Descriptor Library

Lo scene-graph di XCModel 4.0

Giulio Casciola

Università di Bologna

Cos'è la libreria descriptor

- E' una libreria costruita per descrivere una scena
- Le sue funzioni devono essere chiamate da uno script in linguaggio C
- L'esecuzione di tale codice costruisce lo scene-graph che viene salvato in un file (.md)

Classi di oggetti

- Ogni tipo di dato nello scene-graph è di tipo ITEM ed appartiene ad una delle seguenti classi:
- A primitive, list, hierarchy;
- B primitive, list, hierarchy, light;
- C attribute.

dove

A: oggetti che possono avere un attributo;

B: oggetti che possono essere trasformati;

C: sono oggetti attributo

Classi di oggetti

- primitive: consiste di superfici o insiemi di superfici NURBS trimmate o non trimmate;
- list: è una lista di primitive;
- hierarchy: è una lista di primitive e list (graph) che descrive come sono clusterizzate le primitive;
- light: differenti tipi di luce;
- attribute: proprietà materiale e colore delle primitive;

Instanziare ITEM di classe A

- read_nurbs (char *filename)
- create_nurbs (...)
- read_obj (char *filename, char *name)
- create_copy (ITEM item, char *name)
- create_list (ITEM item, char *name_list)
- insert_in_list (ITEM list, ITEM item)
- save_scene (char *name, char *title, ITEM it)

Esempio:

```
ITEM towerA,towerB;
  towerA=read_nurbs(“torre.db”);
  towerB=create_copy(towerA, ”towerB”);
```

Attributi: ITEM di classe C

- create_attribute(char *name);
- set_color(ITEM attribute, r, g, b, ka, kd);
- set_reflectivity(ITEM attribute, ks, n)
- set_trasparency(ITEM attribute, maxt, mint, tpwr, rifr)
- set_attribute(ITEM item, ITEM attribute);

Esempio:

```
ITEM towerA, attr_towerA;
  attr_towerA=create_attribute("torre_nera");
  set_color(attr_towerA,1 ,0.06, 0.03, 0.5, 0.5);
  set_reflectivity(attr_towerA, 0.4, 2);
  set_attribute(towerA, attr_towerA);
```

Definizione luci

- set_ambient_light(r, g, b, intensity)
- set_background(r, g, b, is_refl)
- create_distant_light(dx, dy, dz, r, g, b, intensity, name);
- set_distant_light(dx, dy, dz, r, g, b, intensity, ITEM);

Esempio:

```
ITEM  lightP;
  set_ambient_light(1, 1, 1, 0.7);
  lightP=create_distant_light(35.0, 5.0, 7.0, 1, 1, 1, 0.7, “luce1”);
  ……..
  set_distant_light(35.0, 5.0, 7.0, 1, 1, 1, 1.2, lightP);
```

Definizione luci

- create_point_light(cx, cy, cz, r, g, b, intensity, range, name);
- set_point_light(cx, cy, cz, r, g, b, intensity, range, name);
- create_warn_light(..., ITEM);
- set_warn_light(..., ITEM);

Esempio:

```
ITEM  lightP;
  lightP=create_point_light(35.0, 5.0, 7.0, 1, 1, 1, 0.7, 0, “luce1”);
  ..................
  set_point_light(35.0, 5.0, 7.0, 1, 1, 1, 1.2, 0, lightP);
```

Trasformazioni

Trasformazioni geometriche di scala, traslazione, rotazione intorno agli assi coordinati e shear, sia assolute che relative, per oggetti di classe B:

- set_scale(item, sx, sy, sz)
- set_scale_abs(item, sx, sy, sz)
- set_x_rotate(item, angle, deg_fg)
- set_x_rotate_abs(item, angle, def_fg)
- .....

Texture 3D

Per applicare una texture 3D ad un oggetto della classe A si devono eseguire le seguenti operazioni:

1. creare un oggetto;
2. trasformare l'oggetto per portarlo nello spazio texture;
3. assegnare la texture scelta;
4. applicare la trasformazione inversa all'oggetto;
5. posizionare l'oggetto nella scena.

Texture 3D

set_angle_texture (...)

set_cross_texture (...)

set_cube_texture (...)

set_gradient_texture (...)

set_layer_taexture (...)

set_octant_texture (...)

set_shade_texture (...)

set_triangle_texture (...)

set_weight_rand_texture (...)

set_wood_texture (...)

Texture Mapping

set_domain_texture(ITEM item, ITEM attribute,
char *image_name, float umin, float
umax, float vmin, float vmax, int rflag,
float ka, float kd)

set_projection_texture(...)

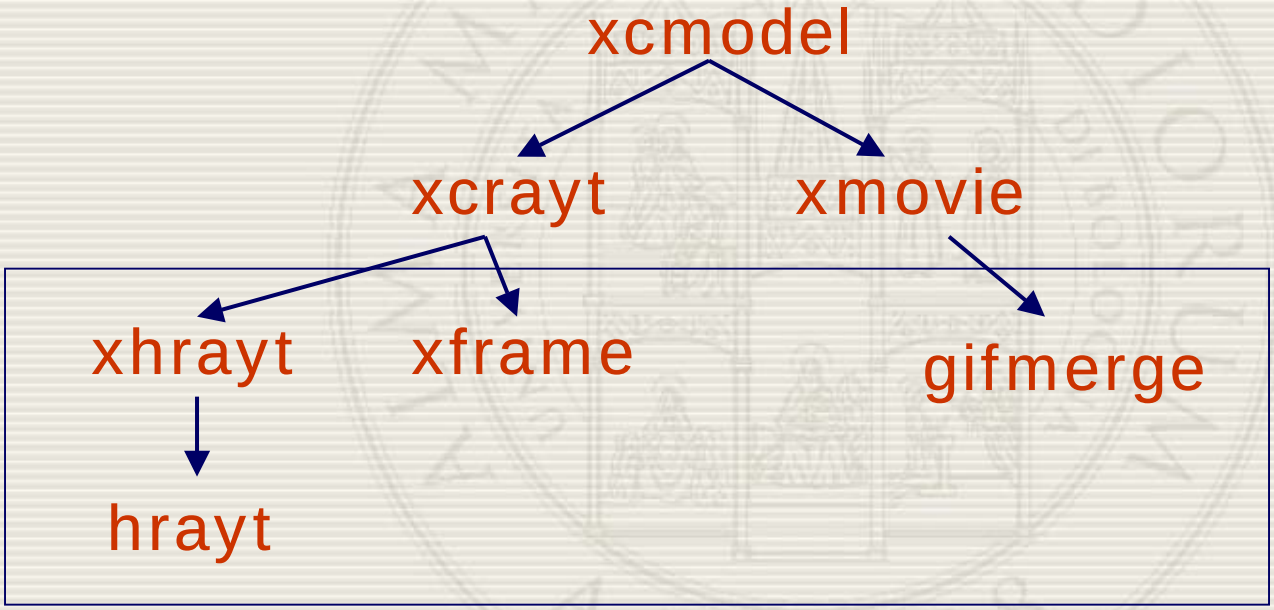
set_border_texture(...)

Bump Mapping

```
set_bump_texture(ITEM item, ITEM attribute,
        char *image_name, float umin, float umax,
        float vmin, float vmax, int rflag, float pa,
        float pd)
```

Lo script C per lo scene graph

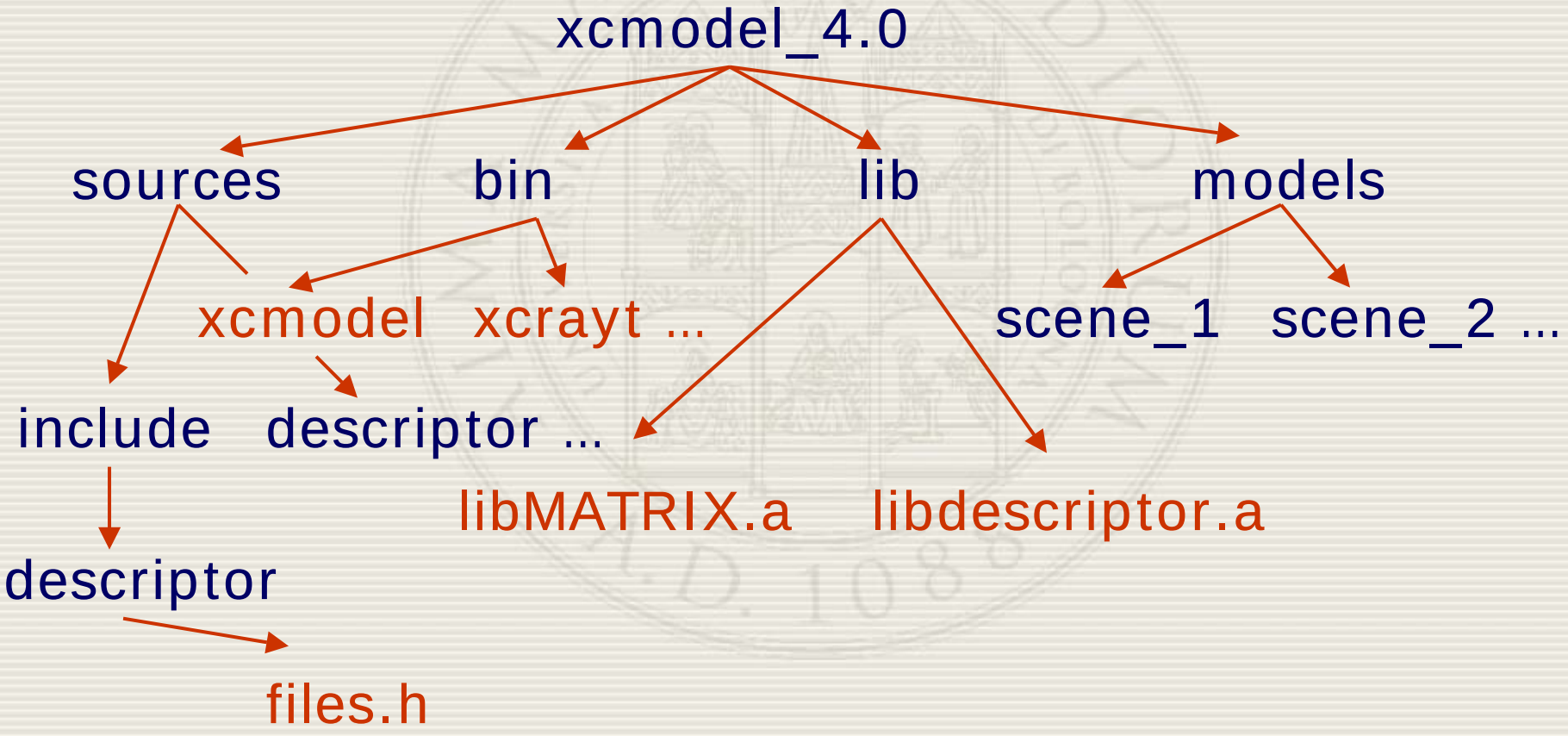
Il pacchetto XCRayt di XCModel si compone dei seguenti eseguibili nella dir. bin:



Chiamati in modo trasparente

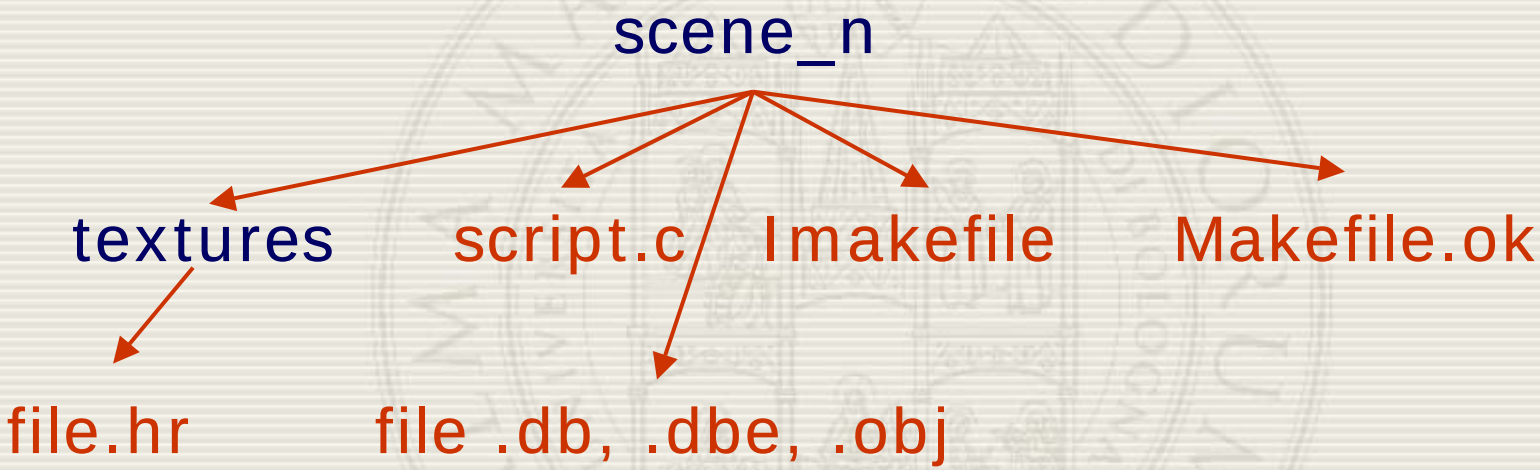
Lo script C per lo scene graph

L'organizzazione delle directory e':



Lo script C per lo scene graph

L'organizzazione delle directory scene_n:



Successivamente la directory scene_n conterrà file con le seguenti estensioni:
.md, .vw, .arg, .sta, .hr, .hra, .ani, …

Uno script C di esempio

```
#include “descriptor/descriptor.h”

int main(void) {

  ITEM     objectB;
  ITEM     tower;

  /* background setting */
  set_background(0.6,0.7,0.8,TRUE);

  /* ambient light setting */
  set_ambient_light(1,1,1,0.7);

  /* distant light creation */
  create_distant_light(0,0,1, 1,1,1, 0.7,"distant_light");

  /* point light creation */
  create_point_light(35.68,5.21,7.49, 1,1,1, 0.7, 0, "point_light");

  /* spot light creation */
  create_warn_light(1,0,0, 0,1,0, 1,1,1, 0.8,40, 5, 70, 1,0,0,1,5,
            "warn_light");
```

Uno script C di esempio

```
/* attribute definition */
 objectB=create_attribute("black_chess");
 set_color(objectB,1,0.062,0.031,0.5,0.5);
 set_reflectivity(objectB,0.4,2);

 tower=create_list(read_nurbs("tower.db"), "tower");

 set_scale(tower,1.0/0.26,1.0/0.26,1);

 /* textute mapping */
 set_projection_texture(tower,objectB,"marble_5_black.hr",0,FALSE,0.6,0.8);

 /* tower repositioning */
 set_scale(tower,0.26,0.26,1);

 /* tower positioning in the scene */
 set_translate(tower, 0.25 ,0.25, 0);
 set_translate(tower, 0, 1.5, 0);

 save_scene("chess_tower.md", "tower", tower);
 return(0);
}
```

Come compilare

Lo script C realizzato deve includere:

#include “descriptor/descriptor.h”

oltre alla definizione completa dello scene graph mediante chiamate a funzioni della libreria descriptor.

Viene fornito un Imakefile per creare il Makefile che compilerà e lancerà l’esecuzione dello script.

xmkmf per creare il Makefile

make per compilare ed eseguire lo script

File .md generato

HEADER: tower 3 2 1 #nome scena, n.luci, n.attributi+1, n.superfici
AMBIENT COLOR: 1.000000e+00 1.000000e+00 1.000000e+00
AMBIENT INTENSITY: 7.000000e-01
BKG COLOR: 6.000000e-01 7.000000e-01 8.000000e-01
BKG IS REFL: 1
LIGHT N. 0
NAME: distant_light
TYPE: 0
LOCATION: 0.000000e+00 0.000000e+00 0.000000e+00
DIRECTION: 0.000000e+00 0.000000e+00 1.000000e+00
COLOR: 1.000000e+00 1.000000e+00 1.000000e+00
INTENSITY: 7.000000e-01
CONCENTRATION: 0
MAX RANGE: 0.000000e+00
CONE: -1.000000e+00
FLAP: 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
LIGHT N. 1
NAME: point_light
TYPE: 1
LOCATION: 3.568000e+01 5.210000e+00 7.490000e+00
DIRECTION: 1.000000e+00 0.000000e+00 0.000000e+00
COLOR: 1.000000e+00 1.000000e+00 1.000000e+00
INTENSITY: 7.000000e-01
CONCENTRATION: 0
MAX RANGE: 0.000000e+00
CONE: -1.000000e+00
FLAP: 0 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00

File .md generato

LIGHT N. 2
NAME: warn_light
TYPE: 2
LOCATION: 1.000000e+00 0.000000e+00 0.000000e+00
DIRECTION: 0.000000e+00 1.000000e+00 0.000000e+00
COLOR: 1.000000e+00 1.000000e+00 1.000000e+00
INTENSITY: 8.000000e-01
CONCENTRATION: 5
MAX RANGE: 4.000000e+01
CONE: 3.420201e-01
FLAP: 1 0.000000e+00 0.000000e+00 1.000000e+00 5.000000e+00
ATTRIBUTES N. 1
NAME: black_chess
COLOR: 1.000000e+00 6.200000e-02 3.100000e-02
5.000000e-01
5.000000e-01
4.000000e-01
INDEX: 1.000000e+00 1.000000e+00
2
0.000000e+00 0.000000e+00
0
SURFACE N. 0
NAME: tower.db
IS SURFACE: 1
NORMAL INSIDE: 0
EXTENT:
-2.001000e-01 -2.001000e-01 -1.000000e-04
2.001000e-01 2.001000e-01 6.401000e-01

File .md generato

HIERARCHY
OBJECT TYPE: 1
OBJECT NAME: tower
OBJECT TYPE: 3
PRIMITIVE NAME: tower.db
GEO SHAPE: 11
TRANSFORM MATRIX TYPE: 1
TRANSFORM MATRIX:
1.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 1.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 1.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00
16
0
TEXTURE MATRIX TYPE: 1
TEXTURE MATRIX:
3.846154e+00 0.000000e+00 0.000000e+00
0.000000e+00 3.846154e+00 0.000000e+00
0.000000e+00 0.000000e+00 1.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00
TEXTURE DATA SIZE = 1
TEXTURE DATA TYPE: 3
0 0
0 0
2.352941e-03 3.137255e-03
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
grid.hr
ATTRIBUTE N. 1
NURBS N. 0
OBJECT TYPE: 0

Giulio Casciola

File .vw

File contenente informazioni Camera:

1.189392 1.189392 0.551708	#camera position
-0.000164 -0.000164 0.311482	#object point
-0.099959 -0.099959 0.989958	#view-up vector
35.000000	#view-volume
1.000000	#aspect ratio

File .arg

File contenente informazioni di ray-tracing:

```
12                      #number of following arguments
-L300                   #image resolution
-e0.004000
-w
-c
-D2
-B1.000000
-AF
-t0.300000
-S chess_tower.sta      #statistic file
-o prova.hr             #image file output in .hr format
```

Esecuzione del ray-tracer

Comando di esecuzione da shell (posizionarsi nella directory bin di XCModel in cui è presente l'eseguibile hrayt):

./hrayt -k ../models/chess_tower -a chess.arg -m chess.md -V chess.vw

Script C per piccole animazioni

- save_scene_ani (int nframe,
  char *ani_name, char *name,
  char *title, ITEM it)

Esempio:

```
for (i=1; i<=10; i++)
{
        definizione i-esimo frame 3D
        save_scene_ani(10,”scacchi.ani”,”scacchi”,
                ”scena_scacchi”, scena);
}
```

Osservazioni per Animazioni

- Luci: creata una luce (create_light), la si può modificare (set_light), ma non eliminare; le luci “point_light” e “warn_light” hanno un parametro “max_range” che limita la sua influenza; se tale valore è 0 avrà influenza all’infinito;
- Trasformazioni geometriche: si possono applicare in ogni istante e provocano una matrice di trasaformazione composta che verrà applicata.
- Texture: ripetute set_texture sullo stesso oggetto implicano che sarà considerata solo l’ultima;
- Gerarchia: insert_in_list ripetute di una stessa istanza (lo stesso item) generano una gerrachia con più nodi, ma nulli tranne l’ultimo; bisogna generare nuove istanze prima di inserirle nella gerarchia.