Textures amb OpenGL

© Professors de VA

Grup MOVING – Dep. LSI – UPC

Introducció

- Texturació en OpenGL:
  - 2 parts:
    - Configuració de la textura
    - Aplicació a la geometria
  - La informació (“imatge”) i la manera en què influeix en la rasterització es configura (estats) via la utilització d’uns objectes especials anomenats *texture objects*

Texture objects

- Objecte que permet manipular les propietats d'una textura:
  - Contingut: bitmap
  - Tipus de textura: 1D, 2D, 3D, CUBE_MAP…
  - Com s'aplica: substitueix el color, el modula…
  - Com es comporta: ampliacions i reduccions
  - Com s'emmagatzema: prioritat, mip maps...

Texture objects

- Funcions principals:
  - Definició de la textura: *glTeximage*. Paràmetres que controla:
    - **Dimensió:** 1, 2 o 3
    - **Mides:** Potència de 2
    - **Amplada costat**
    - **Format intern:** GL_RGB, GL_RGBA …
    - **Imatge**

Texture objects

- Funcions principals:
  - Paràmetres de la textura: *glTexParameter*.
    Paràmetres que controla:
    - **Filtre ampliació**
    - **Filtre reducció**
    - **Mode wrapping**
    - **Color cantonada**
    - **Prioritat**
    - …

Ús de textures

- Tres etapes:
  - **Creació** de la textura:
    - *Creació:* glGenTexture, glBindTexture, glTexImage
    - *Definició paràmetres:* glTexParameter
  - **Dibuix** de les primitives texturades
    - *Activació:* glEnable i glBindTexture
    - *Definició funció texturació:* glTexEnvi
    - *Generació coordenades:* glTexCoord o automàtiques
  - **Destrucció** textures: glDeleteTextures

Ús de textures

// 1. **Activar el texture mapping desitjat**
// Només pot estar activat un mode: GL_TEXTURE_1D, 2D o 3D

```
glEnable(GL_TEXTURE_2D);
```

// **2. Activar el texture object corresponent**

```
glBindTexture(GL_TEXTURE_2D, id);
```

// **3. Establir la funció de texturació**

```
glTexEnvi(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_REPLACE);
```

// **4. Dibuixar la primitiva**

```
glBegin(GL_POLYGON);
glTexCoord2d(0,0);
glVertex3d(...);
...
```

// o utilitzant coordenades automàtiques

Creació de l'objecte textura

- Generar un nou nom:
  - void **glGenTextures**(1, &*texName*);
    - Crea una textura (1) i emmagatzema el seu identificador a *texName*
- Activar la textura
  - void **glBindTexture**(GL_TEXTURE_2D, *texName*);
    - Les següents operacions de textures actuaran sobre *texName*.

Creació de l'objecte textura

- Introducció de les dades:
  - void **glTexImage2D**(GLenum *objective,* GLint *level,* GLint *internalFormat,* GLsizei *width,* GLsizei *height,* GLint *border,* GLenum *format*, GLenum *type,* GLvoid* *pixels*);
    - *objective:* GL_TEXTURE_2D
    - *level:* 0 (nivells de mip mapping)
    - *internalFormat* i *format:* GL_RGB o GL_RGBA
    - *width i height:* de la forma 2^m+2b (mín 64x64)
    - *border:* 0 o 1
    - *type:* de les dades que passem a *pixels* (GL_BYTE, GLFLOAT...)
    - *pixels:* Array de bytes amb valors del tipus *tipus*

Creació de l'objecte textura

- Altres formes de posar les dades
  - void **glTexSubImage2D**(GLenum *target*, GLint *level*, GLint *xoffset*, GLint *yoffset*, GLsizei *width*, GLsizei height,GLenum *format*, GLenum *type*, const GLvoid **pixels*);
    - Substitueix una regió rectangular d'una textura ja definida per un *buffer* en memòria principal.

Creació de l'objecte textura

- Altres formes de posar les dades. A partir de la informació generada:
  - void **glCopyTexImage2D**(GLenum target, GLint level, GLenum internalFormat, GLint x, GLint y, GLsizei width, GLsizei height, GLint border);
    - Defineix la textura a partir d'una regió rectangular del GL_READ_BUFFER actiu (com el *glCopyPixels* però els *pixels* van a memòria de textura en comptes del *framebuffer*).
  - void **glCopyTexSubImage2D**(GLenum *target*, GLint *level*, GLint *xoffset*, GLint *yoffset*, GLint *x*, GLint *y*, GLsizei *width*, GLsizei *height*);
    - Substitueix una regió rectangular d'una textura ja definida per una regió rectangular.

Creació de l'objecte textura

- Altres formes de posar les dades. Gestió de la informació emmagatzemada:
  - int **gluBuild2DMipmaps**(GLenum *target*, GLint *components*, GLint *width*, GLint *height*, GLenum *format*, GLenum *type*, const void **data*);
    - Construeix mipmaps a partir d'un buffer en memòria principal (no admet voreres).

Funcions de textura

- Determinen com es combinen el color de la textura amb el color original del fragment per tal de generar un nou color.
- Mètodes principals:
  - GL_REPLACE: Els valors de textura reemplacen els del fragment
  - GL_MODULATE: La textura *escala* el color de l'objecte. Habitualment utilitzat per a efectes d'il·luminació
  - GL_DECAL: Es barreja el color de la textura amb el del fragment
  - GL_BLEND: Es barreja el color del fragment amb el de la textura i un tercer color: GL_TEXTURE_ENV_COLOR

Funcions de textura

- El resultat es calcula en funció dels valors del fragment actual (un cop calculada la il·luminació i de la textura

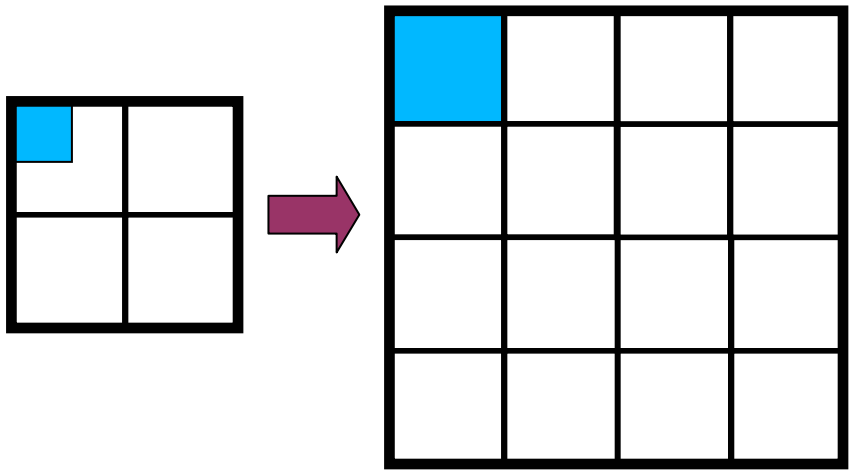
Format intern base	GL_REPLACE	GL_MODULATE	GL_DECAL	GL_BLEND
GL_ALPHA	$C = C_f$	$C = C_f$		$C = C_f$
	$A = A_t$	$A = A_f\ A_t$		$A = A_f\ A_t$
GL_LUMINANCE	$C = L_t$	$C = C_f\ L_t$		$C = C_f\ (1- L_t) + C_c\ L_t$
	$A = A_f$	$A = A_f$		$A = A_f$
GL_LUMINANCE_ALPHA	$C = L_t$	$C = C_f\ L_t$		$C = C_f\ (1- L_t) + C_c\ L_t$
	$A = A_t$	$A = Af\ A_t$		$A = A_f\ A_t$
GL_INTENSITY	$C = I_t$	$C = C_f\ I_t$		$C = C_f\ (1- I_t) + C_c\ I_t$
	$A = I_t$	$A = A_f\ I_t$		$A = A_f\ (1- I_t) + Ac\ I_t$
GL_RGB	$C = C_t$	$C = C_f\ C_t$	$C = C_t$	$C = C_f\ (1-Ct)+ C_c\ C_t$
	$A = A_f$	$A = A_f$	$A = A_f$	$A = A_f$
GL_RGBA	$C = C_t$	$C = C_f\ C_t$	$C = C_f\ (1-At) + Ct\ At$	$C = C_f\ (1- C_t)+ C_c\ C_t$
	$A = A_t$	$A = A_f\ A_t$	$A = A_f$	$A = A_f\ A_t$

Funcions de textura

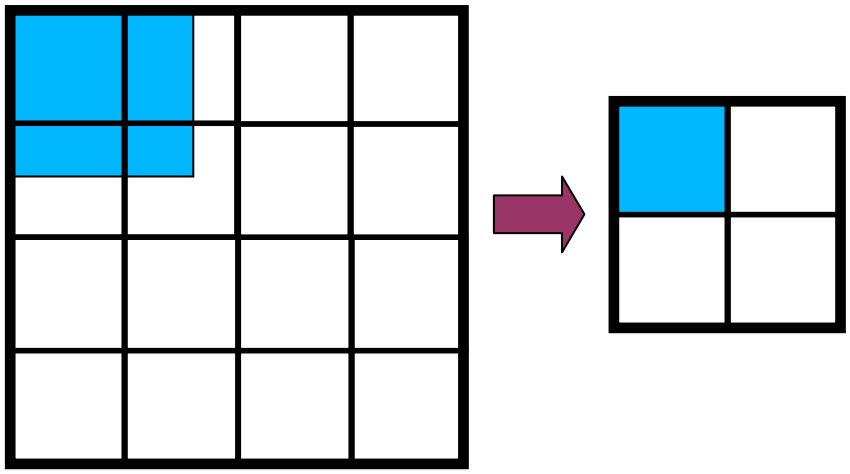
- Definir el comportament en filtrat:
  - void **glTexParameteri**(GL_TEXTURE_2D, *filtre*, *filtrat*);
    - *filtre:* ampliació (GL_TEXTURE_MAG_FILTER) o reducció (GL_TEXTURE_MIN_FILTER)
    - *filtrat:* agafar el més proper (GL_NEAREST) o una interpolació (GL_LINEAR)
  - Si no es defineixen els filtres pot no veure's res!!!

Funcions de textura

Ampliació



Reducció



Dibuixat escena

- S'ha d'indicar com s'alinea la textura relativament als *fragments* als quals s'aplicarà abans de pintar-la
- Si tenim un a textura de dues dimensions, les coordenades de textura van de 0.0 a 1.0 en ambdós eixos.
- Si l'objecte és gran, cal especificar com es comportarà la textura més enllà dels límits.

Dibuixat escena

- Definir coordenades:
  - Primer es posen les coordenades de textura del vèrtex:
    - **glCoord2f** (coordTexX, coordTexY);
  - Després es projecta el vèrtex:
    - **glVertex3f** (coordX, coordY, coordZ);

Dibuixat escena

- Comportament més enllà de [0.0, 1.0]:
  - void **glTexParameteri**(GL_TEXTURE_2D, *param, tipus*);
    - *param:* s (GL_TEXTURE_WRAP_S) o t (GL_TEXTURE_WRAP_T)
    - *tipus:* repetir (GL_REPEAT) o tallar (GL_CLAMP)

Generació de coordenades automàtiques

- Definició explícita de coordenades té algunes implicacions:
  - Geometries no rectangulars han de tenir coordenades per vèrtex no trivials
  - S'han d'emmagatzemar amb el model
  - Alguns efectes (mirall) requereixen la modificació dinàmica

Generació de coordenades automàtiques

- OpenGL proporciona 3 modes de definició automàtica de coordenades de textura:
  - En espai d'objecte:
    - La textura està *enganxada* a l'objecte
  - En espai d'ull
    - L'objecte es mou *dins* l'espai de la textura
  - Sphere map
    - Basada en el vector de reflexió

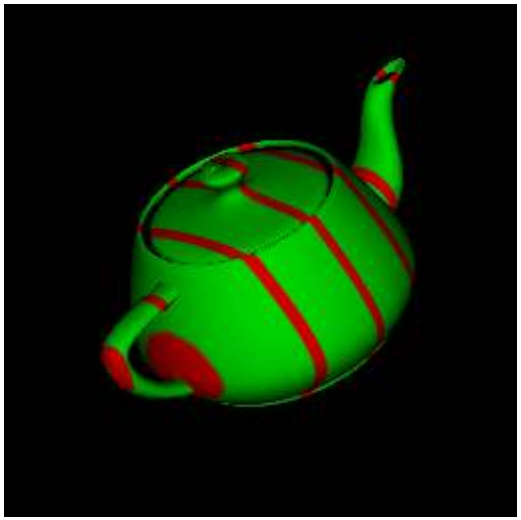
Generació de coordenades automàtiques

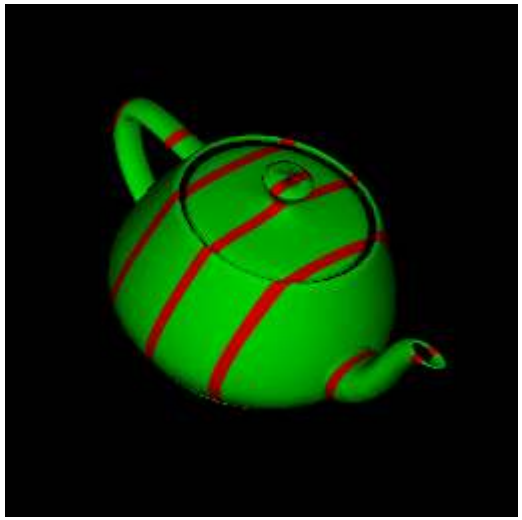
- GL_OBJECT_LINEAR:
  - Cada coordenada de textura es calcula com *ax+by+cz+dw*, on *(x,y,z,w)* són les coordenades del vèrtex especificades amb *glVertex*, i *(a,b,c,d)* és el pla associat a la coordenada
    - El pla s'especifica amb GL_OBJECT_PLANE: Atenció: *(x,y,z,w)* i *(a,b,c,d)* estan en coordenades de món

Generació de coordenades automàtiques

- GL_OBJECT_LINEAR: Exemple:

```
GLfloat params = {A,B,C,D};
glTexGenfv(GL_S, GL_OBJECT_PLANE, params);
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE,
  GL_OBJECT_LINEAR);
glEnable(GL_TEXTURE_GEN_S);
```





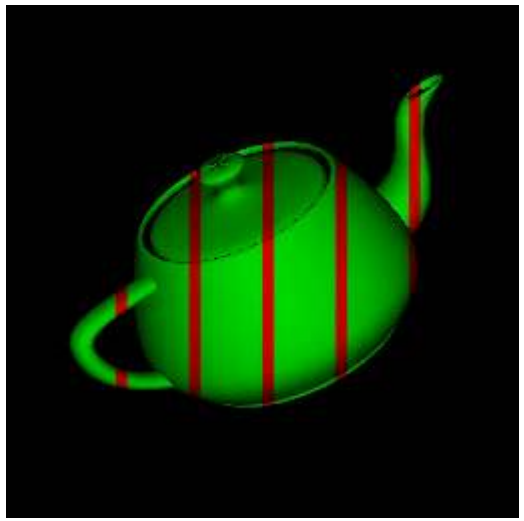
Generació de coordenades automàtiques

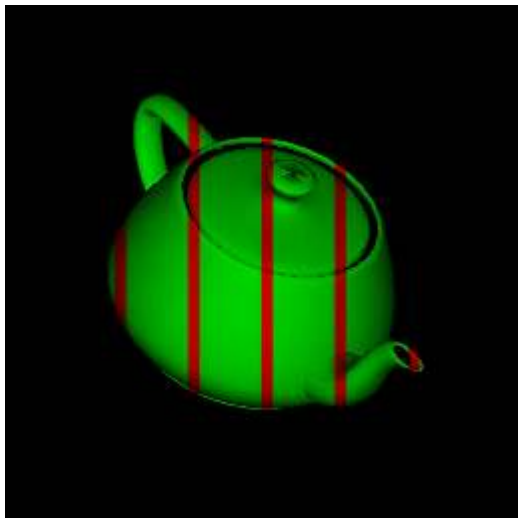
- GL_EYE_LINEAR:
  - Cada coordenada de textura es calcula com *a'xe+b'ye+c'ze+d'we*, on *(xe,ye,ze,we)* són les coordenades del vèrtex en *coordenades d'observador* (per tant, després d' aplicar la *MODELVIEW* actual), i *(a',b',c',d')* és el pla associat a la coordenada.
    - El pla s'especifica amb GL_EYE_PLANE multiplicat per la inversa de la matriu *MODELVIEW* activa en el moment en què es crida a *glTexGen*

Generació de coordenades automàtiques

- GL_EYE_LINEAR. Exemple:

```
GLfloat params = {A,B,C,D};
glTexGenfv(GL_S, GL_EYE_PLANE, params);
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE,
  GL_EYE_LINEAR);
glEnable(GL_TEXTURE_GEN_S);
```





Generació de coordenades automàtiques

- GL_SPHERE_MAP:
  - Serveix per a calcular reflexions de forma aproximada.
  - Es basa en la idea de que si un objecte és molt petit en relació al seu entorn, el que s'hi reflexa depèn principalment de la direcció del vector reflectit
  - El nom general d'aquesta tècnica és *environment mapping* i el mapeig pot ser de textures esfèriques, cúbiques, parabòliques...

Generació de coordenades automàtiques

- GL_SPHERE_MAP. Exemple:

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE,
  GL_SPHERE_MAP);
glEnable(GL_TEXTURE_GEN_S);
```