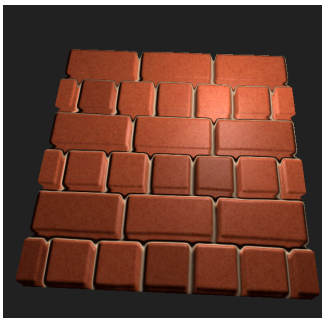
T.P. Textures et Shaders avancés

Le but du TP est d'améliorer 3 programmes, qui réalisent des rendus d'objets 3D, et en utilisant les shaders de manière plus évolué afin de visualiser des rendus plus réalistes.

Sources : https://github.com/Modelisation5ETI/TP8_Texture



Rendu du programme parallax

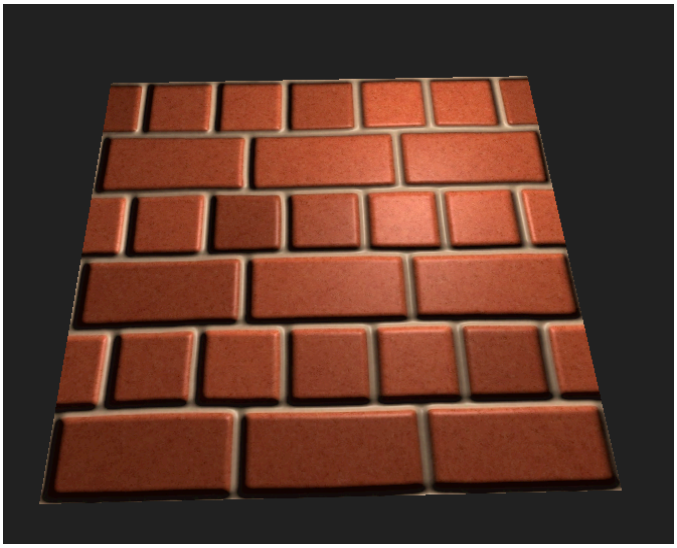
Rendu du programme bidon

Rendu du programme caillou

I. Parallax

Le programme parallax met en évidence les différentes sources d'informations présentes dans une image dans le but d'ajouter du réalisme à la texture d'un maillage.
Pour cela, on implémente deux techniques modifiant les attributs associés à chaque vertex :

Normal Mapping : Le normal mapping consiste à modifier la normale associée à chaque vertex en lisant l'information contenue dans une carte des normales passée au shader en tant que texture.



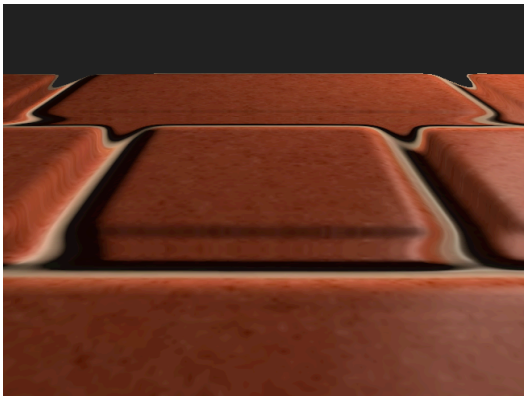
Effet de relief : Normal Mapping

Parallax Mapping : Le parallax mapping vient modifier les coordonnées de textures utilisées pour les couleurs et les normales afin d'améliorer encore le rendu. Chaque coordonnée est modifiée suivant le vecteur view_dir (qui est dirigé de la view_position vers la position du pixel courant), multipliée par une valeur de hauteur lue dans une carte des hauteurs. La projection sur dans le plan du vecteur view_dir résultant permet d'obtenir le décalage à appliquer aux

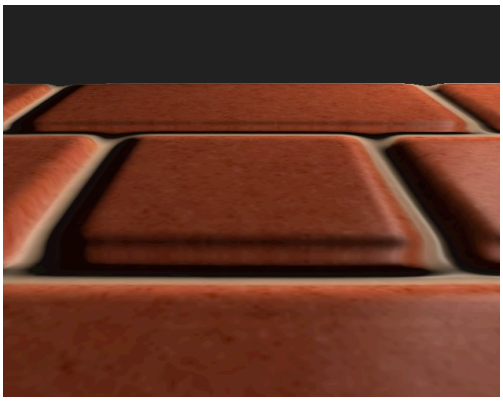
coordonnées.

```
// Get height from heightmap
float texture_height = texture(height_tex, tex_coords).r;
vec3 view_dir_proj =  -view_dir * height_scale * texture_height;
// Shift texture coordinates
tex_coords = tex_coords + view_dir_proj.xy/view_dir.z;
```

La normalisation de view_dir par sa coordonnée z constitue l'implémentation d'origine du parallax mapping et doit améliorer les résultats quand la vue est rasante. En effet , si le vecteur view_dir est parallèle au plan, sa coordonée z vaut 0. En divisant le vecteur par celle-ci on applique alors un plus gros décalage aux coordonnées de textures, et donc un effet de relief accru.
Cependant, dans notre cas, le déplacement limite autorisé avant de voir apparaître les premiers effets de distorsion, n'est pas très important. Le parallax mapping with offset limitting, qui n'effectue pas cette normalisation permet d'obtenir de meilleur résultat si l'échelle de hauteur est grande et que la vue est rasante :

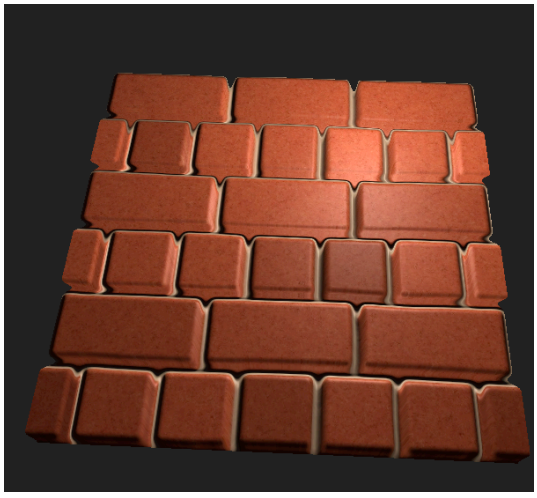


Parallax Mapping



Parallax Mapping with limit offset

En couplant le normal et le parallax mapping, on obtient le résultat suivant :



Parallax et Normal Mapping

L'échelle de hauteur peut être ajustée avec les touches Haut et Bas du clavier. La barre d'espace désactive l'effet de parallax.

La mise en place des techniques présentées ci-dessus permet d'améliorer grandement le réalisme d'une scène en ajoutant du relief aux matériaux utilisés. D'autres techniques de cartographie du relief peuvent également être implémenté. Sur le même principe que le parallax mapping, on aurait pu mettre en place un algorithme itératif utilisant un cône pour trouver l'intersection entre le relief et le view_dir. Cette technique permet d'augmenter le niveau de détail du relief: http://http.developer.nvidia.com/GPUGems3/gpugems3_ch18.html

II. Bidon

Le programme bidon à pour but de représenter un bidon de lessive rouillé de manière réaliste. Pour cela nous utiliserons plusieurs textures (3 textures chargées) à appliquer suivant une carte de texture (4ème texture chargée).

Dans un premier temps, on modifie le shader bidon.frag afin d'utiliser, selon la couleur de carte de texture, une certaine texture chargée.

```
// Get different texture colors
 vec4 col1 = texture(decal_tex, vec2(0.65 - vf_tex_coords.x, vf_tex_coords.y));
 vec4 col2 = texture(rust_tex, vf_tex_coords);
 vec4 col3 = texture(brushed_tex, vf_tex_coords);

// Mix texture according to a map
 vec4 map = texture(shader_map_tex, vf_tex_coords);

 //Compute textureColor according to the texture map
 vec4 textureColor = map.x * col1 + map.y * col3 + map.z * col2;

 //Set textureColor
 color = textureColor;
```

Rendu :

Dans un second temps, nous ajoutons une illumination différente selon les zones. Dans la figure suivante, nous ajoutons une illumination de Phong sur la couleur de la texture affichant l'étiquette du bidon. Le rendu semble beaucoup plus réaliste.

Bidon de lessive en métal rouillé

Pour plus de réalisme, il serai nécessaire d'ajouter une illumination de Phong sur chaque texture, puis jouer sur les paramètres d'illumination selon la texture utilisée (matériau).

III. Caillou

Le programme Caillou a pour but de générer un caillou des plus réalistes. Le maillage constitue un ellipsoïde sans coordonnées de texture. Une manière intéressante de contourner ce problème pour les maillages simples est d'utiliser les coordonnées spatiales du maillage. Cependant, il est nécessaire d'adapter les composantes retenues comme coordonnées de texture en fonction de la face à laquelle elles appartiennent. Prenons l'exemple d'un cube. Sur chaque face, les coordonnées de texture correspondent au coordonnées spatiales des vertex exprimées dans le repaire local de la face. Pour obtenir ces coordonnées, on projette alors les vertex dans dans le plan parallèle à la face courante

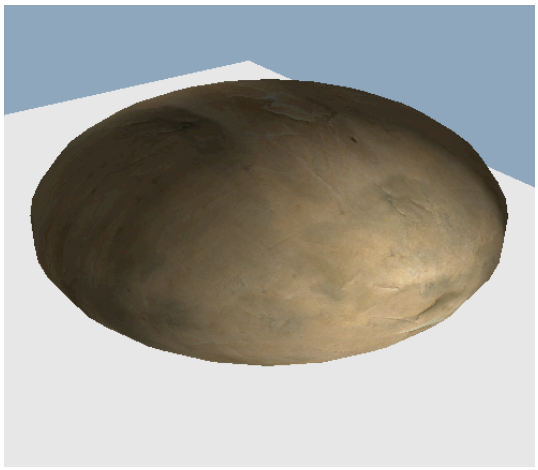
```
vec4 base_color_x = vec4(texture(tex, vf_position.yz)); //Projection sur X
vec4 base_color_y = vec4(texture(tex, vf_position.zx)); //Projection sur Y
vec4 base_color_z = vec4(texture(tex, vf_position.xy)); //Projection sur Z
```

La forme actuelle étant un ellipsoïde, on remarque alors la jonction entre les zones de projection différentes. Il est donc nécessaire de mélanger les différentes textures à l'aide de la valeur des normales :

```
vec4 textureColor = vf_normal.x * base_color_x + vf_normal.y * base_color_y + vf_normal.z * base_color_z;
```

Ainsi les vertices dont les normales sont perpendiculaires à une face ne gardent que la couleur associée à celle-ci.

Cette approche permet de texturer de façon très réaliste le caillou :



Un beau galet

Afin d'appliquer un matériau encore plus réaliste au caillou, nous avons essayer d'implémenter l'effet d'humidité proposé par NaughtyDog p.59 de
http://dindinx.net/cpe/eti5/NaughtyDog_TechArt_Final.pdf

Cette technique modifie les normales et les paramètres de couleurs en fonction d'une carte d'humidité. Ne possédant qu'une partie des paramètres utilisés dans la présentation, nous nous sommes concentrés sur la modification :

- des normales
- des couleurs de textures
- du paramètre spéculaire

La carte d'humidité est grossièrement modélisée à l'aide d'une fonction cosinus selon l'axe Y qui associé une valeur d'humidité à chaque vertex. On l'utilise pour restreindre la valeur des paramètres.

En jouant sur la période du cosinus, nous avons pu mettre en évidence trois zones différentes sur la texture :

Ces zones ne correspondaient pas exactement à nos attentes mais nous donnaient une impression de caillou semi-immergé. Nous avons donc pris l'initiative d'ajouter un effet de bruit caustique dans la zone immergée, dans le but d'augmenter encore plus les effets liés à la présence d'eau.
Le bruit caustique est animé à l'aide de 32 textures qui se succèdent sur le matériau.
Le plan constituant initialement le sol à été élevé et rendu transparent pour marqué le niveau de l'eau.

Un caillou qui flotte

Il serait possible d'améliorer le rendu en utilisant les textures caustiques pour modifier la luminosité et non les couleurs directement. Il serait également intéressant de se consacrer à la réalisation simple d'une surface d'eau plus réaliste en utilisant par exemple le déplacement de textures comme dans l'article suivant : http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter18.html