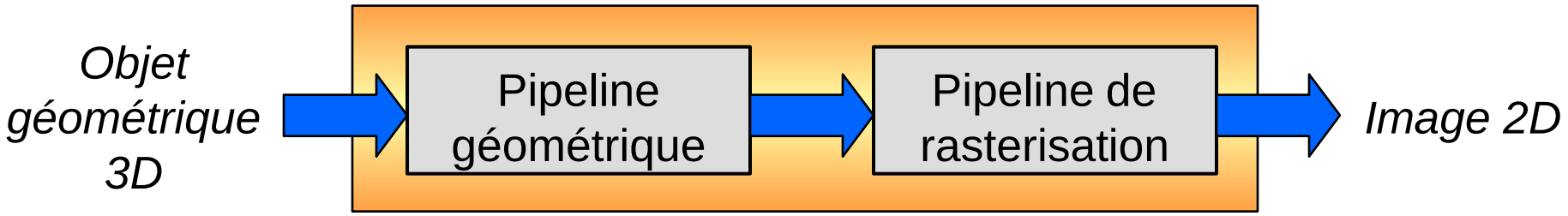
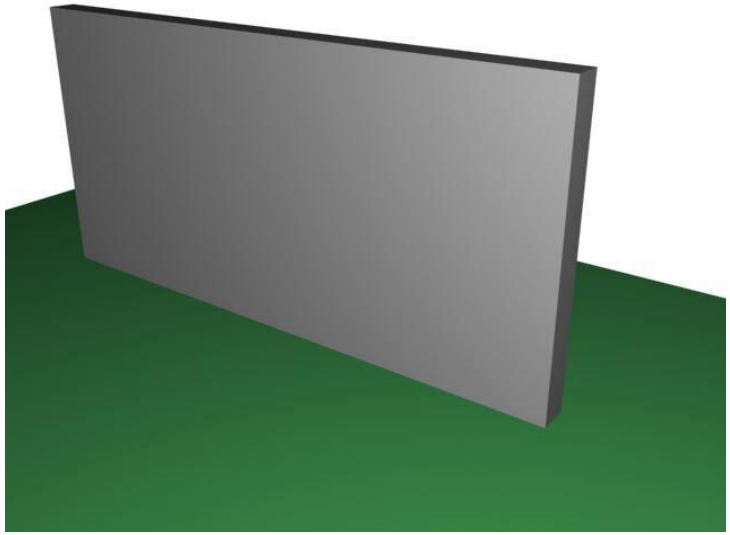
Module: Infographie
3ème licence LMD

Les textutes sous OpenGL

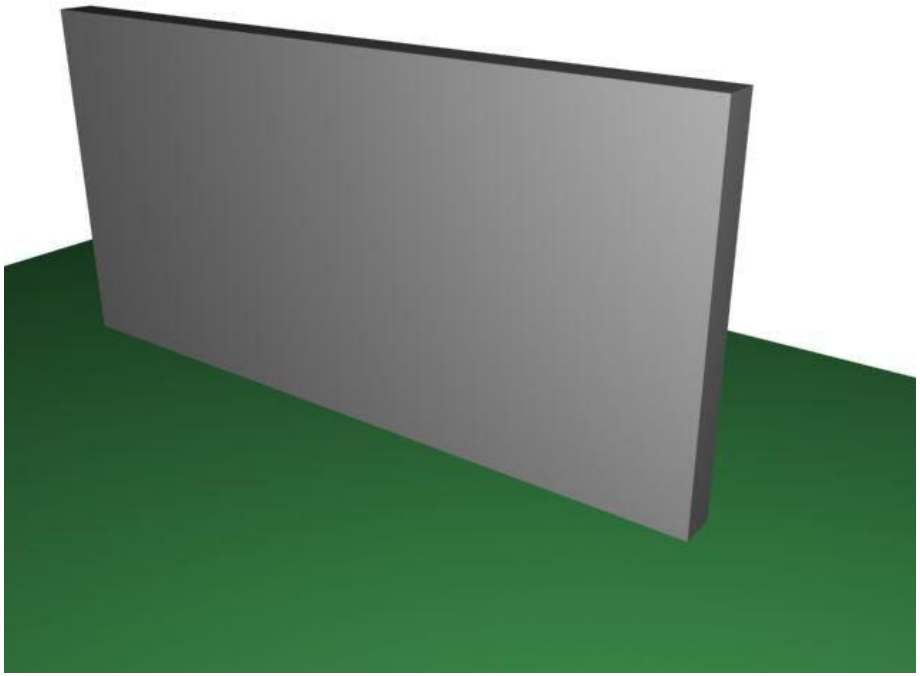
Université de Biskra
2019/2020

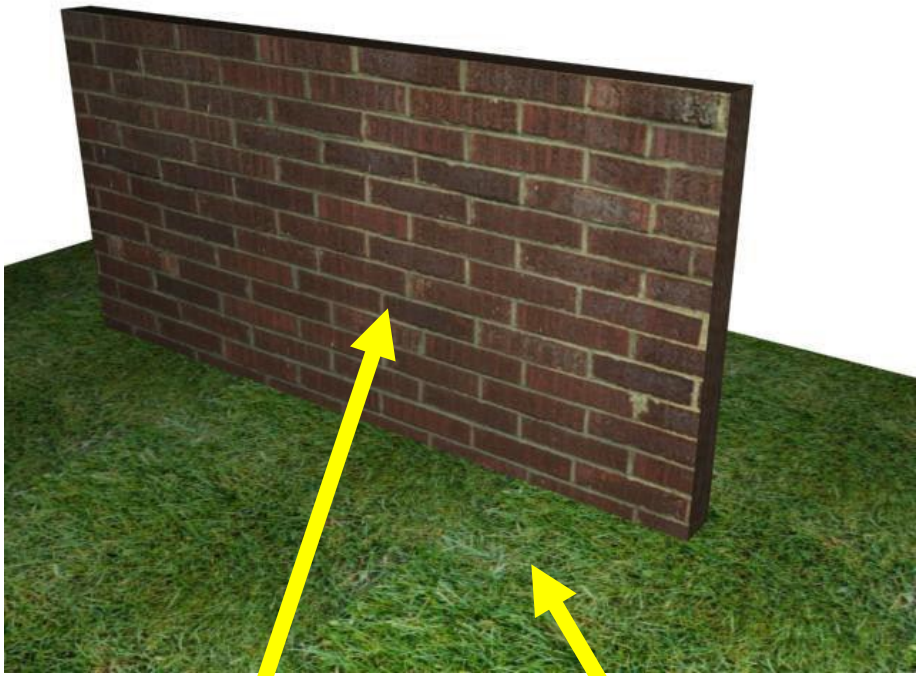
1. Problématique

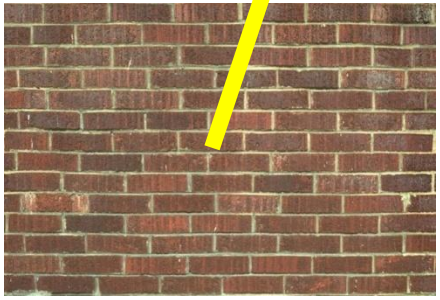




Solution : utilisation de **textures**.



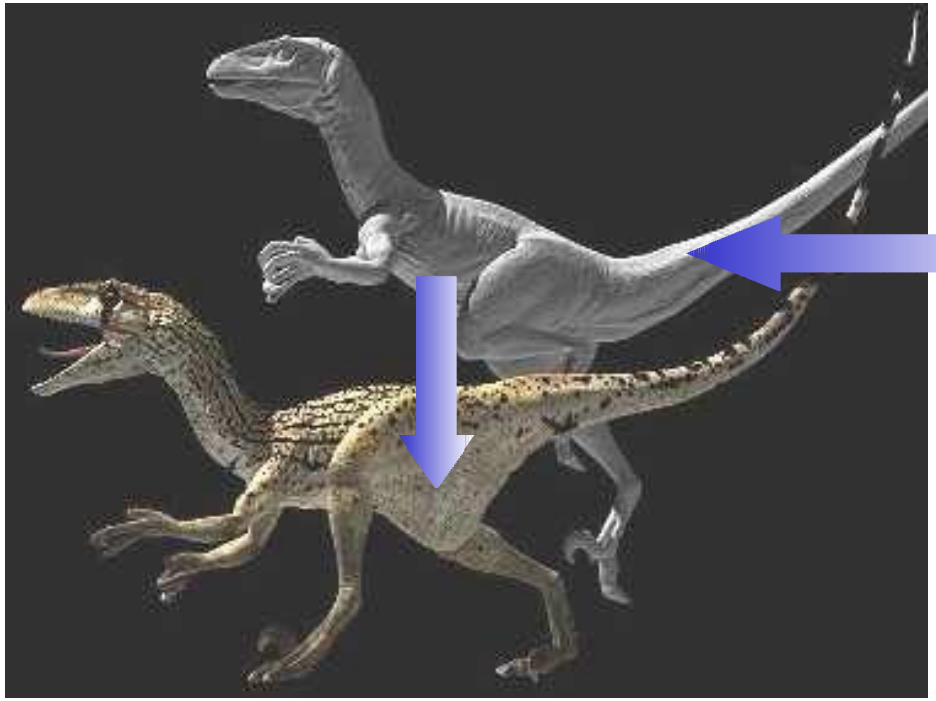




Plaquage de textures (« *texture mapping* »)
Plaquage d'images sur des primitives géométriques.

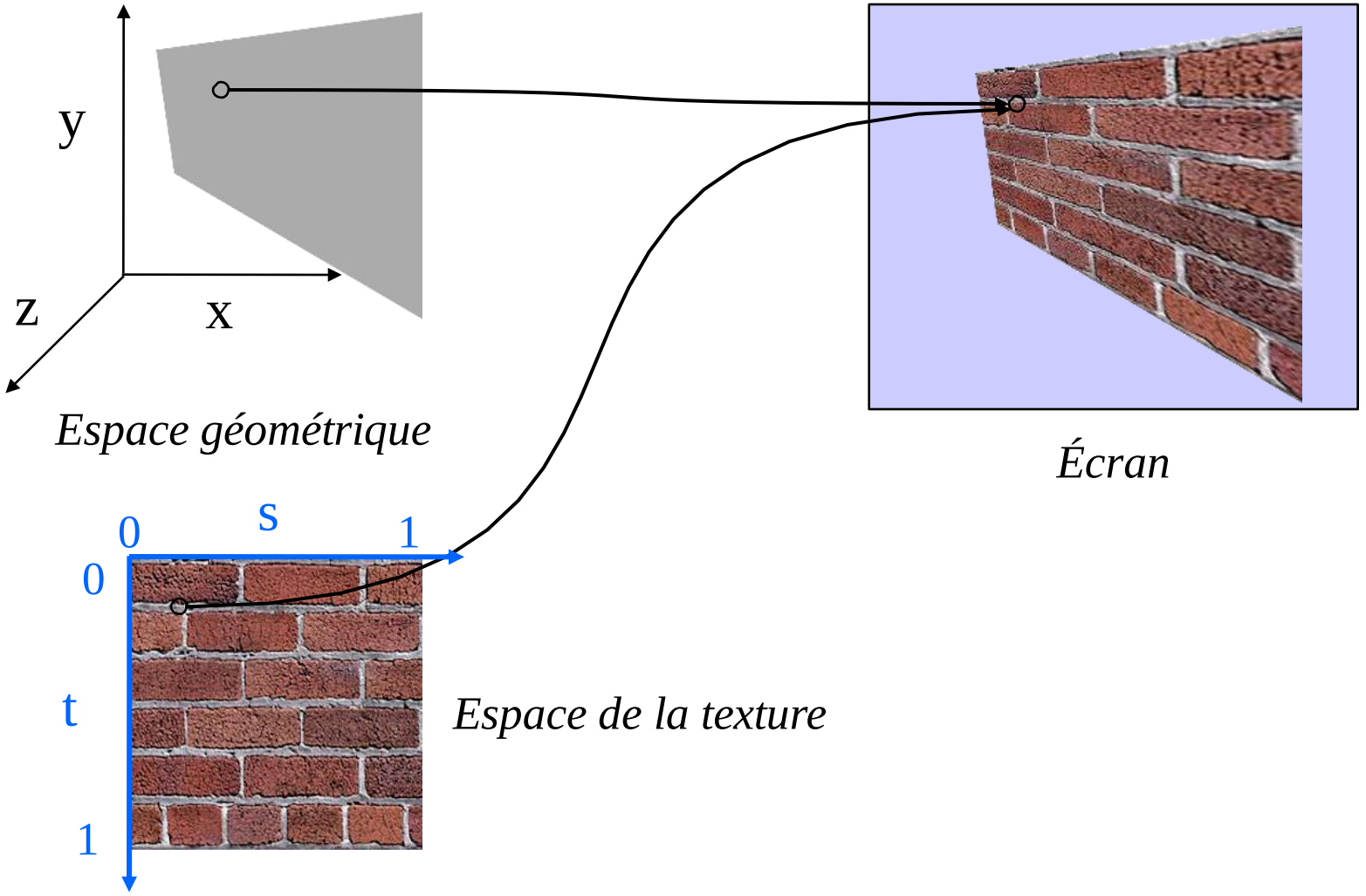
Objectifs

- Simuler des matériaux (pierre, bois, …)
- Réduire la complexité (nb de polygones) d'objets 3D
- Simulation de surfaces réfléchissantes
- …



2. Principe du plaquage de texture

2.1 Coordonnées de texture

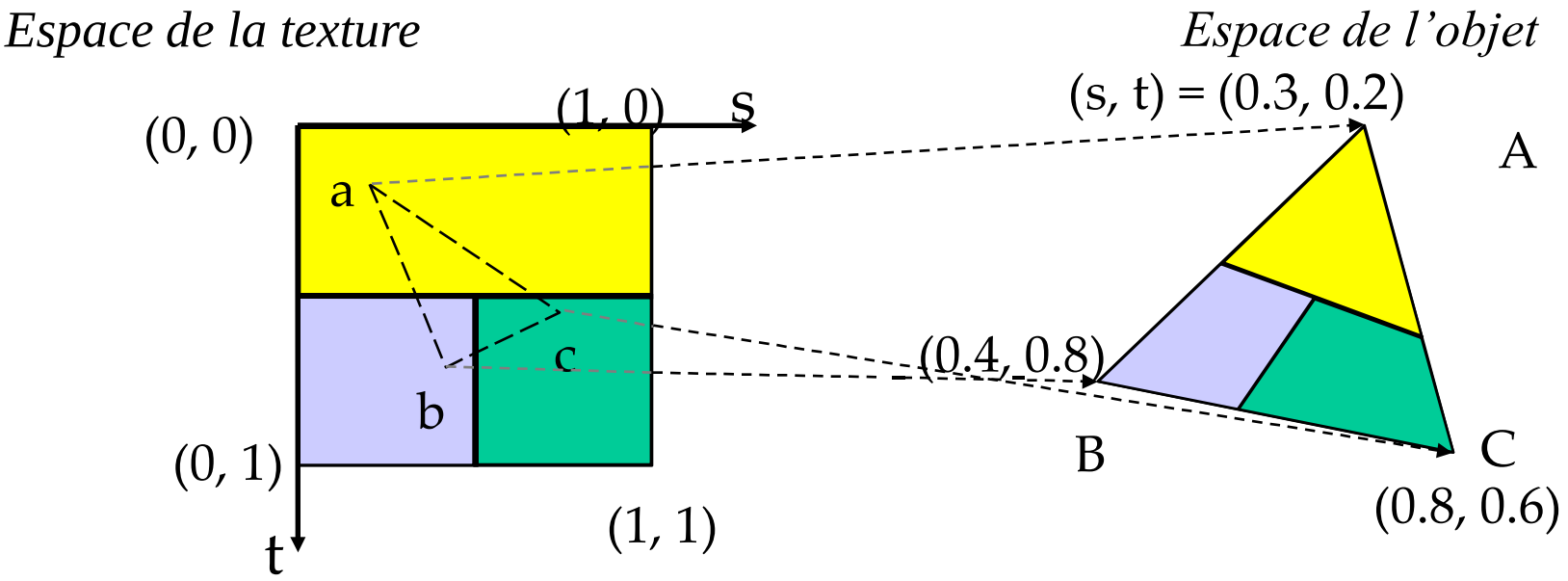


Les textures sont des images qui peuvent être en :

- 1D : coordonnée de texture (s)
- **2D : coordonnées de texture (s, t), cas le plus courant**
- 3D : coordonnées de texture (s, t, r) 4D : coordonnées de texture (s, t, r, q)

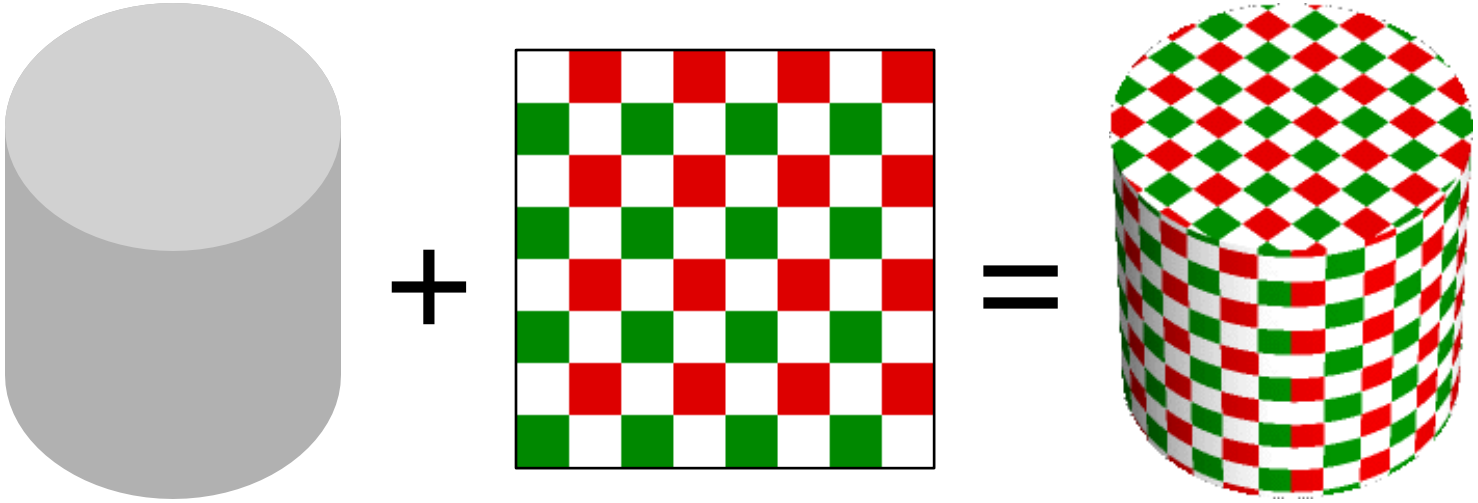
Un pixel d'une texture est appelé *texel*.

Ex: pour une texture 2D, un point dans l'image est donné par ses coordonnées (s,t), le coin supérieur gauche étant (0,0) et le coin inférieur droit étant (1,1).



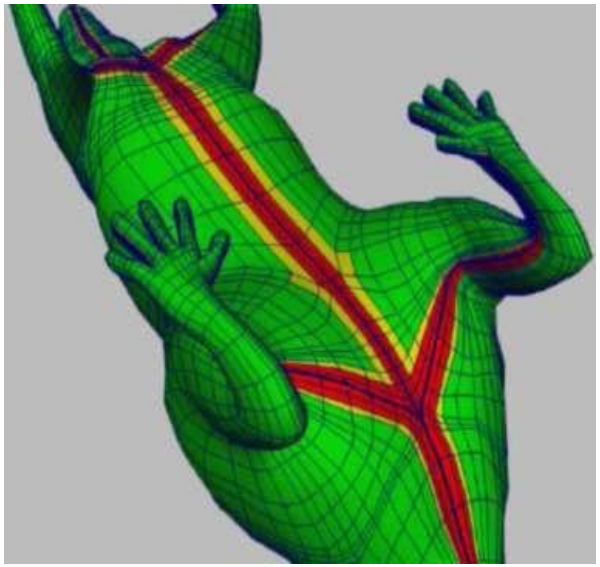
2.2 Paramétrisation

Question : Comment définir les coordonnées de texture des sommets de l'objet 3D à texturer ? « Comment définir les endroits de l'objet 3D où iront les couleurs de l'image ? »

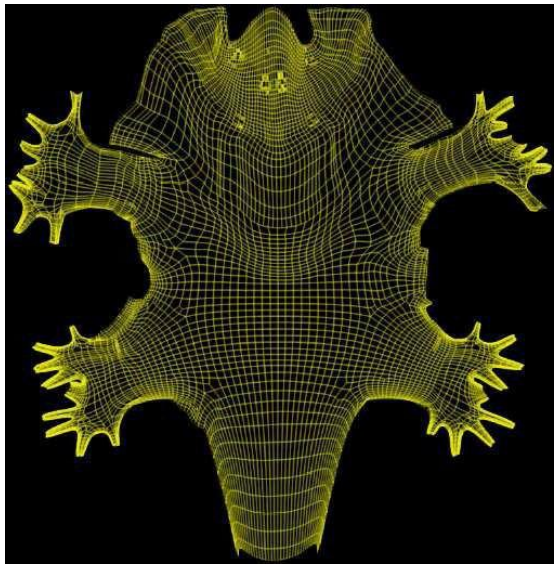


Objet 3D Texture Objet 3D texturé

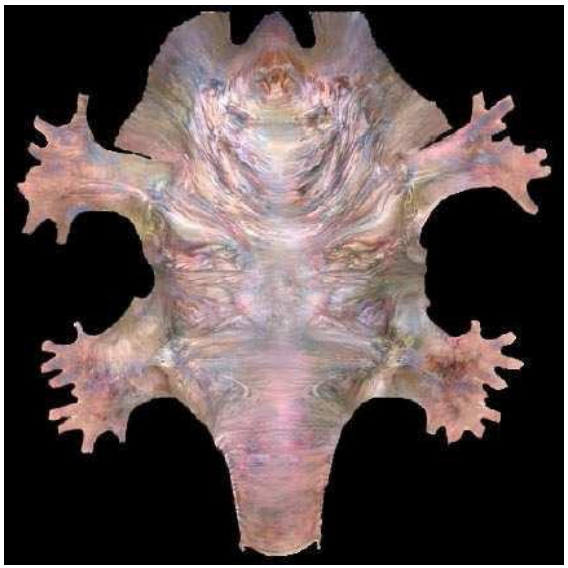
2.2.1 Dépliage de texture



Modèle 3D à texturer



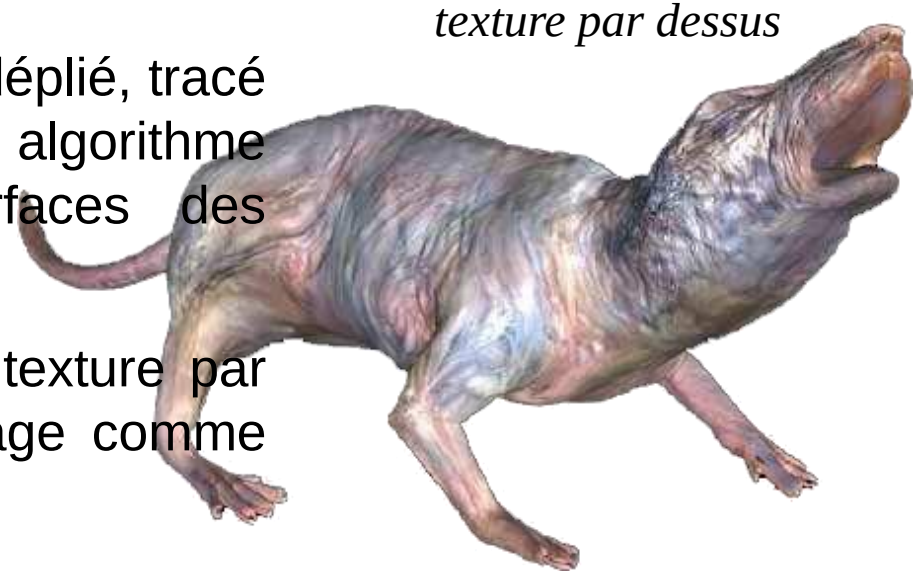
Le maillage 3D déplié en 2D



Un artiste dessine la texture par dessus

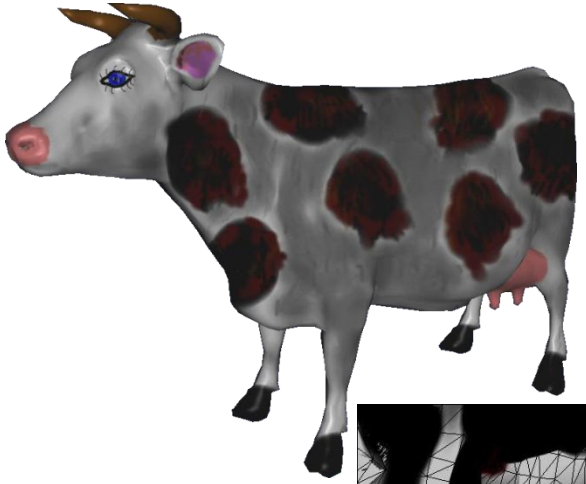
Le maillage du modèle 3D est déplié, tracé dans l'image 2D de la texture par un algorithme qui essaie de conserver les surfaces des triangles.

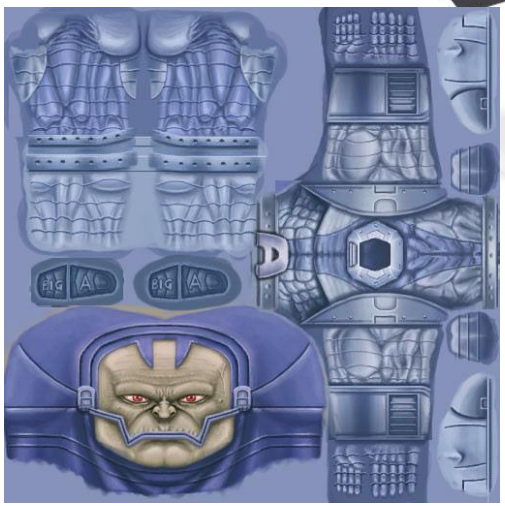
Un artiste dessine ensuite la texture par dessus en se servant du maillage comme guide.

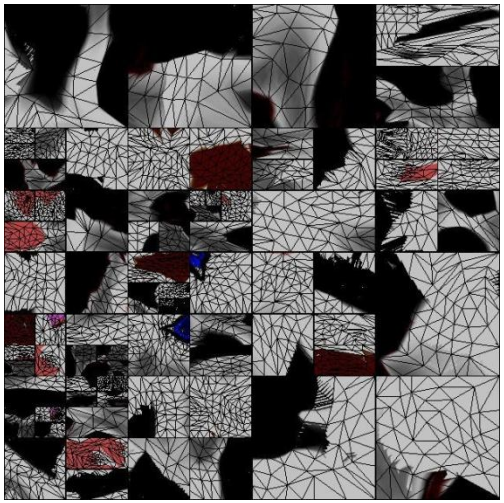


2.2.2 Atlas de textures

Dans l’espace de la texture, les différentes parties de la texture sont disjointes. Elles correspondent à des groupes de triangles du modèle 3D. Il faut répartir ces groupes de triangles de manière à optimiser l’utilisation de la surface de la texture.



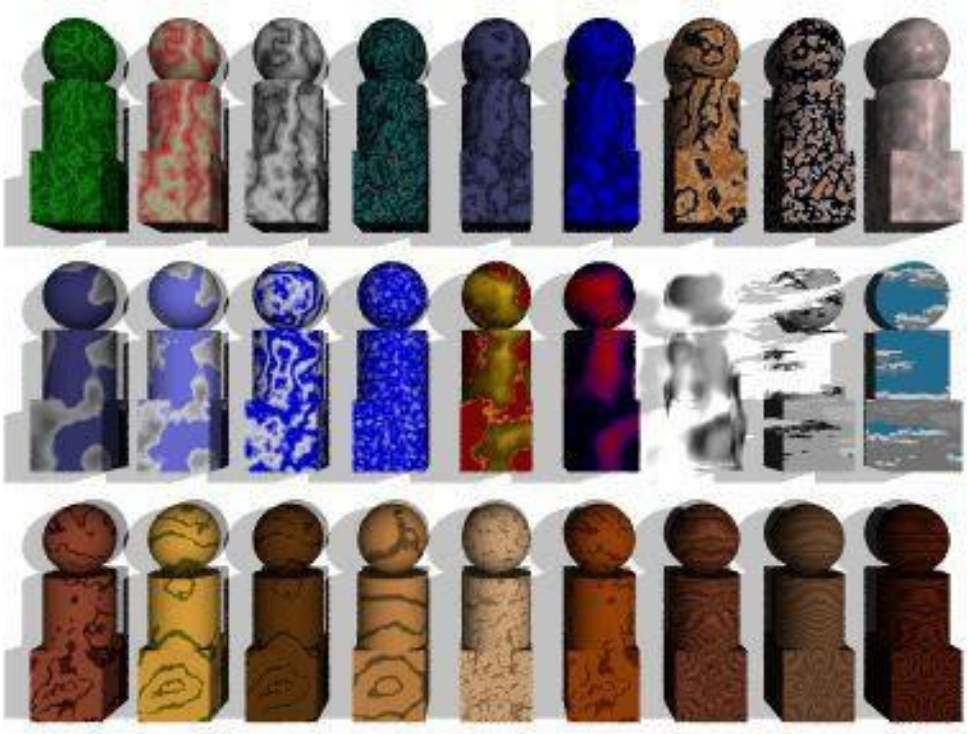




Textures procédurales

Les textures sont de deux types :

- **raster** = image composée de pixels (c'est le cas le plus courant)
- **procédural** = fonction mathématique



Avantages des textures procédurales (2D et 3D)

Ces fonctions nécessitent très peu de mémoire (à peine quelques octets, pour le code de ces fonctions).

Inconvénients des textures procédurales (2D et 3D)

Le calcul de la couleur en un point peut être long (dépend de la complexité de la fonction).

La détermination des paramètres de ces fonctions n'est pas facile (peu intuitif).

3. Utilisation de textures dans OpenGL

On procède en 3 étapes :

1) Spécifier la texture

- Lire ou générer une image En
- faire une texture
- Activer le plaquage de texture

2) Assigner les coordonnées de texture aux sommets de l'objet 3D

3) Spécifier les paramètres de textures

- Wraping, filtering, …

3.1 Spécifier la texture

3.1.1 Lire ou générer une image

```
BYTE    *img;
int     largeur,
GLuint          hauteur;
        texture;
glGenTextures(1, &texture);
img = load_tga( "image.tga", &largeur, &hauteur );
```

Explication:

Dans OpenGL, **chaque texture est référencée par un indice** (entier). Pour obtenir ces indices, on utilise la fonction :

```
glGenTextures(GLuint    n,    GLuint
      *tab_text);
```

Qui crée **n** indices de textures et les place dans le tableau d'entiers **tab_text**.

Ex: obtention d'un ensemble d'indices pour 10 textures (→ tableau de 10 indices) :

```
GLuint      tab_text[10];glGenTextures(10,
      tab_text);
```

On peut aussi utiliser cette fonction pour ne demander qu'un seul indice :

Ex: obtention d'un indice pour une seule texture (→ une seule variable entière) :

```
GLuint texture;
glGenTextures(1, &texture);
```

3.1.2 En faire une texture

Les textures doivent être stockées dans la RAM de la carte graphique.

- Lorsqu'on charge une image pour en faire une texture, il faut ensuite la transférer dans la RAM vidéo :

```
glBindTexture(GL_TEXTURE_2D, texture);

glTexImage2D( GL_TEXTURE_2D, 0, 3,
              largeur, hauteur,
              0, GL_RGB, GL_UNSIGNED_BYTE, img);
```

La fonction **`glBindTexture()`** permet de fixer l'indice de la **texture courante**. Cette texture sera utilisée pour toutes les opérations (plaquage, modification de paramètres, …) jusqu'au prochain appel de **`glBindTexture()`**.

Explication:

```
glTexImage2D( target, level, components, w, h,
              border, format, type, *texels );
```

target	:	**GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D**
level	:	**0** sans mip-mapping
components	:	nombre d'éléments par texel
w, h	:	dimensions de la texture (puissances de 2)
border	:	bordure supplémentaire autour de l'image
format	:	**GL_RGB, GL_RGBA, ...**
type	:	type des éléments des texels
texels	:	tableau de texels

3.1.3 Activer le plaquage de texture

On active la plaquage avec :

```
glEnable(GL_TEXTURE_2D);
```

On peut aussi désactiver le plaquage :

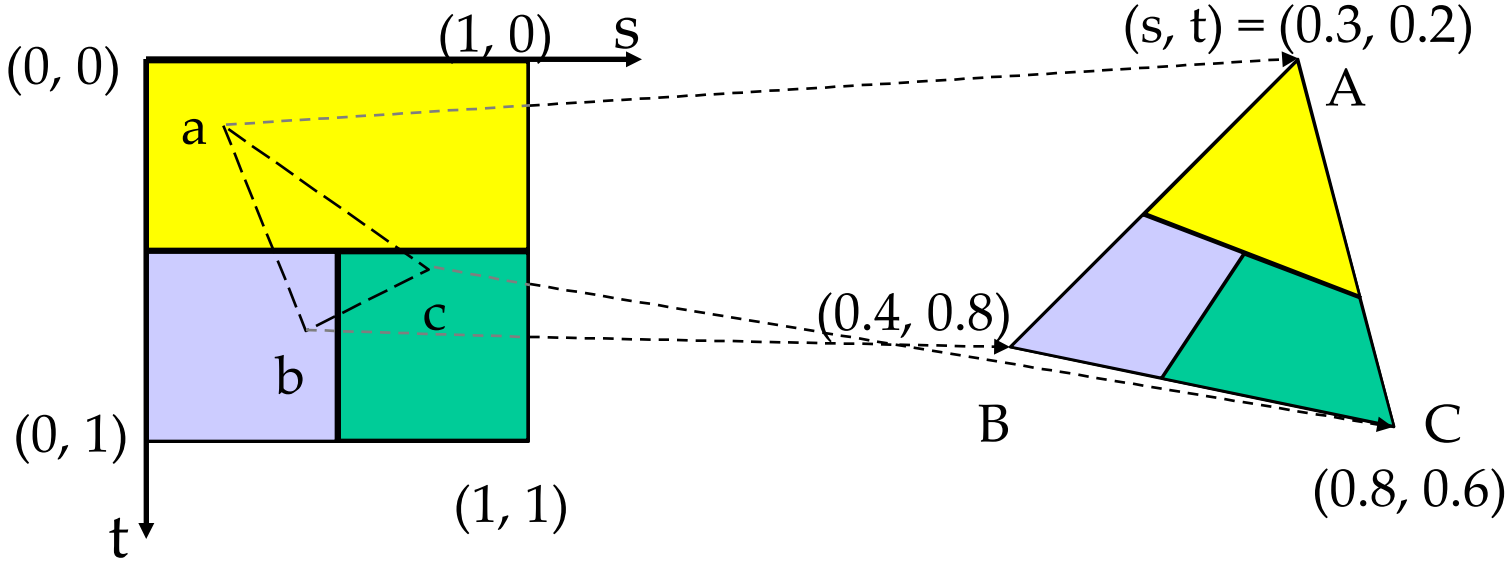
```
glDisable(GL_TEXTURE_2D);
```

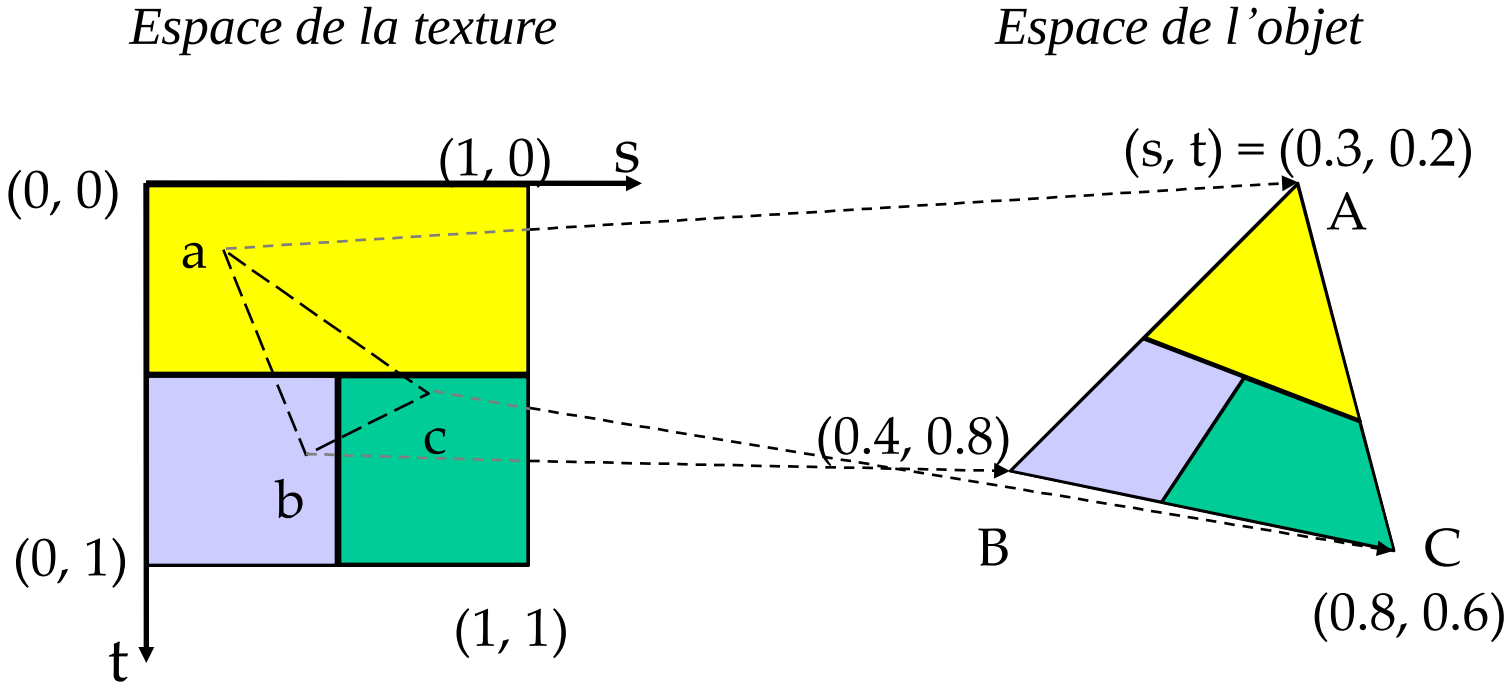
On peut appeler ces deux fonctions à tout moment dans le programme, par exemple pour activer temporairement le plaquage afin d'afficher un objet texturé, puis en le désactivant pour afficher un objet qui ne comporte pas de texture.

3.2 Assigner les coordonnées de texture aux points de l'objet 3D

Pour plaquer une texture sur un objet géométrique, associer à chaque sommet 3D de l'objet les coordonnées 2D de texture (normalisés entre 0 et 1).

Espace de la texture *Espace de l'objet*






```
glBindTexture(GL_TEXTURE_2D, texture);

glBegin(GL_TRIANGLES);
  glTexCoord2f(0.3f,0.2f);
  glVertex3f(10.0f, 12.0f, 0.0f);
  glTexCoord2f(0.4f,0.8f);
  glVertex3f(4.0f, 6.0f, 0.0f);
  glTexCoord2f(0.8f,0.6f);
  glVertex3f(15.0f, 2.0f, 0.0f);
glEnd();
```

3.3 Paramètres de textures

Modes de filtrage

- Réduction, agrandissement
- Mip-mapping

Modes de bouclage

- Répéter, tronquer

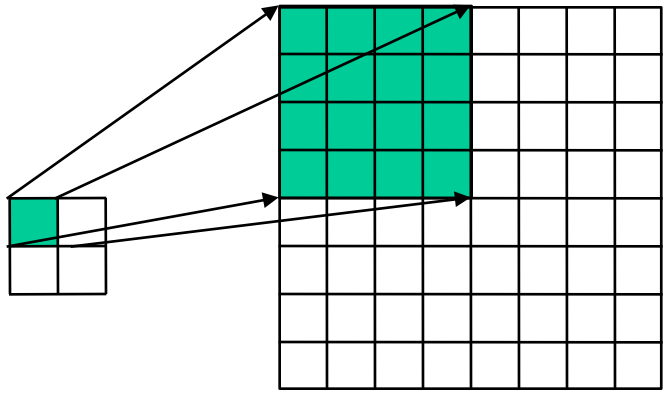
Fonctions de textures

- Comment mélanger la couleur d’un objet avec sa texture

3.3.1 Modes de filtrage

1) Réduction, agrandissement

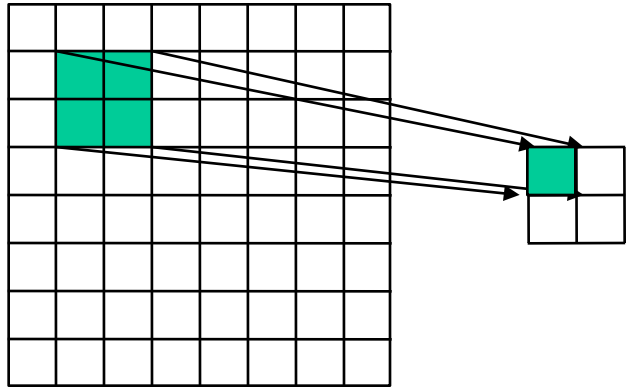
Les textures et les objets texturés ont rarement la même taille (en pixels). OpenGL défini des filtres indiquant comment un texel doit être agrandit ou réduit pour correspondre à la taille d'un pixel.



Texture *Polygone*

Agrandissement

`GL_TEXTURE_MAG_FILTER`



Texture *Polygone*

Réduction

`GL_TEXTURE_MIN_FILTER`

```
glTexParameteri(target, type,          mode);
```

Avec :

target : **GL_TEXTURE_2D, ...**

type : **GL_TEXTURE_MIN_FILTER,**
GL_TEXTURE_MAG_FILTER

mode : **GL_NEAREST,**
GL_LINEAR

GL_NEAREST : la couleur du pixel est donnée par celle du texel le plus proche.

GL_LINEAR : la couleur du pixel est calculée par interpolation linéaire des texels les plus proches.

GL_NEAREST

GL_LINEAR

Ex:

```
glBindTexture(GL_TEXTURE_2D,
      texture);
glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER,GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER,GL_LINEAR);
```

3.3.2 Modes de bouclage (Wrap)

Ce mode indique ce qui doit se produire si une coordonnée de texture sort de l'intervalle [0,1].

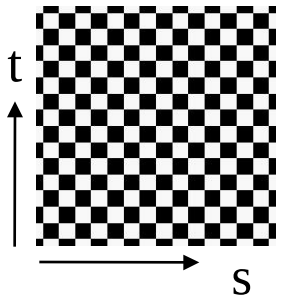
Deux possibilités :

- Répéter (« *Repeat* »)
- Tronquer (« *Clamp* »)

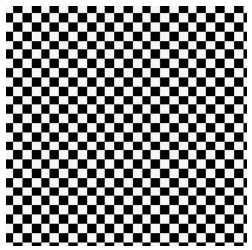
1) Répéter

Si le mode GL_REPEAT est utilisé, pour les coordonnées <0 ou >1, la partie entière est ignorée et seule la partie décimale est utilisée.

```
glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_WRAP_S,
                 GL_REPEAT );
```



Texture



GL_REPEAT

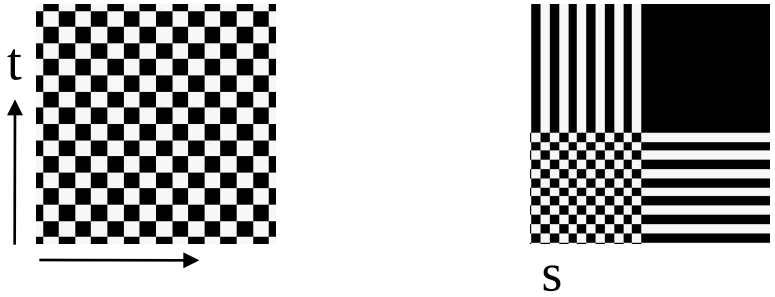
Exemple : Répétition d'une texture en mode GL_REPEAT.

→ Il faut créer une texture de telle sorte qu'elle boucle naturellement (ici, selon les 2 axes), sans qu'on voit les bords.

2) Tronquer

Si le mode GL_CLAMP est utilisé, la valeur de la texture aux extrêmes (0 ou 1) est utilisée.

```
glTexParameteri( GL_TEXTURE_2D,
                 GL_TEXTURE_WRAP_S,
                 GL_CLAMP );
```



GL_CLAMP
Texture

C’est le mode qu’on utilisera si la texture ne doit pas boucler.

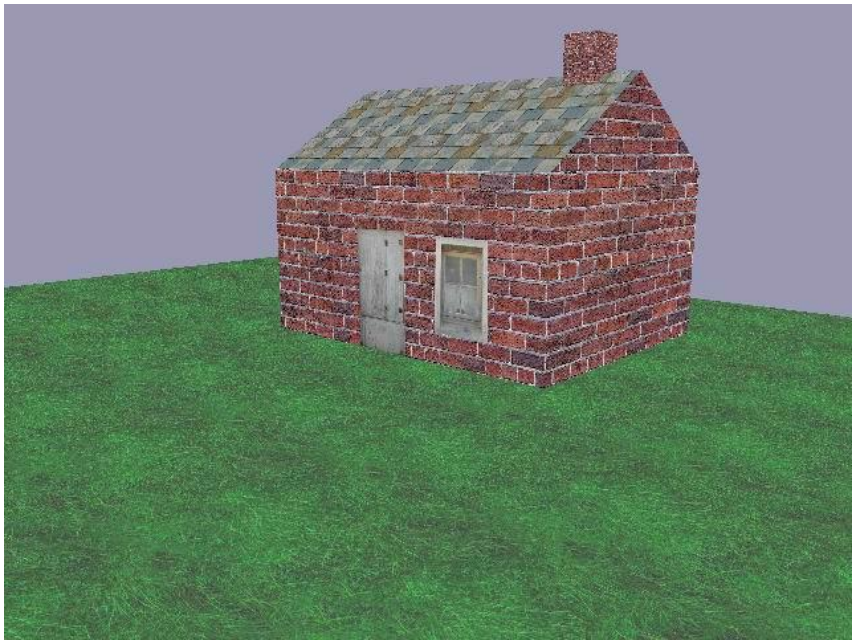
3.3.3 Fonctions de textures

Contrôle la manière selon laquelle la texture est mélangée à la couleur de l'objet.

```
glTexEnvf( GL_TEXTURE_ENV,
           GL_TEXTURE_ENV_MODE,
           param );
```

`param` peut prendre l'une des valeurs suivantes :

- **`GL_DECAL`** : remplace la couleur par le texel.
- **`GL_MODULATE`** : multiplie le texel par la couleur.

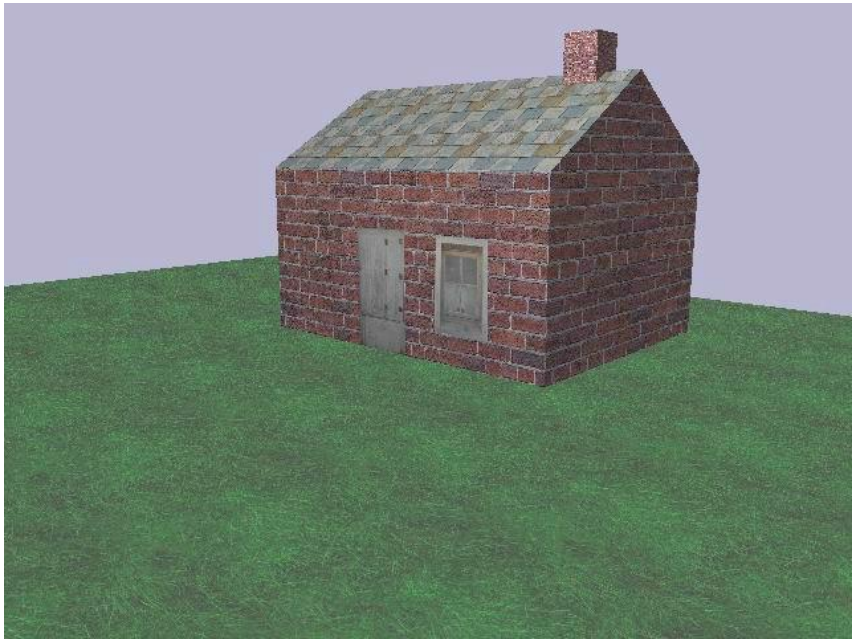


GL_DECAL

Remplace la couleur donnée par l'illumination de Phong par celle du texel.

```
glTexEnvf (GL_TEXTURE_ENV,

GL_TEXTURE_ENV_MODE,
GL_DECAL);
```



GL_MODULATE

Multiplie la couleur donnée par l'illumination de Phong par celle du texel.

```
glTexEnvf (GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE,
GL_MODULATE);
```

→ résultats plus réalistes

Exemple complet de lecture d'une texture RGB et de son application sur un triangle

```
// Déclarations de variables

BYTE    *img;
int     largeur,
Gluint          hauteur;
        texture;
// Fin des déclarations de variables
```

```
// Création d'une texture
//    - lecture d'une image
//    - chargement en mémoire vidéo
//    - réglage des paramètres de la texture

glGenTextures(1, &texture);


img = load_tga( "image.tga", &largeur, &hauteur );
if( img != NULL )
{
   glBindTexture(GL_TEXTURE_2D, texture);
   glTexImage2D( GL_TEXTURE_2D, 0, 3,
                 largeur, hauteur,
                 0, GL_RGB, GL_UNSIGNED_BYTE, img);
   delete[] img;
}
```

```
glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER,GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER,GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_WRAP_S,GL_REPEAT);

glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_WRAP_T,GL_REPEAT);

glTexEnvf(GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE,GL_MODULATE);

//    Fin   de    la    création
      d'une texture
```

```
// Utilisation d'une texture

// Si le mode "texture" avait été désactivé,
// on l'active :
glEnable(GL_TEXTURE_2D);


glBegin(GL_TRIANGLES);
   glTexCoord2f(0.0f,0.0f);
   glVertex3f(4.0f, 5.0f, 0.0f);
   glTexCoord2f(1.0f,0.0f);
   glVertex3f(10.0f, 5.0f, 0.0f);
   glTexCoord2f(0.0f,1.0f);
   glVertex3f(4.0f, 12.0f, 0.0f);
glEnd();


// Fin de l'utilisation d'une texture
```

Exemple complet de lecture d'une texture RGB et de son application en mip-mapping sur un triangle

```
// Déclarations de variables

BYTE    *img;
int     largeur,
GLuint          hauteur;
        texture;
//      des déclarations de variables

Fin
```

```
// Création d'une texture mip-map
//    - lecture d'une image
//    - chargement en mémoire vidéo en tant que mip-map
//    - réglage des paramètres de la texture

glGenTextures(1, &texture);


img = load_tga( "image.tga", &largeur, &hauteur );
glBindTexture(GL_TEXTURE_2D, texture);
gluBuild2DMipmaps(GL_TEXTURE_2D, 3,
                  largeur, hauteur,
                  GL_RGB, GL_UNSIGNED_BYTE, img);
```

```
glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER,
GL_LINEAR_MIPMAP_LINEAR);

glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER,
GL_LINEAR_MIPMAP_LINEAR);

glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_WRAP_S, GL_REPEAT);

glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_WRAP_T, GL_REPEAT);



glTexEnvf(GL_TEXTURE_ENV,
GL_TEXTURE_ENV_MODE, GL_MODULATE);

//    Fin   de    la    création
      d'une texture                    mip-map
```

```
// Utilisation d'une texture

// Si le mode "texture" avait été désactivé,
// on l'active :
glEnable(GL_TEXTURE_2D);


glBegin(GL_TRIANGLES);
   glTexCoord2f(0.0f,0.0f);
   glVertex3f(4.0f, 5.0f, 0.0f);
   glTexCoord2f(1.0f,0.0f);
   glVertex3f(10.0f, 5.0f, 0.0f);
   glTexCoord2f(0.0f,1.0f);
   glVertex3f(4.0f, 12.0f, 0.0f);
glEnd();


// Fin de l'utilisation d'une texture
```