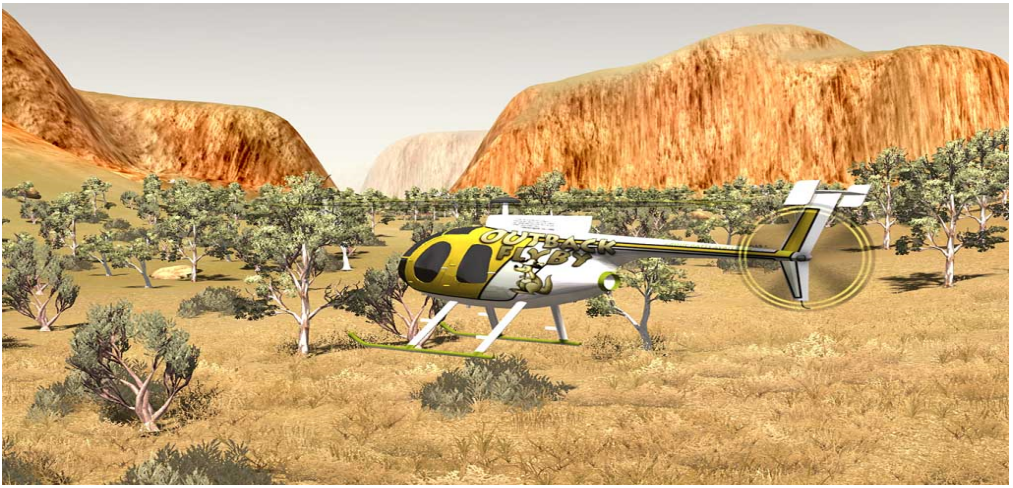
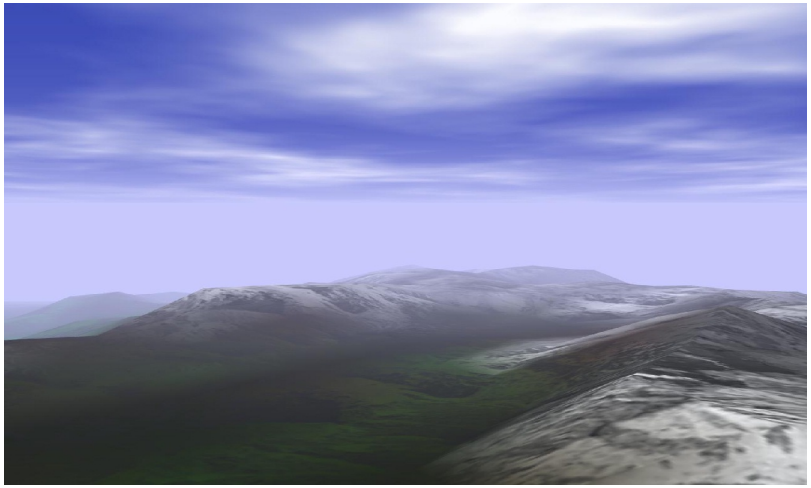




Texture et transparence

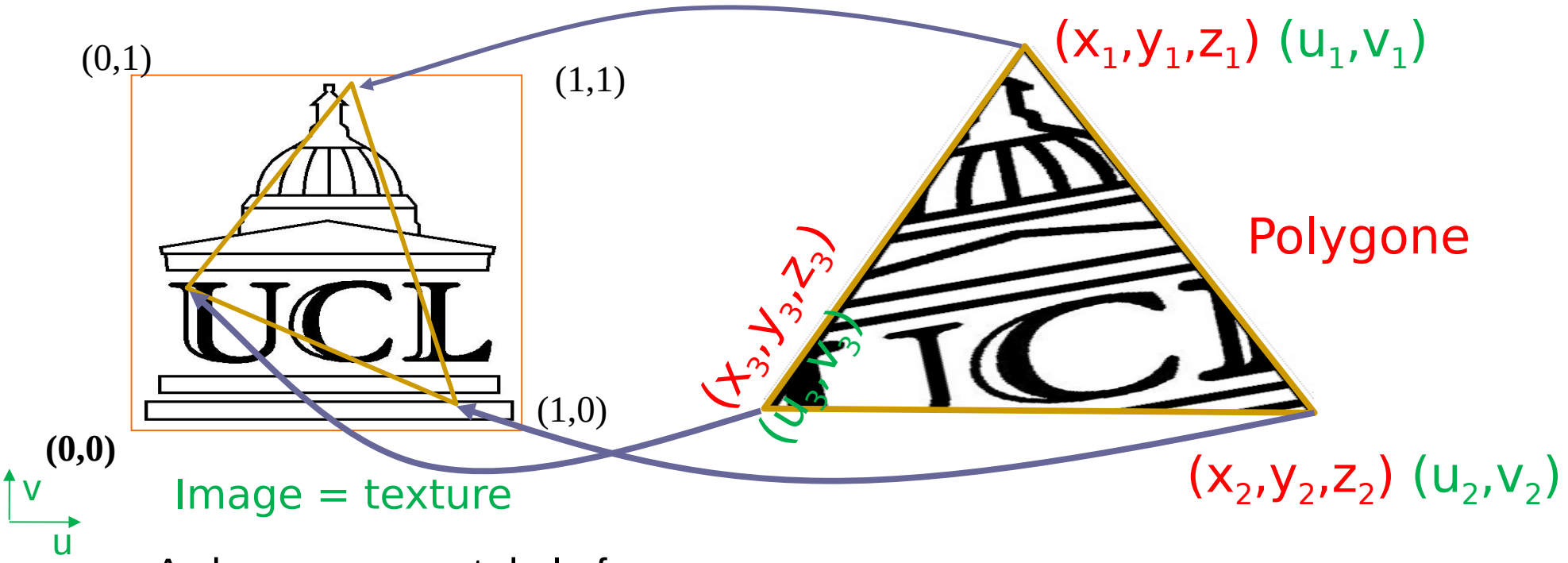
Florence Zara (semestre automne)
LIRIS-ORIGAMI, Université Lyon 1

Introduction aux textures

- Objectif
  - Réalisme
  - Simplicité
- Image
  - Unique primitive
  - Complexe comme des millions
- Vocabulaire
  - Pixel de l'image (texture)

  = TEXEL pour TEXture ELement

Plaquage de la texture



- A chaque sommet de la face
  - Coordonnées textures (u,v) à 2 dimensions
  - (u,v) correspond à une position dans l'image (dans la texture)

Utilisation des textures - code gkit

Première étape : création de la texture à partir d'une image

- Activation de la gestion des textures 2D

```
glEnable(GL_TEXTURE_2D);
```

- Déclaration de la texture dans la class `Viewer_etudiant`

```
GLuint papillon_texture; // Texture avec image papillon
```

- Initialisation de la texture - à faire dans `Viewer_etudiant::init()`

```
papillon_texture = read_texture(0, "data/papillon.png");
```

A partir d'une image, on a créé une texture

Utilisation des textures – code gkit

Deuxième étape : définir les coordonnées de texture du Mesh

(= comment on va plaquer une image sur le Mesh)

- Spécification des coordonnées de textures **pour chaque sommet** du maillage lors de la création du `Mesh` :

```
void ViewerEtudiant::init_votreObjet()
{
        m_quad.normal(0, 0, 1);
        m_quad.texcoord(0,0); // coordonnées de texture
        m_quad.vertex(-1, -1, 0);
}
```

- Rq : changements possibles dans `ViewerEtudiant::update()` :

```
m_quad.texcoord(numero-du-vertex, newu, newv);
```

Exemple – code gkit lors création Mesh

```
Mesh m_votreObjet;   // Declaration dans class ViewerEtudiant

void ViewerEtudiant::init_votreObjet()
{
    m_votreObjet = Mesh(GL_TRIANGLE_STRIP); // Primitive OpenGL

    m_votreObjet.normal(  0, 0, 1 );  // Normale au sommet 0
    m_votreObjet.texcoord( 0, 0 );    // Coord (u,v) sommet 0
    m_votreObjet.vertex( -1, -1, 0 ); // Creation du sommet 0

    m_votreObjet.normal(  0, 0, 1 );  // Normale au sommet 1
    m_votreObjet.texcoord( 0, 1 );    // Coord (u,v) sommet 1
    m_votreObjet.vertex(  1, -1, 0 ); // Creation du sommet 1

   ...
}
```

Attention à l'ordre :

1. Définition normale / texture
2. Définition des coordonnées du sommet

Utilisation des textures - code gkit

Troisième étape : associée une texture à un objet

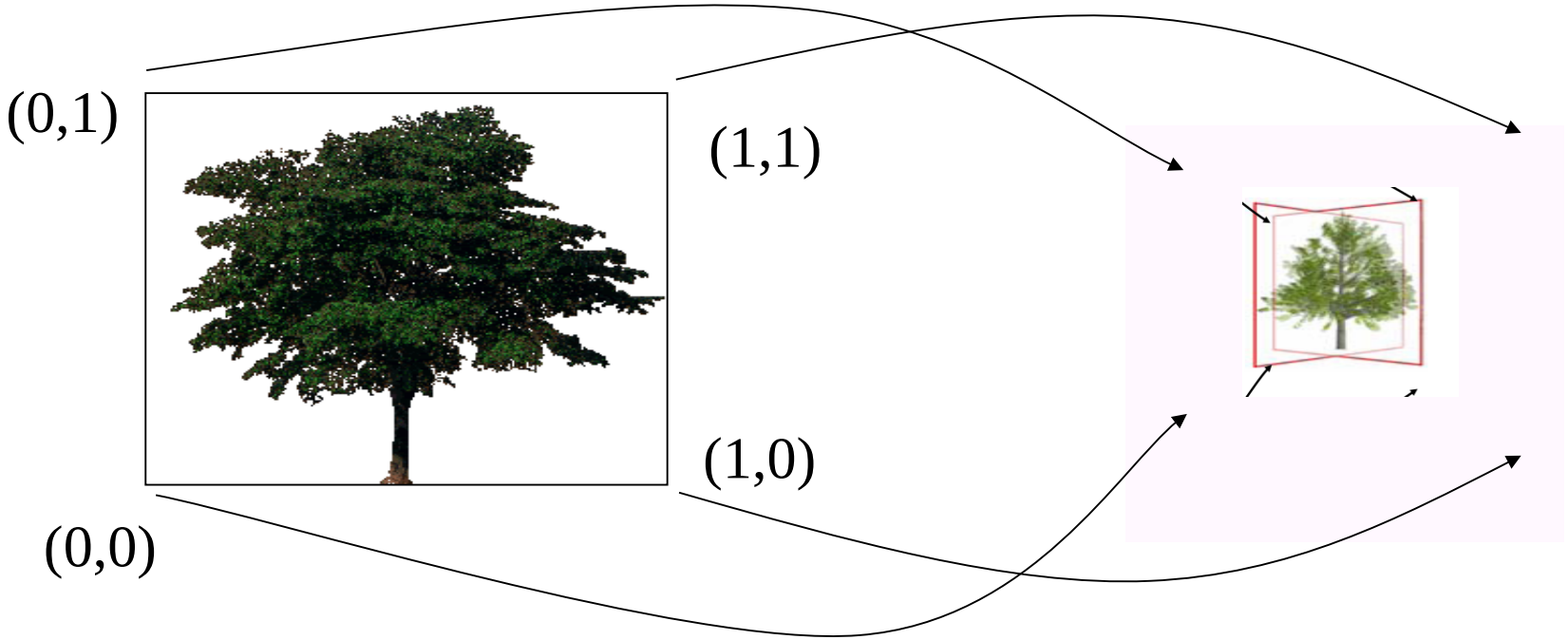
- Choix de la texture de l'objet se fait lors de son affichage

```
int Viewer::render(){

    // Choix de la texture
    gl.texture(papillon_texture);

    gl.model(Tquad);  // Transformation géométrique

    gl.draw(m_quad); // Affichage

    }
```

- Rq : si on ne souhaite pas de texture `gl.texture(0);`

Exemple d’utilisation : les billboards

- On bouge l’objet pour qu’il soit toujours face à la caméra
  - En TP : on va simplement en croiser 2

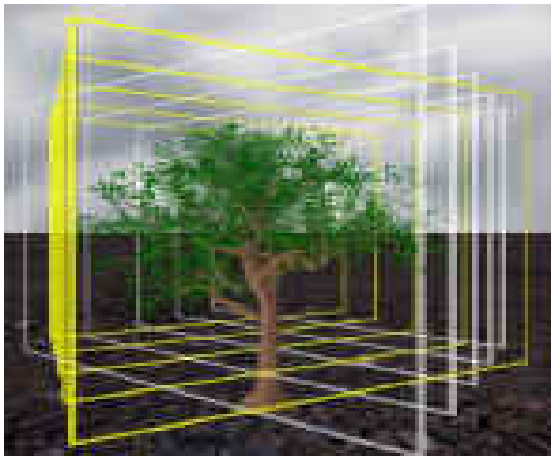


Plus généralement les arbres

Nombreux triangles par arbre (>100) et peu réaliste

2 quads par arbre

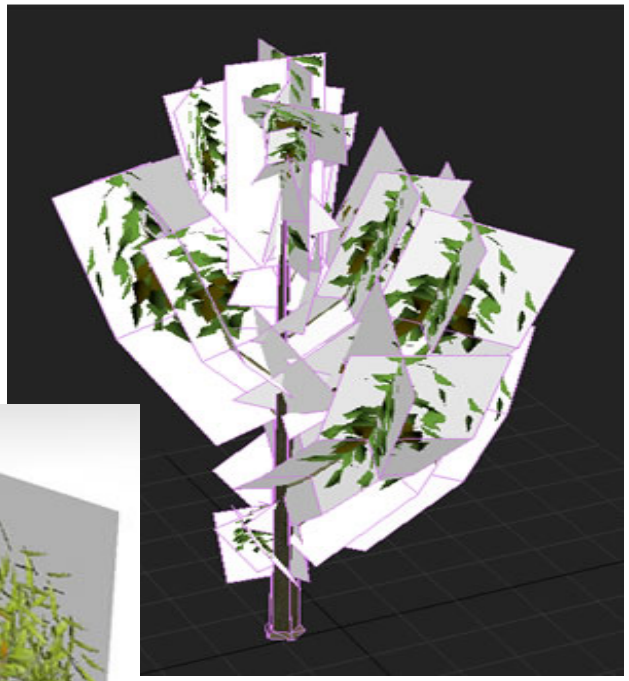
16 couches de quad par arbre

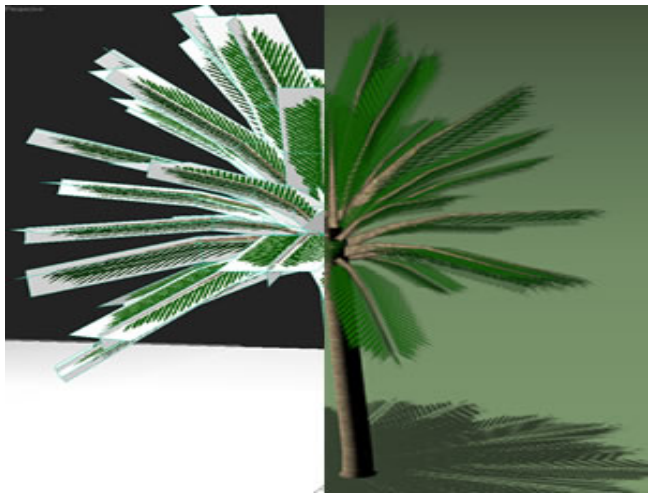


Plus généralement les arbres

Hiérarchie de billboards

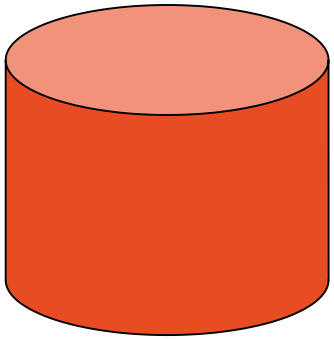
- 1 billboard par grande branche
- 1 billboard par petite branche
- 1 billboard par arbre





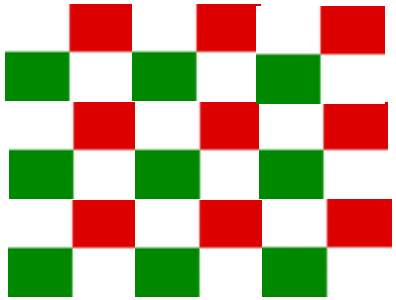
Texture = problème de la paramétrisation

Comment décider où chaque texel de la texture
se projette sur la géométrie ?



Géométrie

+



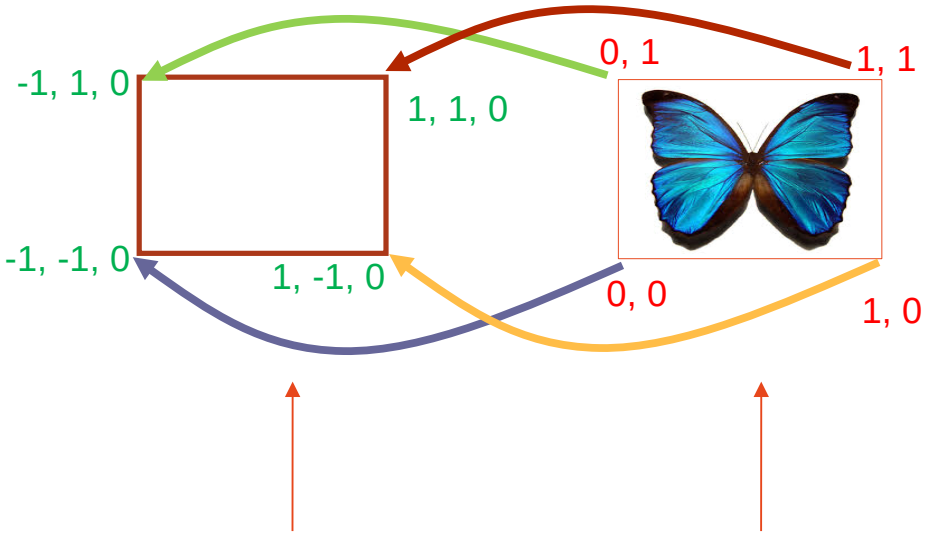
Texture

=

Objet texturé

Exemple - Quad

```
void Viewer::init_quad()
{
   m_quad = Mesh(GL_TRIANGLE_STRIP);
   m_quad.color( Color(1, 1, 1));

   m_quad.normal(  0, 0, 1 );

   m_quad.texcoord(0,0 );
   m_quad.vertex(-1, -1, 0 );

   m_quad.texcoord(1,0);
   m_quad.vertex(  1, -1, 0 );

   m_quad.texcoord(0,1);
   m_quad.vertex( -1, 1, 0 );

   m_quad.texcoord( 1,1);
   m_quad.vertex(  1,  1, 0 );
}
```



Exercices faits en TD / TP

- Rajouter les **coordonnées de texture aux formes de base**
  - Cube : 1 texture par face
  - Cylindre : 1 texture (sans dessus/dessous)
  - Cône : 1 texture (sans dessous)
  - Sphère : 1 texture

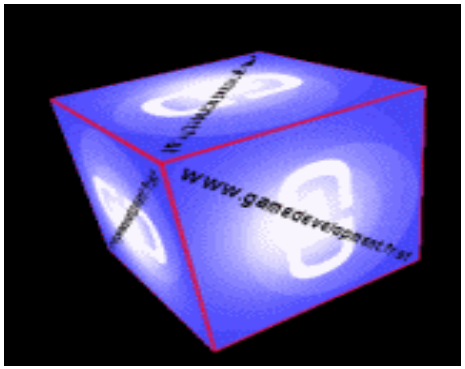
- Rajouter les textures sur le **vase créé par révolution**
  - 1 texture par facette carrée du vase

- Rajouter texture sur le **terrain fait avec carte de hauteur**

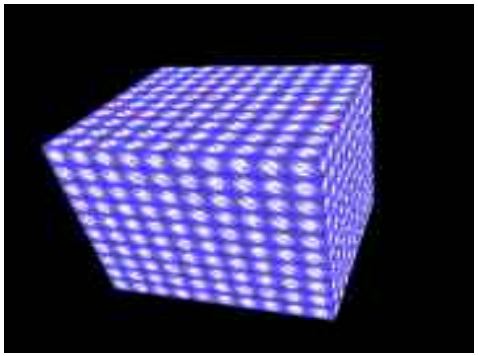
Coordonnées de texture

Les coordonnées de texture vont généralement de 0.0 à 1.0

On peut mettre des coordonnées supérieures, si on souhaite répéter la texture (activer GL_REPEAT)



Valeurs entre 0.0 et 1.0

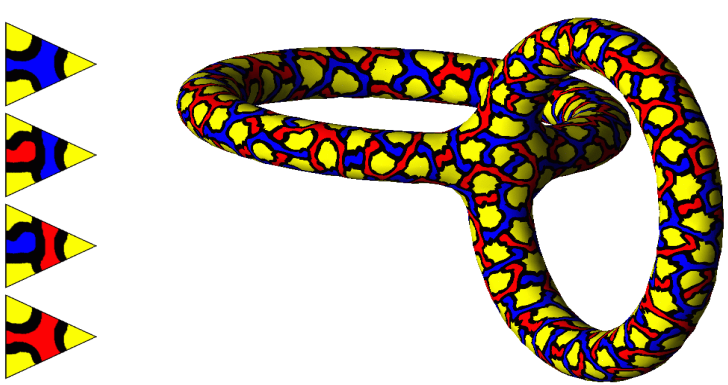


Valeurs entre 0.0 et 10.0 : texture répétée
10 fois en longueur et largeur

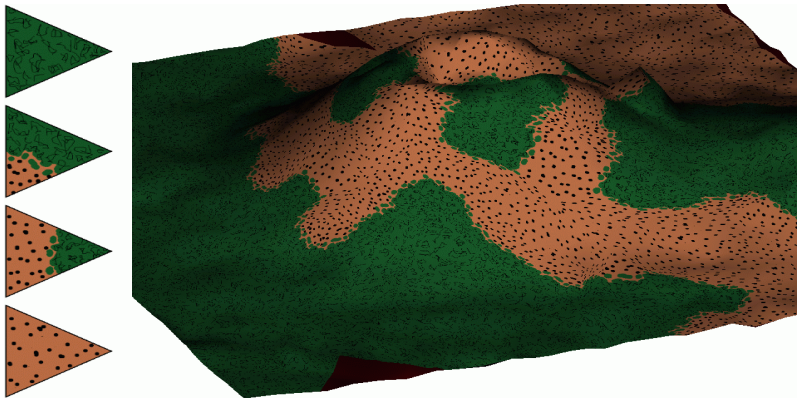
Répétition de texture

- Utile pour représenter des petits détails
  - crépi, routes, etc.

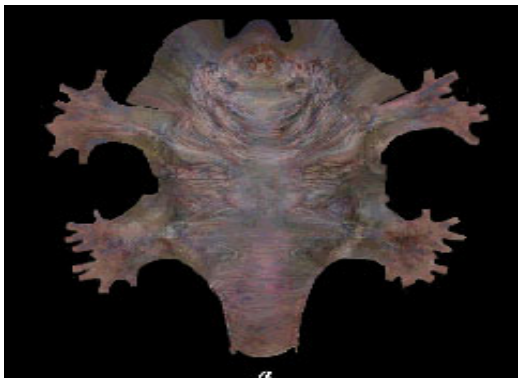
- Problème :
  - Textures répétitives : remarque la régularité
  - Solution : utiliser n textures avec des raccords continus et les combiner

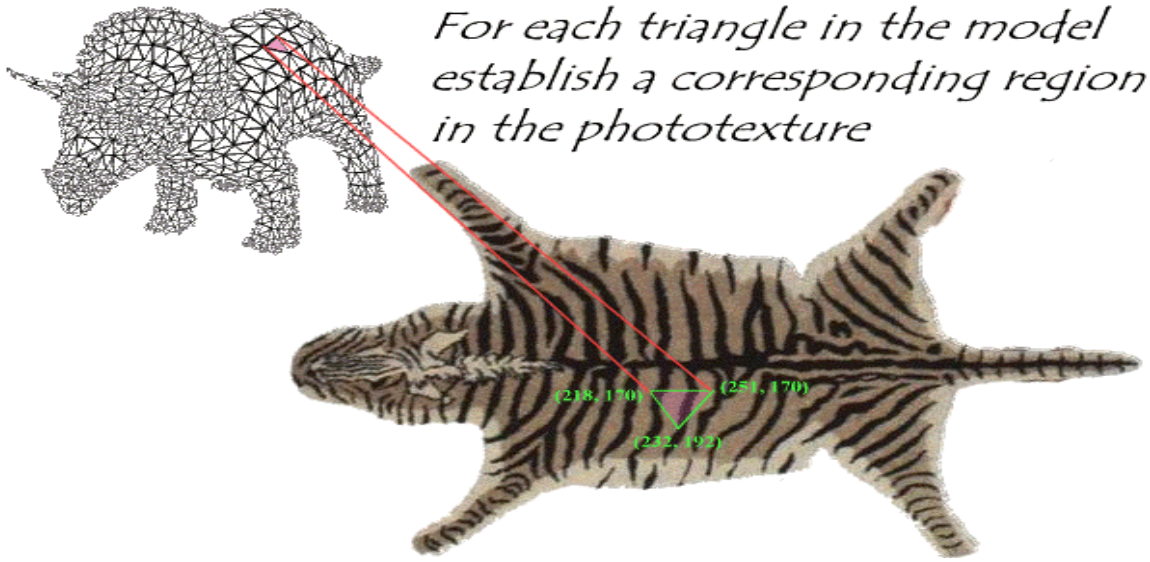


Neyret99



Problème de la paramétrisation





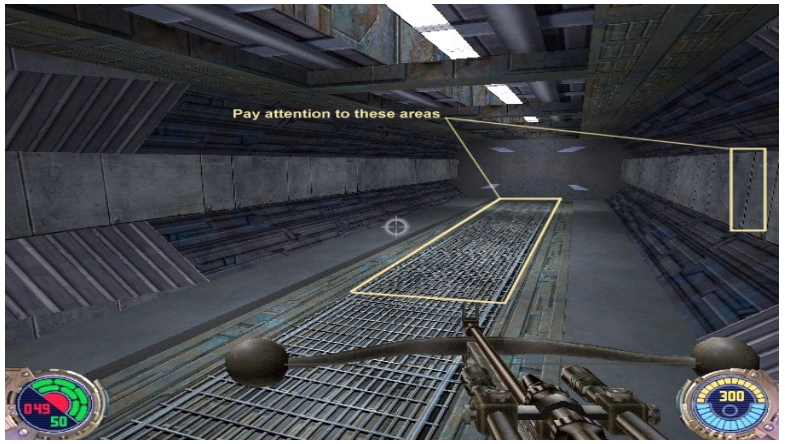
Un artiste peut créer une texture et paramétriser son modèle à la main

Atlas de texture

- Une texture peut être utilisée par plusieurs *mesh* en compactant différentes images

Aliasing : correspondance pixel / texel

Taille de la texture correspond rarement à la taille du polygone sur l'écran

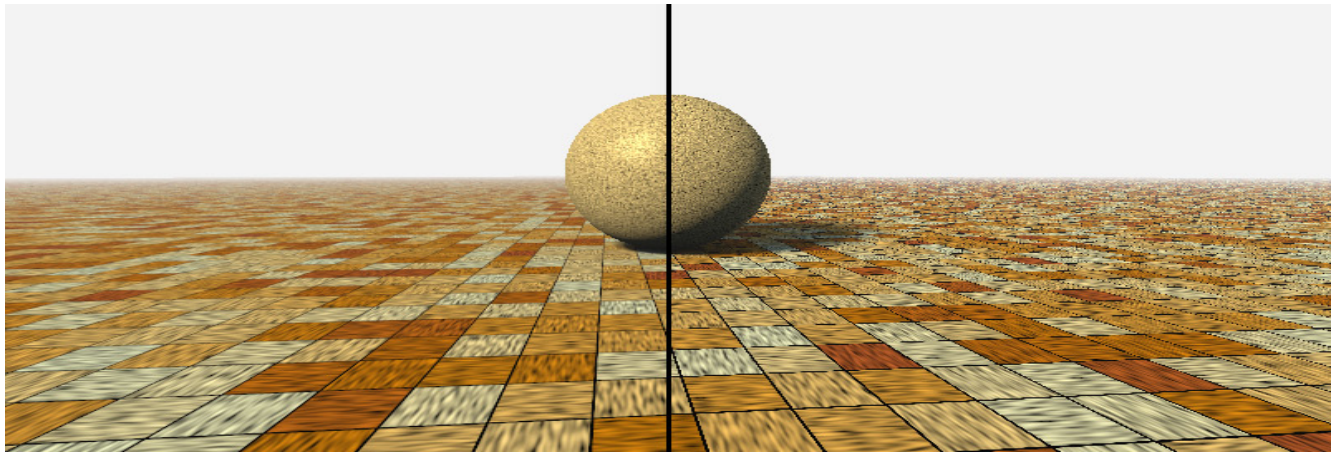


- Si la taille texture > taille polygone sur écran
  - Réduction de la texture = minification
  - Pixel = ensemble de texels

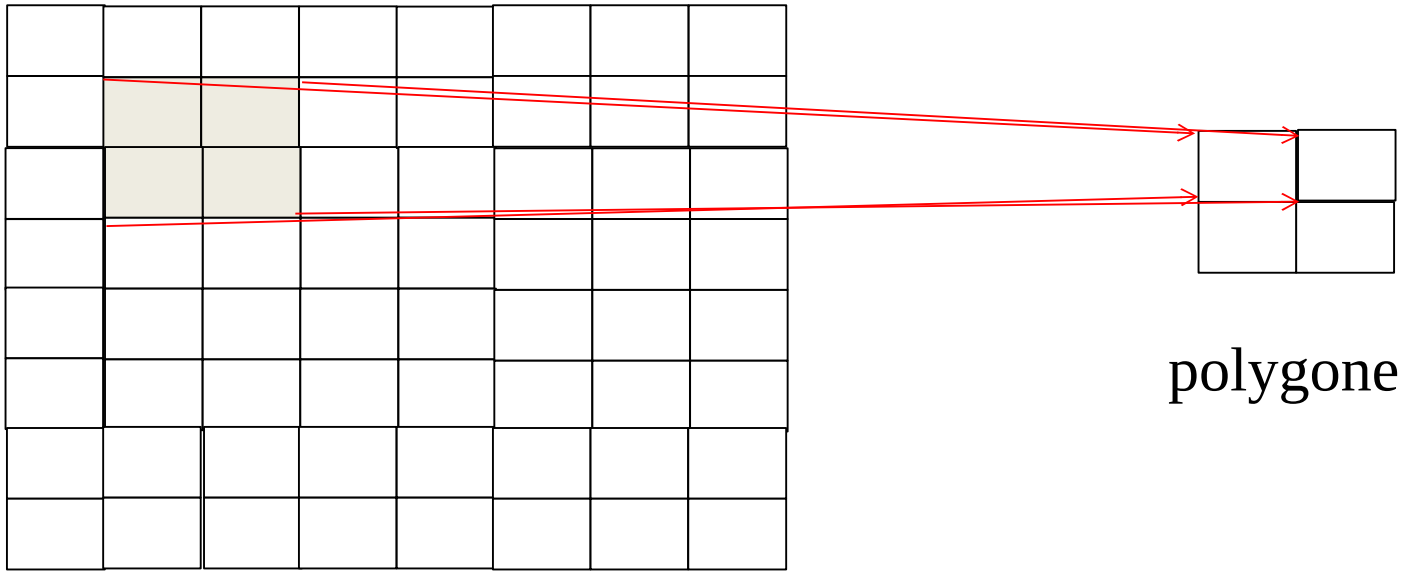
- Si la taille texture < taille polygone sur écran
  - Agrandissement de la texture = magnification
  - Pixel = portion de texel

Aliasing : correspondance pixel / texel

- Problème d’échantillonnage



Filtre de texture : minification

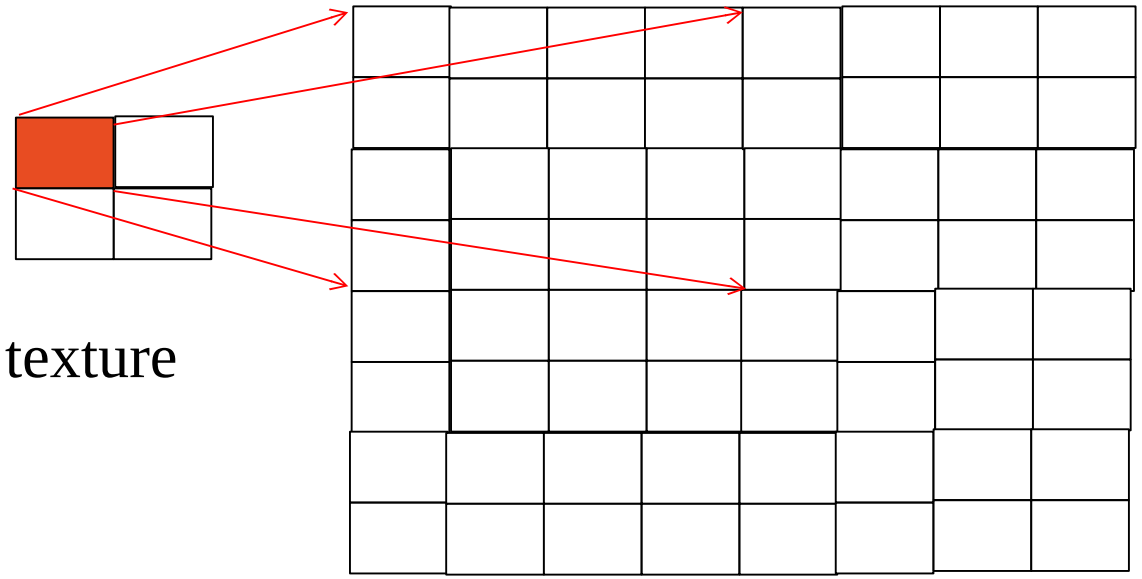


polygone

texture

pixel = ensemble de texels

Filtre de texture : magnification



texture

polygone

pixel = portion de texel

Filtre de texture

- Un certain nombre de paramètres à spécifier en OpenGL
- Spécification du type d’interpolation à faire
- Etc.

Mipmapping

- Contrôle du niveau de détail des textures avec les mipmaps
- L’idée est de construire une pyramide d’image à différentes résolutions

Mipmapping

- Texture de taille initiale 256 x 256 avec différents niveaux de détails

Mipmapping

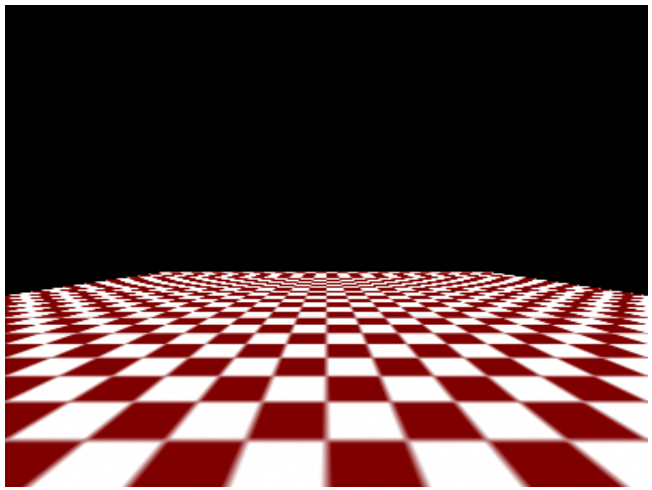
- OpenGL détermine l'image de texture à utiliser en fonction de la résolution du polygone

- Polygone proche de la caméra
  - grand nb de pixels sur écran = grande résolution
  - Utilisation de la texture de niveau de détail le + élevé

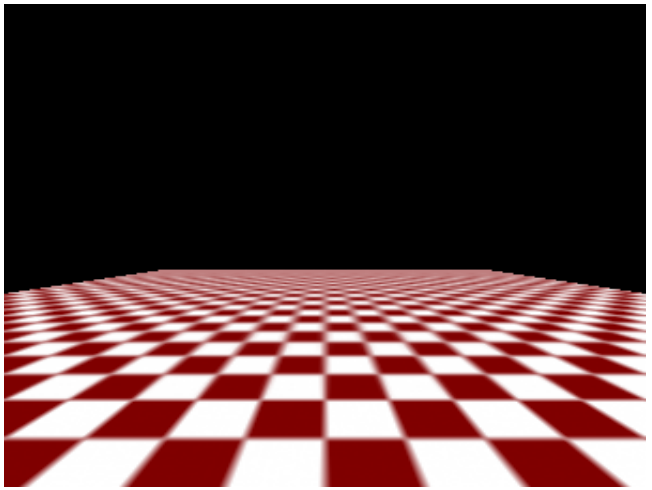
- Polygone loin de la caméra
  - Moins de pixels à l'écran
  - Utilisation de la texture de niveau de détail moins élevé

- Il faut spécifier les textures jusqu'à la résolution 1x1
  - Si texture initiale 128x128 (level 0), il faut définir 64x64 (level1), 32x32 (level 2), ... , 1x1 (level 7)

Mipmapping : exemple

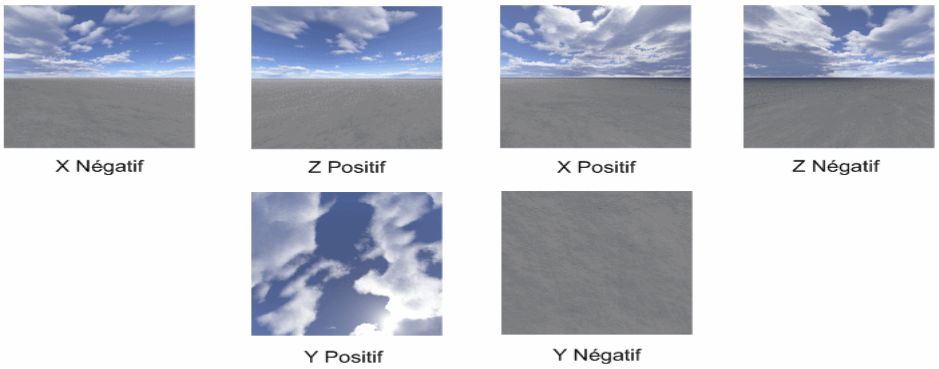


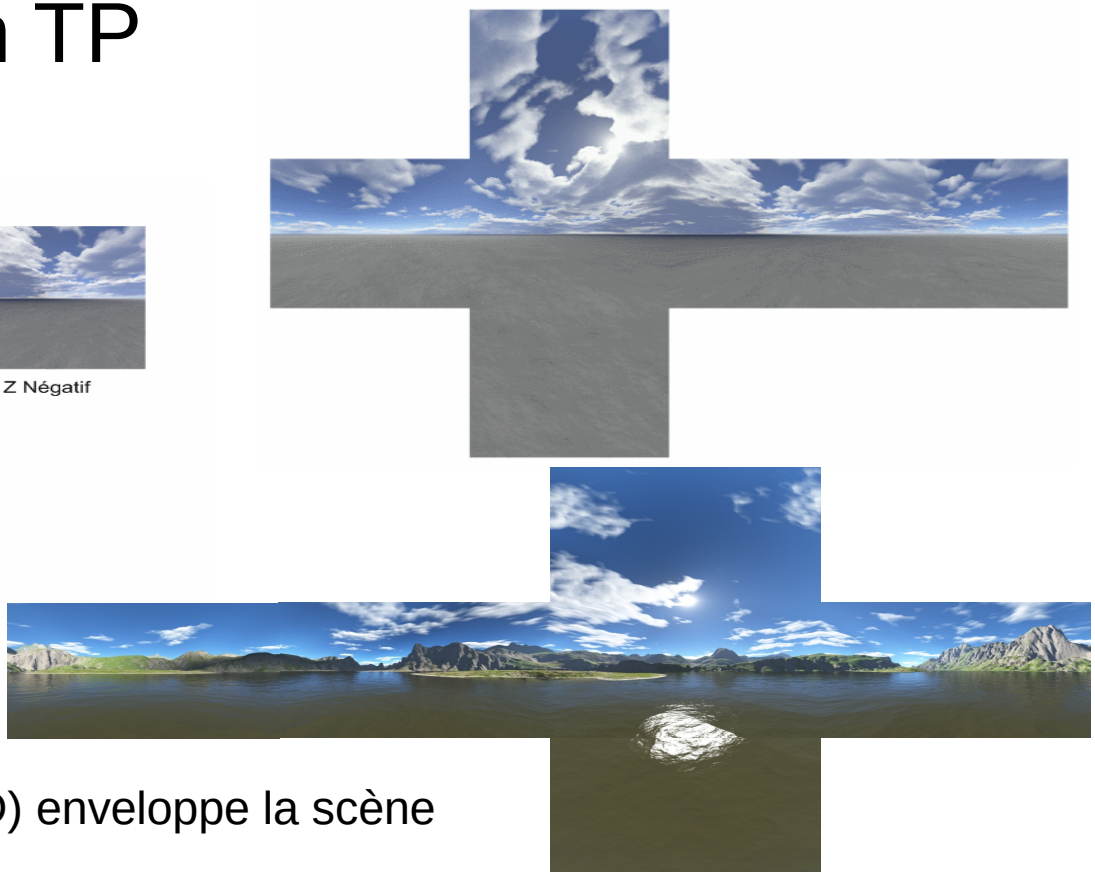
Sans mipmapping



Avec mipmapping

Cube map - à faire en TP





Un cube texturé par 6 images (textures 2D) enveloppe la scène

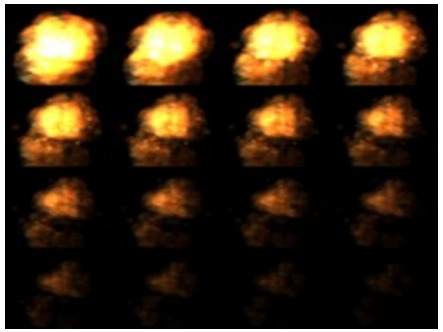
→ Impression d'espace

Cube map

- Permet de visualiser les reflets de l'environnement sur l'objet

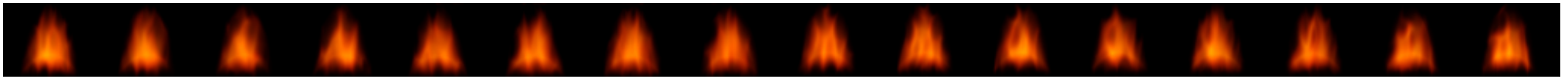
Texture animée

- N images côte à côte : effet d'animation
  - Feu, personnage marche, etc.
  - Mouvement doit être cyclique



- Changer coordonnées texture (u,v) en fonction du temps

  `m_quad.texcoord(num, u(t),v(t));` avec t le temps



Exercice : trouver coordonnées UV pour afficher le feu ci-dessus

Où trouver des textures ?

- Photographie
- Artistes
- Bibliothèque de textures sur le Web
- Techniques de génération automatique (non abordées en L2)

Pour résumer

- Texture
  - Outil très puissant
  - Simple
  - Réaliste

- Également de nombreux problèmes
  - Paramétrisation
  - Génération de textures
  - De près ne remplace pas la géométrie

Transparence

Opacité ou transparence

- Transparence pour la couleur d'un polygone ou les pixels d'une texture

- RGB → RGBA
  - Paramètre A (alpha) = opacité
    - 0.0 = transparent
    - … = semi-opaque
    - 1.0 = opaque

- Effet de transparence pour simuler les vitres, l'eau, etc.

Transparence

- Formule
  - Calcul pendant la rasterization
  - Fusion (*blending*) des valeurs chromatiques sources et des valeurs chromatiques de destination
  - C_buffer = A x C_polygone + (1 - A) x C_buffer

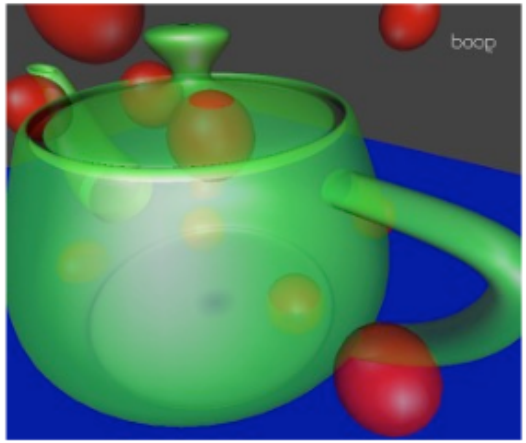
Transparence

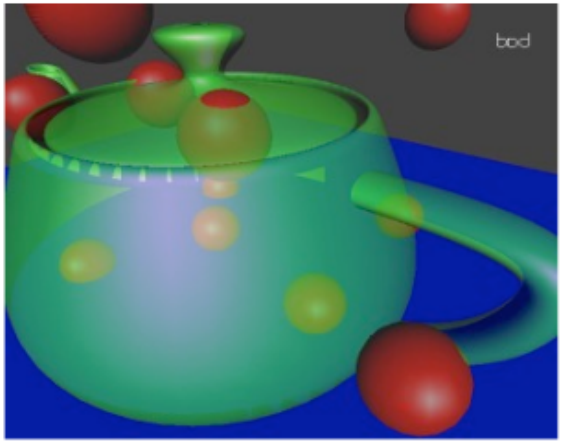
- Exemple les billboards
  - Les pixels blancs doivent être transparents

Transparence

Comment faire pour dessiner un objet transparent devant un objet opaque ?

- Il faut activer le **test de profondeur** pour afficher les objets de derrière à devant
  → Il faut connaître la couleur de l'objet opaque pour pouvoir faire le *blend* pour l'objet transparent





Transparence et texture – code gkit

```
int Viewer::render(){
    // Choix de la texture
    gl.texture(papillon_texture);   // texture opaque
    gl.model(Identity());  // Transformation géométrique
    gl.draw(m_quad); // Affichage

    // Choix de la texture
    gl.alpha_texture(papillon_texture); // texture transparente, par défaut alpha à 0.5
    gl.model(Translation(1, 0, 0));  // Transformation géométrique
    gl.draw(m_quad); // Affichage

    // Choix de la texture
    gl.alpha_texture(papillon_texture, 0.8); // texture transparente, alpha mis à 0.8
    gl.model(Translation(0, 1, 0));  // Transformation géométrique
    gl.draw(m_quad); // Affichage

}
```