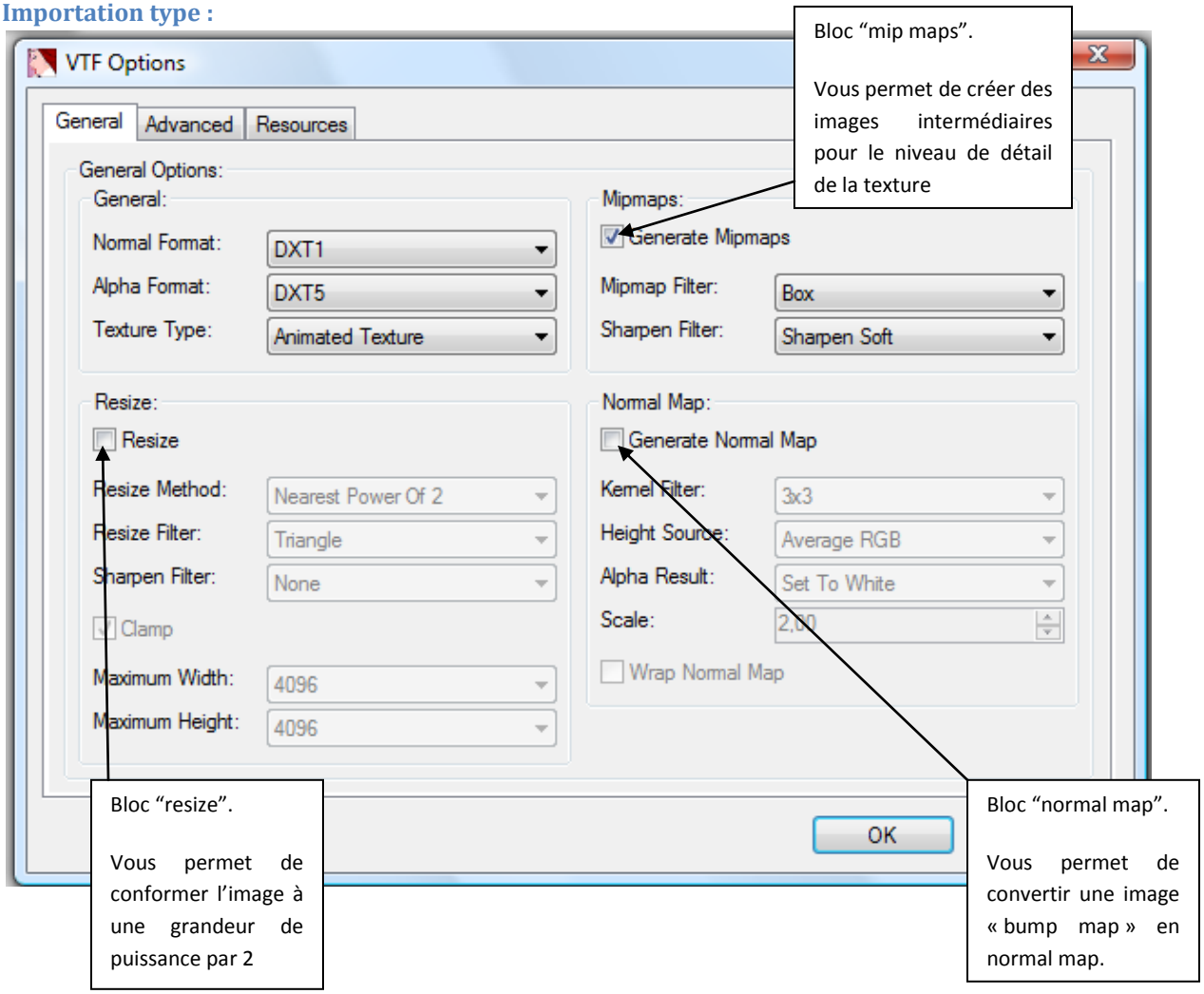
Ouvrez le menu File et sélectionnez IMPORT. VTFEdit supporte plusieurs formats d'images tels que :

BMP, DDS, GIF, JPEG, PNG et TGA.

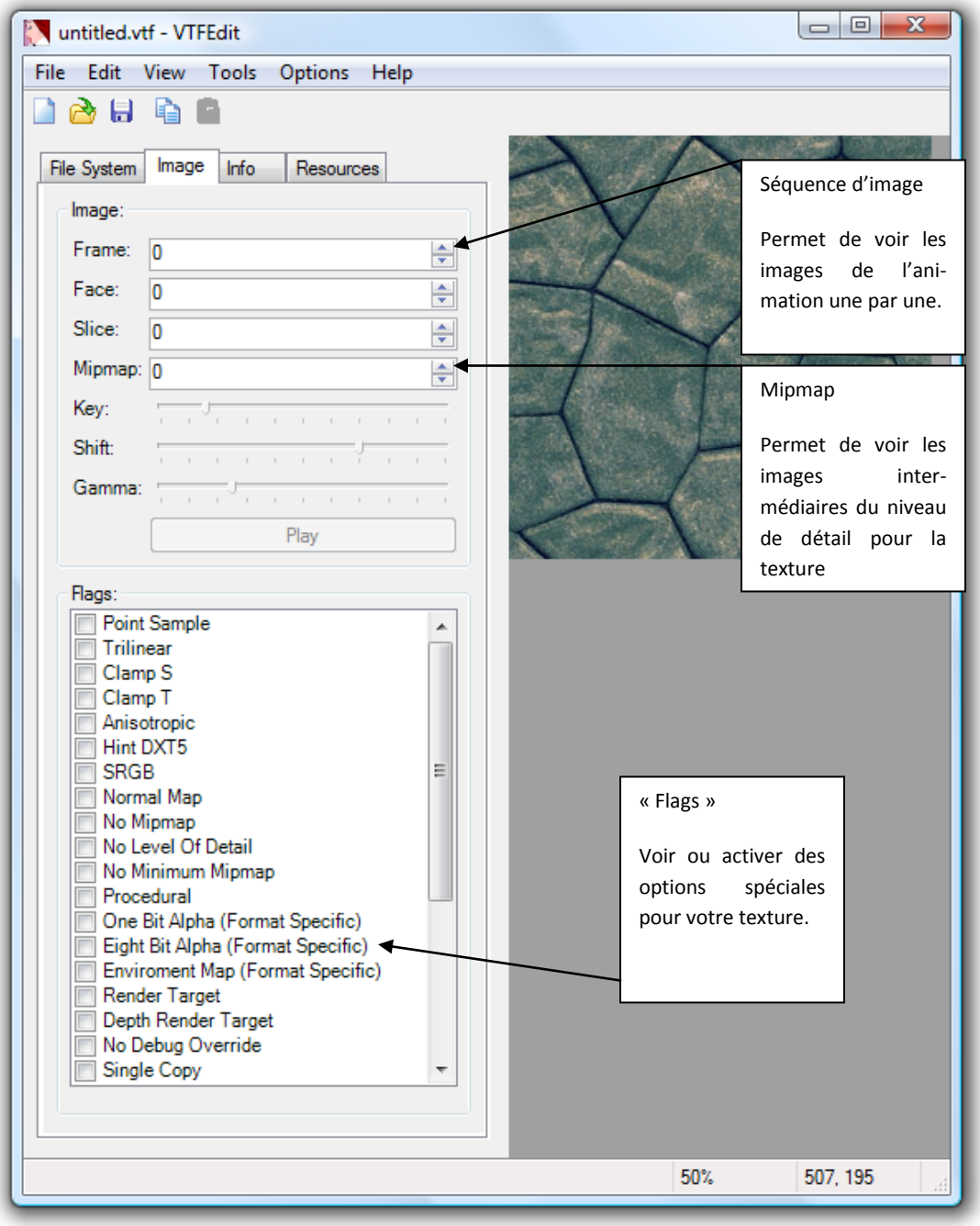
VTFedit peut importer aussi des images en séquences pour faire des animations dans la texture. (image001, images002, image003, etc.) L'importation se fait alors par multi-sélection.

VTFedit supporte les profondeurs de 32bit dans les images BMP et TGA ce qui peut permettre d'utiliser la couche alpha pour la transparence ou d'autres effets.

Importation type :



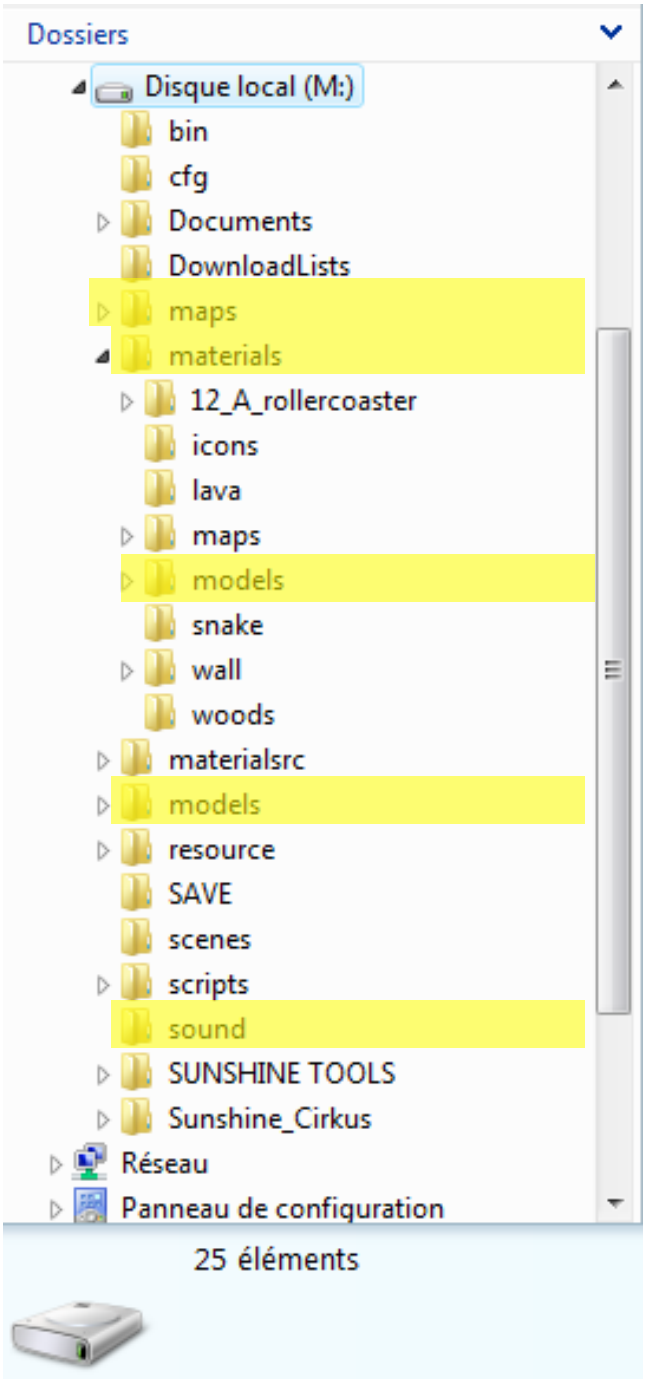
Exemple d'une texture importée :



Sauvegarde de la texture VTF :

Je recommande fortement de sauvegarder votre texture dans le dossier final, et d'ensuite si besoin est d'en faire un sauvegarde d'urgence. VTF Edit à un outil très convivial pour créer des scripts de matériel mais votre texture doit être avant tout dans le bon dossier si le script doit être valide.

Structure des dossiers d'un jeu Source SDK



Les dossiers important sont marquées en jaune.

Je recommande d'utiliser des dossiers pour les types de texture (modèles ou textures pour l'éditeur de mondes Hammer, décalques. Etc.)

Pour plus de clarté et tout dépendant du projet en cours, il serait bien aussi de placer ces textures dans le même dossier, si par exemple, ces textures font parti d'un même niveau de jeu.

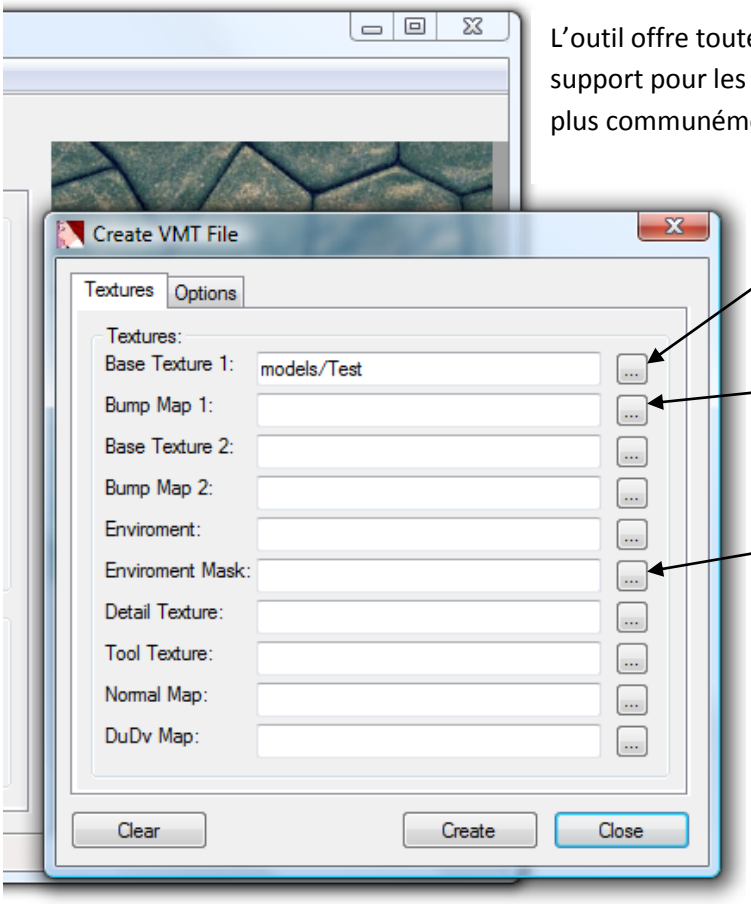
Dans un script de matériel VMT, le chemin de la texture doit être toujours référencé à partir de la base de la hiérarchie.

Exemple : Une texture d'un modèle qui s'appelle `trash.vtf` devrait être référencée comme suit : `models/trash.vtf` dans votre fichier de script VMT, même si les deux fichiers sont dans le même répertoire.

Si votre fichier VTF est déjà à la bonne place dans les dossiers, l'outil d'assistance de création de script de matériel VMT fera le référencement adéquat.

Note : Par soucis de simplification, la racine du jeu est placée sur le lecteur M :. Un jeu STEAM est normalement placé dans votre dossier STEAM dans un sous-répertoire.

Utilisation de l'outil d'assistance de script de matériel VMT

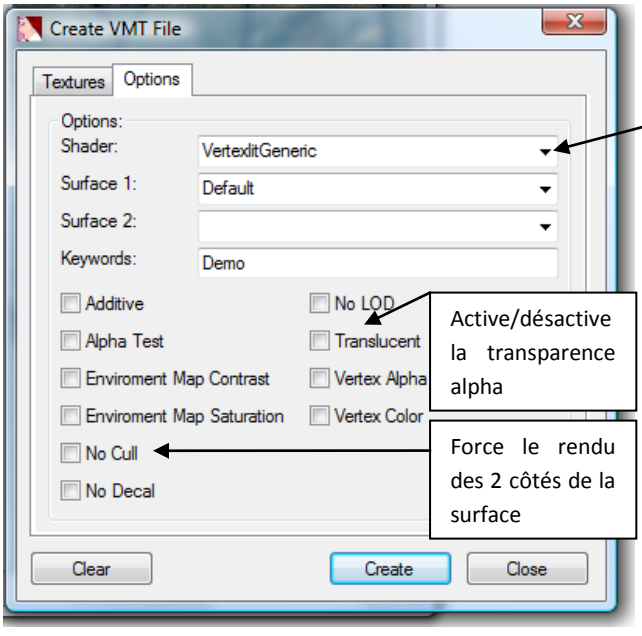


L'outil offre toute une panoplie de type de support pour les textures. Voici les textures les plus communément utilisés :

Texture de diffusion (diffuse map)

Texture « bump » Image de type : normal map

Texture de spécularité (specular map)



L'onglet options offre des réglages spécifiques au matériel définis :

Type de "shader". Doit être appliqué spécifiquement à la tâche attribuée au matériel.

Les shaders les plus utilisés sont :

`VertexlitGeneric` -> Modèles

`LightmappedGeneric` -> Texture Hammer

`Decal` -> Décalques

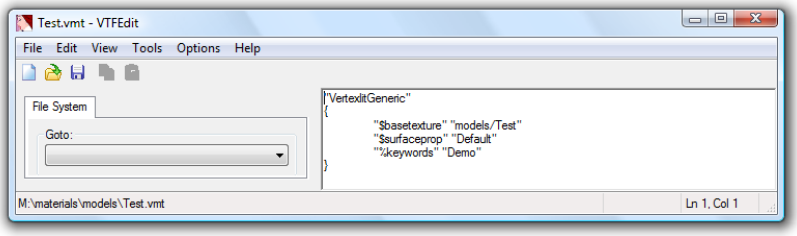
`UnlitGeneric` -> GUI_overlay

`Sprite` -> Effet (fumée, feu, particules)

Active/désactive la transparence alpha

Force le rendu des 2 côtés de la surface

Script VMT pour une simple texture diffuse sur un modèle



Voici le script généré plus haut par l'outil (create VMT) :

```
"VertexlitGeneric"
{
        "$basetexture" "models/Test"
        "$surfaceprop" "Default"
        "%keywords" "Demo"
}
```

Ce VMT référence la texture qui sera dans `materials/models /` et la texture s'appelle `Test.vtf`
La propriété de la surface est celle par défaut. Cette surface peut avoir certains attributs lors des impacts et des sons produits. Vérifiez dans VTFEdit pour voir tous les attributs disponibles.

VertexlitGeneric définit ce matériel comme étant un matériel utilisé spécifiquement pour un modèle et le mot *keyword* définit un mot clé pour le matériel qui peut être retrouvé par les éditeurs en mode recherche.

Script VMT pour une texture diffuse + normal map + specular map

```
"VertexLitGeneric"
{
        "$basetexture" "Chemin vers votre texture diffuse"
        "$surfaceprop" "Default"
        "$bumpmap" "Chemin vers votre texture normap map + specular"
        "$normalmapalphaenvmapmask" 1
        "$envmap" "env_cubemap"
}
```

Explications:

Ce script utilise une texture diffuse (image 24bit) et une texture de 32bit qui contient l'image des « normals » et l'image de spécularité. Explication pour les autres commandes :

$normalmapalphaenvmapmask 1 :

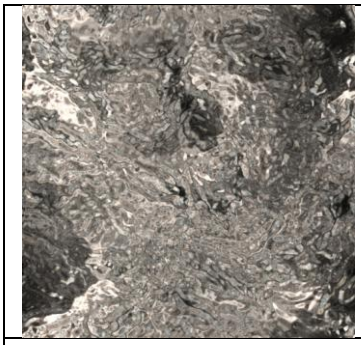
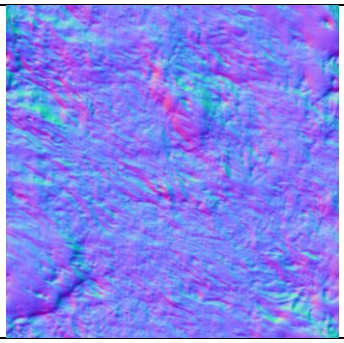
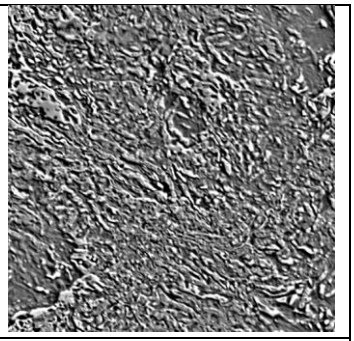
Définit la texture de spécularité (specular map) faisant partie de la couche alpha de votre texture de « normal »

$envmap env_cubemap :

Définis comme spécularité, l'environnement provenant des entités « cubemaps » placés par les level designers. Cette entité donne une image de réflexion de l'environnement autour de l'entité.

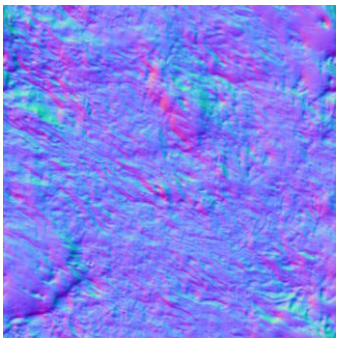
$envmaptint

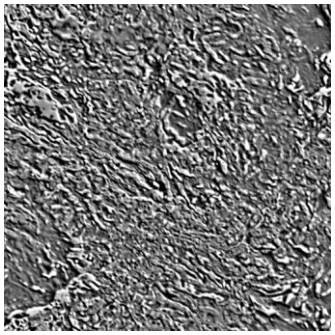
Les images :

		
Texture de diffusion	Texture normal map	Texture spéculaire

Composition dans Photoshop :

Texture de diffusion + no alpha = `image.tga` (24 bit)





Texture normal (RGB) + Texture spéculaire (ALPHA) = `image_nrm.tga` (32bit)

Script VMT pour une texture diffuse + normal map

```
"VertexLitGeneric"
{
	"$basetexture" "Chemin vers votre texture diffuse"
	"$surfaceprop" "Default"
	"$bumpmap" "Chemin vers votre texture normap map"

}
```

Script VMT pour une texture diffuse + specular map (pas de normal)

```
"VertexLitGeneric"
{
	"$basetexture" "Chemin vers votre texture diffuse"
	"$surfaceprop" "Default"
	"$envmap" "env_cubemap"
	"$envmapmask" " Chemin vers votre texture de spécularité"

}
```

Script VMT pour une texture diffuse + rendu forcé des 2 côtés de la face

```
"VertexLitGeneric"
{
	"$basetexture" "Chemin vers votre texture diffuse"
	"$surfaceprop" "Default"
	"$nocull" "1"

}
```

La commande **`$nocull`** désactive la fonction de « face culling » et donc le rendu de la surface se fait maintenant des 2 côtés.

À n'utiliser qu'à de rares occasions car le modèle sera rendu des 2 côtés et on perdra une optimisation, donc un ralentissement possible pour des modèles lourds en polygones.

On recommande donc aux modélisateurs à construire les 2 deux faces du modèle qui seront vus par l'observateur et ainsi les faces qui ne seront pas visibles ne seront pas rendues.

Les « skins » des modèles

Une option intéressante permet de créer plusieurs textures appliquées à un même modèle. Ceci permet par exemple d'avoir plusieurs conserves avec des étiquettes différentes et utiliser seulement le même modèle.

Ces commandes doivent être ajoutées directement dans le script de compilation .QC du modèle. Les fichiers référencés dans le modèle seront tous les matériaux qui composent le modèle et les autres versions.

Exemple simple

Voici un exemple de code à insérer dans votre fichier .QC de nodèle:

```
$texturegroup "rockcliff_cluster01"
{
        { "rockcliff02a" } // skin principale
        { "rockcliff02b" } // skin numéro 2
        { "rockcliff02c" } // skin numéro 3
}
```

Le modèle s'appelle `rockcliff_cluster01` et la premiere texture appliquée est `rockcliff02a.vmt`

Exemple plus complexe :

Si votre modèle est composé de plus que un matériel faites comme ceci :

```
$texturegroup "conserve"
{
        { "conserve_metal" "conserve_etiquette1>" } // Skin principale
        { "conserve_metal" "conserve_etiquette2>" } // Skin numéro 2
}
```

Donc si vous regardez dans l'exemple plus haut; seulement l'étiquette du modèle change. Mais nous devons toujours spécifier tous les matériaux qui composent notre modèle.