Utilisation de texture

Utilisation de texture

- Donner un rendu plus réaliste
  - Mur de briques
  - Végétation
  - Ciel avec nuages (skybox)
  - Objets brillants (aspect métallique)
  - Etc.
- Sujet intéressant (textures 1D, 2D, 3D)

Utilisation de texture

- Coller une photo à un polygone (surface)
- Semblable au papier peint

Cliché 21
Objet auquel on a appliqué une texture réfléchie. L'illustration ci-dessus montre la texture originale, qui est une photographie scannée d'un café de Palo Alto, prise avec un objectif à très grand angle. Ci-contre, une coupe à laquelle on a appliqué cette texture de manière que l'objet semble renvoyer l'image de la texture. Voir Chapitre 9.

- Textures peuvent être appliquées à
  - des surfaces composées de plusieurs polygones
  - des surfaces courbes
- La texture peut :
  - peindre directement la surface et remplacer sa couleur
  - moduler la couleur de la surface et fusionner avec sa couleur
  - fusionner une couleur de texture et une autre couleur

Utilisation de texture

- Placage de texture :
  - Synonymes : Application de texture, mappage de texture, plaquage de texture, texturage, multitexturage
  - En anglais : Texture mapping
- Technique visant à dessiner un objet de manière à ce que les surfaces apparaissant sur cet objet soient couvertes d'une image.
- Attribuer une couleur aux fragments d'un objet en fonction :
  - d'une ou de plusieurs sources lumineuses,
  - d'une orientation de l'objet en ce pixel :
    → utilisation d'un modèle d'illumination,
  - des coordonnées réelles du point de l'objet correspondant au pixel de l'image :
    → calcul d'une couleur fonction de la couleur dans la texture plaquée.

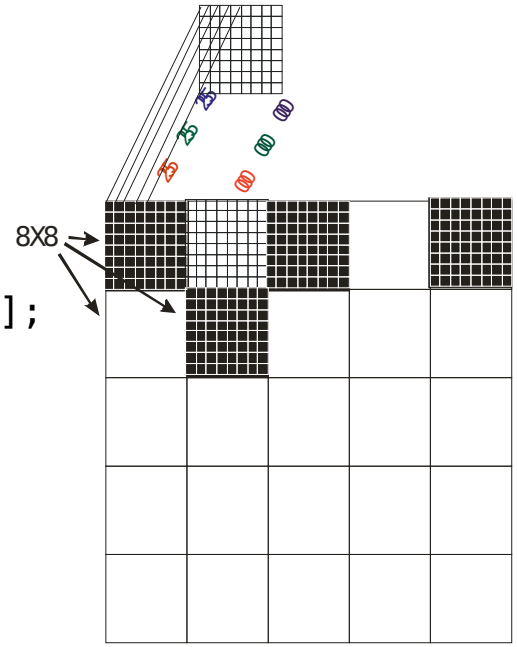
Obtenir une image pour la texture

- L'image est souvent une texture 2D :
  - Tableau rectangulaire de données contenant
    - Valeurs chromatiques (RGB)
    - Données de luminance ou de couleur
    - et possiblement alpha
  - Ces valeurs se nomment « texels »
- Les textures peuvent être 1D ou 2D ou 3D.

- La première étape est d'obtenir une image qui servira de texture.
  - On peut créer directement une image en mémoire
  - On peut lire un fichier qui contient une image et la charger en mémoire. (Possible en plusieurs formats.)

Obtenir une image pour la texture

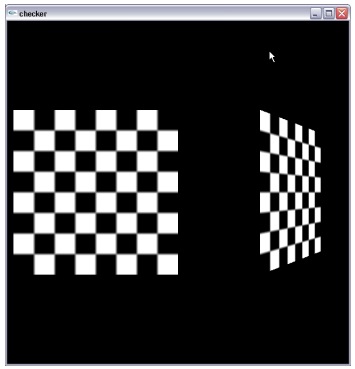
Exemple: (création de l'image)




```
// ... du livre OpenGL
const int checkImageWidth = 64;
const int checkImageHeight = 64;
GLubyte checkImage[checkImageWidth][checkImageHeight][3];

void makeCheckImage(void)
{
   int i, j, c;

   for (i = 0; i < checkImageWidth; i++)
   {
      for (j = 0; j < checkImageHeight; j++)
      {
         c = ((((i&0x8)==0)^((j&0x8)==0)))*255;
         checkImage[i][j][0] = (GLubyte) c;
         checkImage[i][j][1] = (GLubyte) c;
         checkImage[i][j][2] = (GLubyte) c;
      }
   }
}
```



Obtenir une image pour la texture

Exemple: (lecture d'un fichier d'image en format JPEG)

```
tImageJPG *LoadJPG(const char *filename)
{
  struct jpeg_decompress_struct cinfo;
  FILE *pFile = fopen(filename, "r");
  // Ouvrir un fichier jpeg (pointeur au fichier)
  if ( pFile == NULL) { Message("Erreur avec JPG!"); return NULL; }
  // Créer un gestionnaire d'erreur [error handler]
  jpeg_error_mgr jerr; cinfo.err = jpeg_std_error(&jerr);
  // Initialiser l'objet de décompression
  jpeg_create_decompress(&cinfo);
  // Spécifier la source des données
  jpeg_stdio_src(&cinfo, pFile);
  // Allouer la structure qui contiendra les données jpeg
  tImageJPG *pImageData = (tImageJPG*)malloc(sizeof(tImageJPG));
  // remplir la structure de données
  DecodeJPG(&cinfo, pImageData);
  // Libérer la mémoire utilisée
  jpeg_destroy_decompress(&cinfo);
  fclose(pFile);
  return pImageData;
}
```

Spécification de la texture

Créer l'objet texture et spécifier la texture associée à cet objet

- Nommer la texture pour OpenGL

```
glGenTextures( GLsizei n, GLuint *textureNames );
```

`n` : nombre de noms

`textureNames` : noms de texture

- Créer ou rendre courant un objet texture

```
glBindTexture( GLenum target, GLuint textureNames );
```

  - si l'entier non signé `textureNames` est utilisé pour la première fois, un objet texture est créé.
  - si on charge un objet texture existant, il devient actif.
  - si on utilise une valeur de zéro, on désactive la texture.

Spécification de la texture

Créer l'objet texture et spécifier la texture associée à cet objet

- Spécifier une texture 2D :

```
glTexImage2D( GLenum target, GLint level, GLint internalFormat,
              GLsizei width, GLsizei height, GLint border,
              GLenum format, GLenum type, const GLvoid *texels );
```

target : GL_TEXTURE_2D ou GL_PROXY_TEXTURE_2D
level : 0, 1, 2 pour spécifier des textures pour des niveaux de détail multiples
internalFormat : format des composantes RGBA (valeurs entre 1 et 4) ou
GL_ALPHA[4 8 12 16], GL_LUMINANCE[4 8 12 16],
GL_LUMINANCE[4 8 12 16]_ALPHA[4 8 12 16],
GL_INTENSITY{4 8 12 16}, GL_RGB[4 5 8 10 12 16], GL_R3_G3_B2,
GL_RGBA[2 4 8 12 16], GL_RGB5_A1, GL_RGB10_A2
width et *height* : dimension de l'image de texture (forme 2^m+2b : puissance de 2)
border : largeur de bordure
format : GL_COLOR_INDEX, GL_RGB, GL_RGBA, GL_RED, GL_GREEN, GL_BLUE,
GL_ALPHA, GL_LUMINANCE, GL_LUMINANCE_ALPHA
type : GL_BYTE, GL_UNSIGNED_BYTE, GL_ SHORT, GL_UNSIGNED_SHORT,
GL_INT, GL_INSIGNED_INT, GL_FLOAT, GL_BITMAP
texels : tableau contenant la texture (au moins 64 x 64)

Spécification de la texture

- Autres fonctions pour spécifier une texture 2D :

```
glTexImage2D( GLenum target, GLint level, GLint internalFormat,
              GLsizei width, GLsizei height, GLint border,
              GLenum format, GLenum type, const GLvoid *texels );

glCopyTexImage2D( GLenum target, GLint level, GLenum internalFormat,
                  GLint x, GLint y, GLsizei width, GLsizei height,
                  GLint border );

glTexSubImage2D( GLenum target, GLint level, … );

glCopyTexSubImage2D( GLenum target, GLint level, … );

gluScaleImage( GLenum format,
            GLsizei wIn, GLsizei hIn, GLenum typeIn, const void *dataIn,
            GLsizei wOut, GLsizei hOut, GLenum typeOut, GLvoid* dataOut );
```

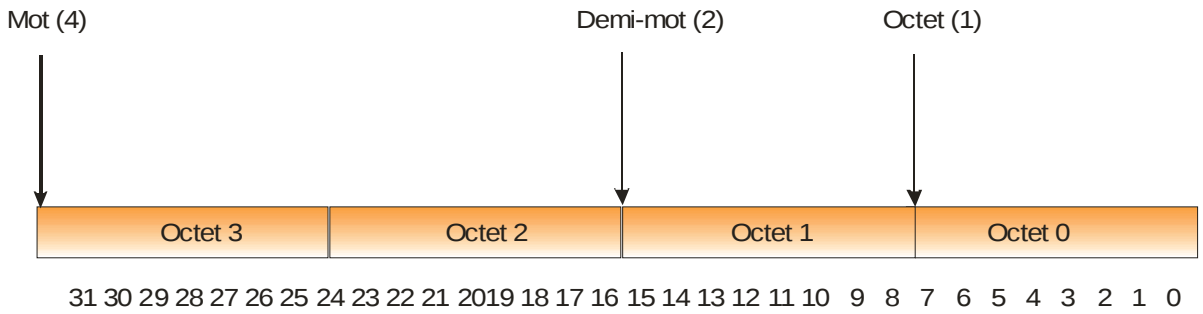
Compactage de données en mémoire

Définir le mode de stockage des pixels :

```
glPixelStore{if}( GLenum pname, TYPE param );
```

Affecte le fonctionnement des fonctions :

`glDrawPixels()`	`glReadPixels()`	`glBitmap()`	`glPolygonStipple()`
`glTexImage1D()`	`glTexImage2D()`	`glTexImage3D()`	`glTexSubImage1D()`
`glTexSubImage2D()`	`glTexSubImage3D()`	`glGetTexImage()`	`glGetColorTable()`
`glGetConvolutionFilter()`	`glGetSeparableFilter()`	`glGetHistogram()`	`glGetMinmax()`



Compactage de données en mémoire

Nom du paramètre (pname)	*Type*	*Valeur initiale*	*Plage valide*
GL_UNPACK_SWAP_BYTES, GL_PACK_SWAP_BYTE	GLboolean	FALSE	TRUE/FALSE
GL_UNPACK_LSB_FIRST, GL_PACK_LSB_FIRST	GLboolean	FALSE	TRUE/FALSE
GL_UNPACK_ROW_LENGTH, GL_PACK_ROW_LENGTH	GLint	0	Tout entier non négatif
GL_UNPACK_SKIP_ROWS, GL_PACK_SKIP_ROWS	GLint	0	Tout entier non négatif
GL_UNPACK_SKIP_PIXELS, GL_PACK_SKIP_PIXELS	GLint	0	Tout entier non négatif
GL_UNPACK_ALIGNEMENT, GL_PACK_ALIGNEMENT	GLint	4	**1**, 2, 4, 8
GL_UNPACK_IMAGE_HEIGHT, GL_PACK_IMAGE_HEIGHT	GLint	0	Tout entier non négatif
GL_UNPACK_SKIP_IMAGES, GL_PACK_SKIP_IMAGES	GLint	0	Tout entier non négatif

Sélection du filtre de la texture

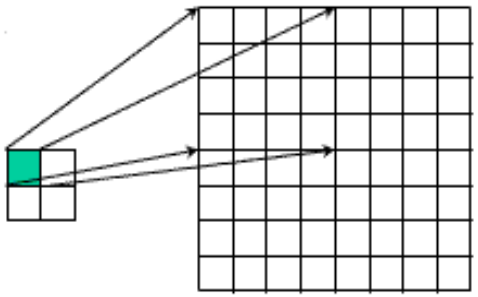
Il faut aussi indiquer comment doit être interpolée la texture

- si la texture est grossie lorsqu'affichée à l'écran :

```
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR );
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST );
```

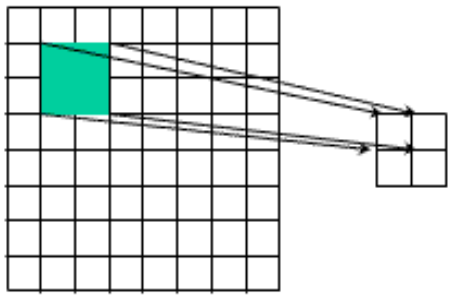
- si la texture est rapetissée lorsqu'affichée à l'écran :

```
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR );
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST );
```



texture polygone

(GL_TEXTURE_MAG_FILTER)



texture polygone

(GL_TEXTURE_MIN_FILTER)

Niveaux de détail: *mipmaps*

- Spécifier une série de textures pré-filtrées de résolutions décroissantes (*mipmaps*)

```
int gluBuild2DMipmaps( GLenum target, GLint internalFormat,
                       GLint width, GLint height, GLenum format,
                       GLenum type, void *texels );
```

Construire une série de *mipmaps* entre

`GL_TEXTURE_BASE_LEVEL (0)` et `GL_TEXTURE_MAX_LEVEL (1000)`

Exemple : 64 x 64, 32 x 32, *16 x 16*, 8 x 8, 4 x 4, 2 x 2, 1 x 1

Niveau de détails	0	1	2	3	4	5	6

```
int glGenerateMipmap( Glemun target );
```

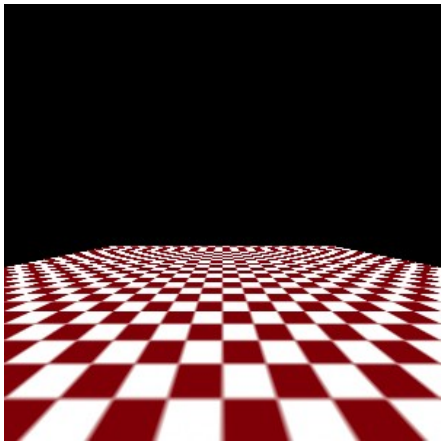
[OpenGL 3.0 et plus]

Niveaux de détail: *mipmaps* - exemple

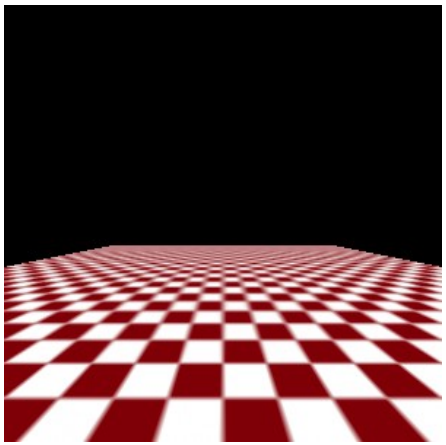
```
void CreateTexture( GLuint textureArray[], char* strFileName, int *textureID )
{
  tImageJPG *pImage = LoadJPG(strFileName);  // Charger l'image
  // Générer une texture
  glGenTextures( 1, textureID );
  // Lier la texture à l'index du tableau de texture et initialiser la texture
  glBindTexture( GL_TEXTURE_2D, *textureID );
  // Construire les Mipmaps
  gluBuild2DMipmaps( GL_TEXTURE_2D, 3, pImage->sizeX, pImage->sizeY,
                     GL_RGB, GL_UNSIGNED_BYTE, pImage->data );
  // Informer OpenGL de la qualité du placage de la texture
  glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_NEAREST );
  glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR );
  // Libérer la structure de l'image puisqu'OpenGL a stocké la texture
  if ( pImage )
  {
    if ( pImage->data ) free(pImage->data);
    free( pImage );
  }
}
```

Niveaux de détail: *mipmaps* - exemple

- L'utilisation de *mipmaps* demande plus de mémoire, mais peut produire de plus beaux rendus
- Exemple:



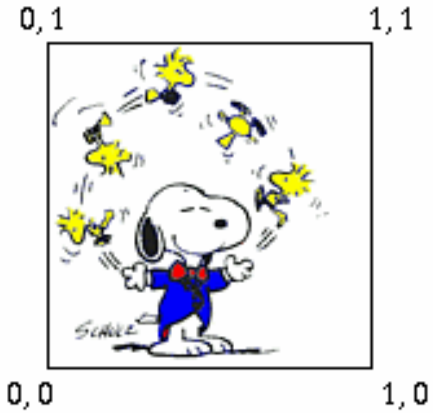
sans mipmaps



avec mipmaps

Coordonnées de texture

- Les coordonnées de texture associent la texture à la géométrie
  - Avec glTexCoord*( ) pour chaque sommet des objets à texturer, on assigne des valeurs qui indiquent les coordonnées de texture de ce sommet.
  - Les coordonnées de texture vont de 0.0 à 1.0 pour chaque dimension

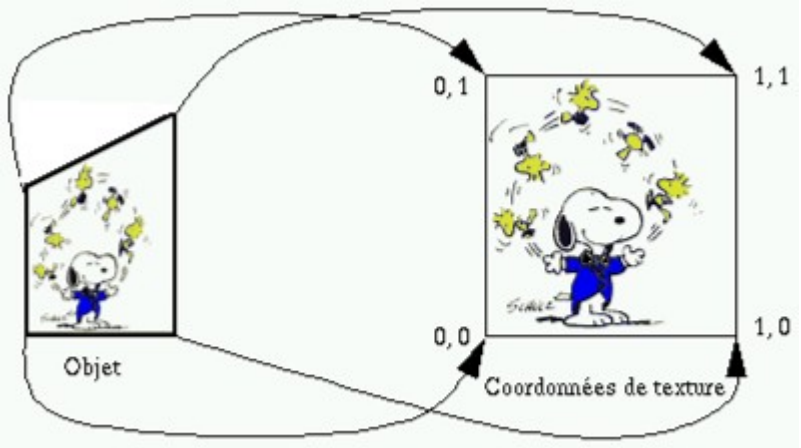


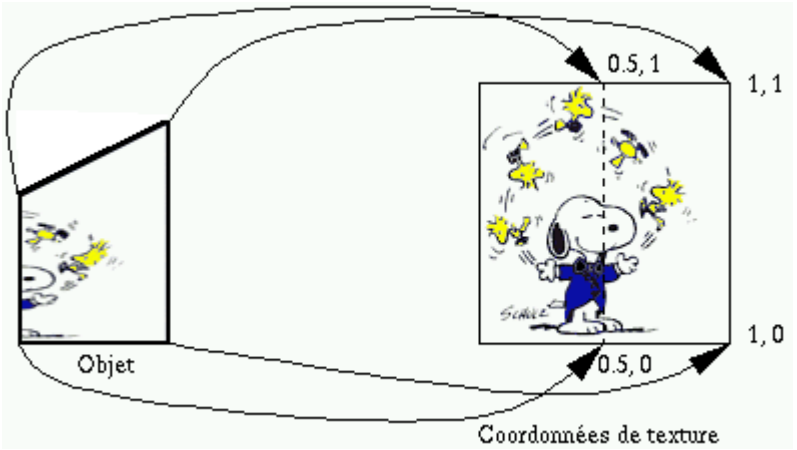
Coordonnées de texture

  - Les coordonnées peuvent aussi déborder cet intervalle.

Coordonnées de texture

- Les coordonnées de texture associées à la géométrie:
  - Si l'on désire que toute l'image soit affichée sur un quadrilatère, on assigne les valeurs de texture (0.0, 0.0), (1.0, 0.0), (1.0, 1.0) et (0.0, 1.0) aux coins du quadrilatère.
  - Si l'on désire appliquer uniquement la moitié droite de la texture sur ce quadrilatère, on assigne les valeurs de texture (0.5, 0.0), (1.0, 0.0), (1.0, 1.0) et (0.5, 1.0) aux coins du quadrilatère.





Coordonnées de texture

- Déclarer les coordonnées d'objet et de texture pour chaque sommet
  - Coordonnées d'objet : (*x, y, z, h*) (*glVertex**, *glColor**, *glNormal**, etc.)
  - Coordonnées de texture : (*s, r, t, q*) (*glTexCoord**)

- Spécifier les coordonnées de texture

```
glTexCoord{1 2 3 4}{s,i,f,d}(Type coords);
glTexCoord{1 2 3 4}{s,i,f,d}v(Type *coords);
```

- À noter : les coordonnées de texture sont transformées par la matrice de texture.

```
glMatrixMode( GL_TEXTURE_MATRIX );
glLoadIdentity( );
glTranslatef( 1.0, 1.0, 0.0 );
glRotatef( 45.0, 0.0, 0.0, 1.0 );
glScalef( 0.2, 0.3, 1.0 );
...
```

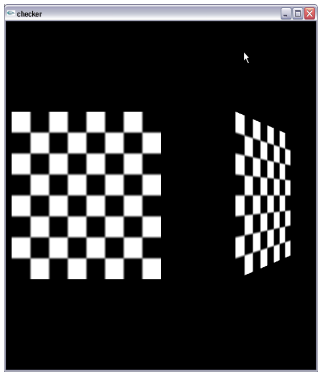
Coordonnées de texture

- Exemple : spécifier les coordonnées de texture associées à la géométrie

```
void AfficherScene()
{
  ...
  glEnable( GL_TEXTURE_2D );
  // Lier la texture stockée à l'indice zéro de  g_Texture[]
  glBindTexture( GL_TEXTURE_2D, g_Texture[0] );
  // Afficher un quadrilatère texturé à l'écran
  glBegin(GL_QUADS);
  // Afficher le sommet haut-gauche
  glTexCoord2f( 0.0, 0.0 );   glVertex3f( -1, 1, 0 );
  // Afficher le sommet bas-gauche
  glTexCoord2f( 0.0, 1.0 );   glVertex3f( -1, -1, 0 );
  // Afficher le sommet bas-droit
  glTexCoord2f( 1.0, 1.0 );   glVertex3f( 1, -1, 0 );
  // Afficher le sommet haut-droit
  glTexCoord2f( 1.0, 0.0 );   glVertex3f( 1, 1, 0 );
  glEnd();
  glDisable( GL_TEXTURE_2D );
  ...
}
```

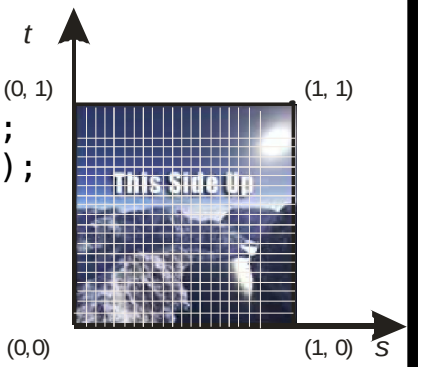
Coordonnées de texture

- Exemple : spécifier les coordonnées de texture associées à la géométrie



```
void Afficher()
{
    ...
    glBegin( GL_QUADS );
          // Coord texture              Coord. Géométrie
    glTexCoord2f( 0.0, 0.0 ); glVertex3f( -2.0, -1.0, 0.0 );
    glTexCoord2f( 0.0, 1.0 ); glVertex3f( -2.0, 1.0, 0.0 );
    glTexCoord2f( 1.0, 1.0 ); glVertex3f( 0.0, 1.0, 0.0 );
    glTexCoord2f( 1.0, 0.0 ); glVertex3f( 0.0, -1.0, 0.0 );

    glTexCoord2f( 0.0, 0.0 ); glVertex3f( 1.0, -1.0, 0.0 );
    glTexCoord2f( 0.0, 1.0 ); glVertex3f( 1.0, 1.0, 0.0 );
    glTexCoord2f( 1.0, 1.0 ); glVertex3f( 2.41421, 1.0, -1.41421 );
    glTexCoord2f( 1.0, 0.0 ); glVertex3f( 2.41421, -1.0, -1.41421 );
    glEnd();
    ...
}
```



Répétition de la texture

- Répétition de la texture dans l'espace texture
  - Il faut indiquer comment doivent être traitées les coordonnées de texture en dehors de l'intervalle [0.0, 1.0]. Est-ce que la texture est répétée pour recouvrir l'objet ou au contraire seuillée ("*clamped*")?
  - On utilise la fonction `glTexParameter()` pour positionner les paramètres `GL_TEXTURE_WRAP_S` pour la dimension horizontale de la texture ou `GL_TEXTURE_WRAP_T` pour la dimension verticale à `GL_CLAMP` ou `GL_REPEAT`.

```
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
ou
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

Répétition de la texture

- Répéter des textures
  - Assigner des coordonnées hors de l'intervalle [0, 1]
  - Par exemple, si les coordonnées de texture sont dans l'intervalle [0, 5] dans les 2 directions, la texture est reproduite 5x5 fois.
- Exemple :

```
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT );
glBegin( GL_QUADS);
glTexCoord2f( 0.0, 0.0 ); glVertex3f( 0.0, 0.0, 0.0 );
glTexCoord2f( 0.0, 5.0 ); glVertex3f( 0.0, 1.0, 0.0 );
glTexCoord2f( 5.0, 5.0 ); glVertex3f( 1.0, 1.0, 0.0 );
glTexCoord2f( 5.0, 0.0 ); glVertex3f( 1.0, 0.0, 0.0 );
glEnd();
```

Paramètres de glTexParameter*

```
void glTexParameter{if}( GLenum target, GLenum pname, TYPE param );
void glTexParameter{if}v( GLenum target, GLenum pname, TYPE *param );
```

target : GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D
pname : paramètre
param : valeurs

Paramètres de glTexParameter*

Paramètre [*pname*]	Valeurs [*param*]
GL_TEXTURE_WRAP_S	**GL_CLAMP, GL_CLAMP_TO_EDGE, GL_REPEAT**
GL_TEXTURE_WRAP_R	**GL_CLAMP, GL_CLAMP_TO_EDGE, GL_REPEAT**
GL_TEXTURE_WRAP_T	**GL_CLAMP, GL_CLAMP_TO_EDGE, GL_REPEAT**
GL_TEXTURE_MAG_FILTER	**GL_NEAREST, GL_LINEAR**
GL_TEXTURE_MIN_FILTER	**GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_NEAREST, GL_LINEAR_MIPMAP_LINEAR**
GL_TEXTURE_BORDER_COLOR	Une des quatre valeurs (RGBA) situées entre [0.0, 1.0]
GL_TEXTURE_PRIORITY	[0.0, 1.0] pour l'objet de texture actif
GL_TEXTURE_MIN_LOD	Niveau de détail (min) : une valeur quelconque en virgule flottante
GL_TEXTURE_MAX_LOD	Niveau de détail (max) : une valeur quelconque en virgule flottante
GL_TEXTURE_BASE_LEVEL	Niveau de base de la texture : un entier non négatif quelconque
GL_TEXTURE_MAX_LEVEL	Niveau maximal de la texture : un entier non négatif quelconque

Espace de texture

Coordonnées de texture

Espace de texture T(*s*, *t*)

Paramétrisation de la surface

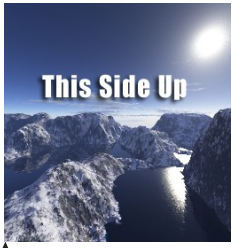
Coordonnées virtuelles

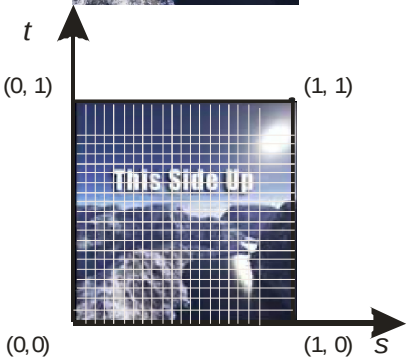
Espace de l'objet (x_w, y_w, z_w)

Projection

Coordonnées d'affichage

Espace de d'affichage (*i*, *j*)

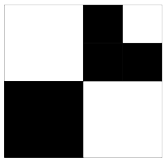




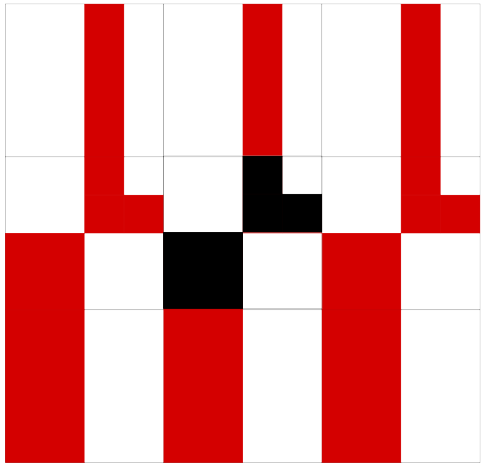
ÉCOLE POLYTECHNIQUE MONTRÉAL

Espace de texture

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



la texture

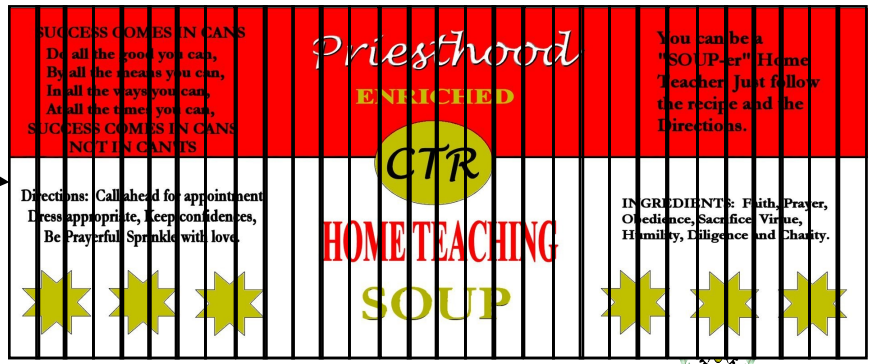


l'espace de texture avec ces énoncés

(autres exemples au tableau)

Espace de texture

1	(0, 0)	(1/30, 0)	(1/30, 2/3)	(0, 2/3)
2	(1/30, 0)	(2/30, 0)	(2/30, 2/3)	(1/30, 2/3)
3	(2/30, 0)	(3/30, 0)	(3/30, 2/3)	(2/30, 2/3)
…				
30	(29/30, 0)	(1, 0)	(1, 2/3)	(29/30, 2/3)



Combinaison des couleurs

- Application de texture : les valeurs associées aux textures peuvent
  - moduler la couleur dans laquelle la surface serait restituée en l'absence de texture
  - combiner les deux composants

void glTexEnv{if} (GLenum *target*, GLenum *pname*, TYPE *param*);

void glTexEnv{if}v (GLenum *target*, GLenum *pname*, TYPE *param*);

target : GL_TEXTURE_ENV (, GL_TEXTURE_FILTER_CONTROL)

pname : GL_TEXTURE_ENV_MODE, GL_TEXTURE_ENV_COLOR, ... (, GL_TEXTURE_LOD_BIAS)

- GL_TEXTURE_ENV_MODE : GL_DECAL, GL_REPLACE, GL_MODULATE, GL_BLEND, GL_ADD ou GL_COMBINE
- GL_TEXTURE_ENV_COLOR : Constante symbolique ou valeur RGBA
- GL_COMBINE_RGB, GL_COMBINE_ALPHA : GL_REPLACE, GL_MODULATE, GL_ADD, G L_ADD_SIGNED, GL_INTERPOLATE, GL_SUBTRACT (, GL_DOT3_RGB, GL_DOT3_RGBA)
- GL_RGB_SCALE, GL_ALPHA_SCALE : facteur de mise à l'échelle
- ...

ex. : glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);

Combinaison des couleurs

- CpAp: couleur du fragment en entrée (ou celle calculées à l'étape précédente lorsqu'on utilise le multitexturage)
- CsAs: couleur (source) de la texture 

- CcAc: couleur de l'environnement (GL_TEXTURE_ENV_COLOR)
- CvAv: Couleur résultante calculée
- Allez voir toutes les combinaisons possibles *en ligne*: glTexEnv

Résumé

- Obtenir d'un fichier un tableau 2D (une image) ou 1D ou 3D
- Nommer la texture pour OpenGL

```
glGenTextures(…);
```

- Créer ou rendre courant un objet texture

```
glBindTexture(…);
```

- Spécifier une texture 2D (pour plusieurs niveaux de détails [mipmaps])

```
glTexImage2D(…), gluBuild2DMipmaps(…);
```

- Spécifier les filtres et les paramètres de l'environnement

```
glTexParameteri(…);
glTexParameteri(…);
glTexEnvi(…)
```

- Spécifier les coordonnées de texture pour la surface à texturer

```
glBindTexture(…..);
```

- Dessiner la scène en déclarant des coordonnées de texture

```
glTexCoord2f( 0.0, 0.0 );      glVertex3f( -1, 1, 0 );
glTexCoord2f( 0.0, 1.0 );      glVertex3f( -1, -1, 0 );
glTexCoord2f( 1.0, 1.0 );      glVertex3f( 1, -1, 0 );
glTexCoord2f( 1.0, 0.0 );      glVertex3f( 1, 1, 0 );
```

Résumé

- Créer l'objet texture et spécifier la texture associée à cet objet

```
void myinit(void)
{
    MakeCheckImage(); // ou charger une image
    glPixelStorei( GL_UNPACK_ALIGNMENT, 1 );
    glTexImage2D( GL_TEXTURE_2D, 0, 3,
                  checkImageWidth, checkImageHeight, 0, GL_RGB,
                  GL_UNSIGNED_BYTE, &checkImage[0][0][0]);
```

- Indiquer comment la texture doit être appliquée à chaque pixel et spécifier le type de filtres utilisés

```
    glTexParameterf(...);
    glTexEnvf(...);
```

- Activer et désactiver le placage de texture

```
    glEnable( GL_TEXTURE_2D );
    glDisable( GL_TEXTURE_2D );
}
```

Accès aux textures dans les nuanceurs

- Dans le nuanceur de sommets :
  - `gl_MultiTexCoord0` permet d'accéder aux coordonnées spécifiées dans le programme principal avec `gl_TexCoord*();`
  - Le nuanceur de sommets doit produire les coordonnées de texture pour le sommet dans `gl_TexCoord[0]`, souvent ainsi :

```
gl_TexCoord[0]= gl_MultiTexCoord0;
```

- Les coordonnées de texture sont interpolées lors du tramage

- Dans le nuanceur de fragments :
  - `gl_TexCoord[0]` est la coordonnée de texture pour le fragment courant
  - On l'utilise pour accéder à la texture :

```
uniform sampler2D laTexture;
couleur = texture2D( laTexture, gl_TexCoord[0].xy );
```