University of Bologna - Department of Mathematics

Piazza di Porta S.Donato, 5 - 40127 - Bologna

xcrayt: the scene descriptor User's Guide - Version 1.0

G. Casciola

Department of Mathematics
University of Bologna

Bologna 2000

Abstract

This report describes the *xcrayt* and *xmovie* systems. They are two separate programs executable independently, but are to be considered part of a single working instrument to phisically describe a geometrically modelled scene in order to represent it realistically.

G. Casciola
Department of Mathematics, University of Bologna, P.zza di Porta S.Donato 5, Bologna, Italy. E-mail: `casciola@dm.unibo.it`.

Contents

CHAPTER 1

What is *xcrayt*?

xcrayt is the scene desriptor of the *xcmodel* system [XCMODEL00]. This program is self-contained and executable from the *xcmodel* console window. It is distributed with the archive **xcraytdev.tar.gz** (version in its development phase) and **xcraytusr.tar.gz** (executable version). Downloading and installation instructions are in [XCMODEL00].
xcrayt is a system allowing a virtual scene to be described interactively, in order to represent it realistically using the hrayt program. *xcrayt* could be defined as the make-up room and film director's office of a virtual film set. The actors present on the set are the individual surfaces or objects (groups of surfaces) that have been modelled and/or put together using *xcsurf* (see [XCSURF00]) and *xcbool* (see [XCBOOL00]). These surfaces are geometrically described by NURBS surfaces, trimmed in parametric form.
The costumes and make-up of the actors are defined by attributes of color and the visual properties of the materials and/or by the application of texture.
The lighting for the set (or scenery) is divided into "ambience" lighting, background colour and properties and the different types of puntiform light sources.
The position of the actors on the set and the filming is respectively made by the application of geometric transformations and the setting of the parameters of the camera.

Obviously, the system would be difficult to use for anyone not familiar with 3D transformations and the parameters regulating them. Similarly, some notions of optical physics, and of the lighting models used to simulate realistic effects, are essential. For an introduction to these subjects see [FOVD90], [GLAS89] and [CASC98].

However, the system is quite simple, so that even a user with little specific knowledge can learn to use it by experimentation.

xcrayt is supported by a program, hrayt, that implements a ray tracing algorithm optimized for trimmed NURBS surfaces. This algorithm is based on an algorithm of NURBS ray/surface intersection. The present version of

hrayt implements and gives the user the possibility to choose between three different ray/surface intersection algorithms.
In order to run the program, no specific knowledge of these algorithms, or of how a ray tracer works, is required. Obviously, a knowledge of how they function, together with the possibility of setting many parameters, allows the user to improve the performance in the rendering phase, sometimes even by a few minutes. For an introduction to ray tracing see [GLAS89] and [CASC98].

In this first version of *xcrayt* it is not yet possible to describe a scene interactively from nothing. This is because some of the functions have not yet been completely implemented. You are therefore advised to make a first sketch of the scene using the descriptor library (see [DESCR99]), in which all of the objects in the scene should be included (in fact, it is not possible to add them interactively at the moment). Save this scene sketch as an .md file (see section 4.), then open it with *xcrayt* and continue to describe it interactively.

1.1 How to work with *xcrayt*

When *xcrayt* is set up, it opens the window shown in fig.1.1. Note that the GUI of *xcrayt* is made with *xtools* [XTOOLS99]. The work area is the large window (its size is 1000×700) subdivided in six sectors that we call:

- **Control** top left, that controls camera and lighting parameters;
- **Object list** under the previous, from which it is possible to select scene objects;
- **Buttons** two rows of buttons at the bottom, which control a lot of operations that we will see in detail;
- **View** the large square area on the right, where the scene is visualised in wire-frame;
- **Selector View** to select some usual view points;
- **Transformation** to control and management some object geometric transformations.

As soon as this window appears, you will immediately see which actions are possible. The buttons, cursors, textboxes, bars, etc. are accompanied by one or more labels; for convention, if a label is drawn in black, the related function is 'able', whereas white means 'unable'. More labels near an xtools object mean multifunctionality. Look now at the able buttons only.

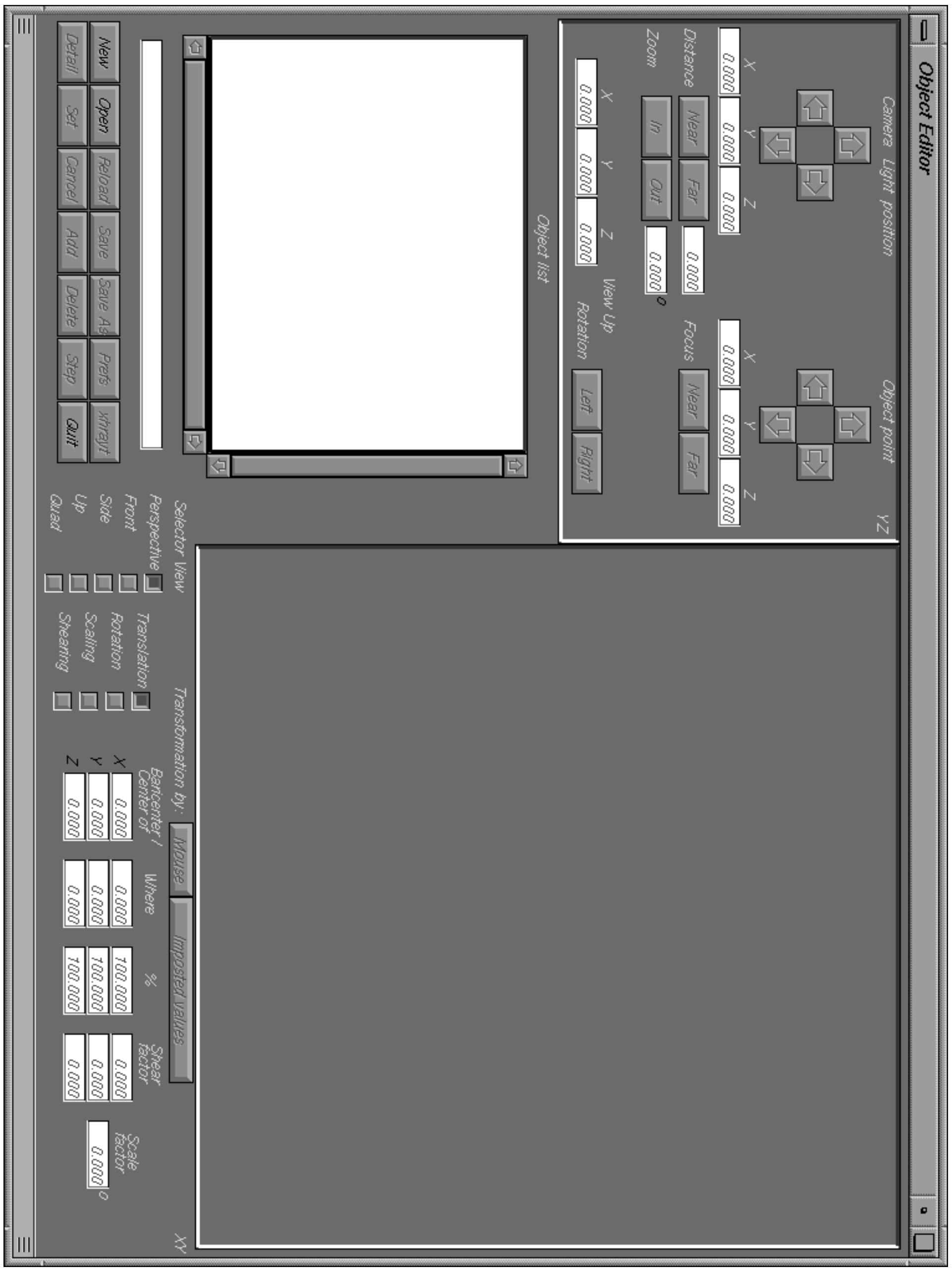


Figure 1.1: *xcrayt* panel, before to load or create a scene

1.1.1 Quit button

This button allows the user to quit *xcrayt*.

1.1.2 Open button

This button opens a file request window to load an (.md) file, which describes a scene or project. From its directory site and name, *xcrayt* deduces where to look for the scene components:

surfaces or objects are simple or trimmed NURBS surfaces or lists of simple or trimmed NURBS surfaces. These are respectively stored in files with .db, .dbe and .obj extensions. The file names are included in the scene description file (.md) (see Data file formats section);

camera data consists of camera position, scene point, view-up, view volume included in a .vw data file (it is looked for in the same directory of the project);

rendering parameters consists of the parameters to define the ray tracer execution included in a data file .arg (it is found in the same project directory);

texture useful for texture mapping; consists of image files with .hr or .ppm extensions.

The only essential file is the one that describes the scene (.md), while the others (.arg e .vw) are optional and serve to provide a particular rendering of the set.

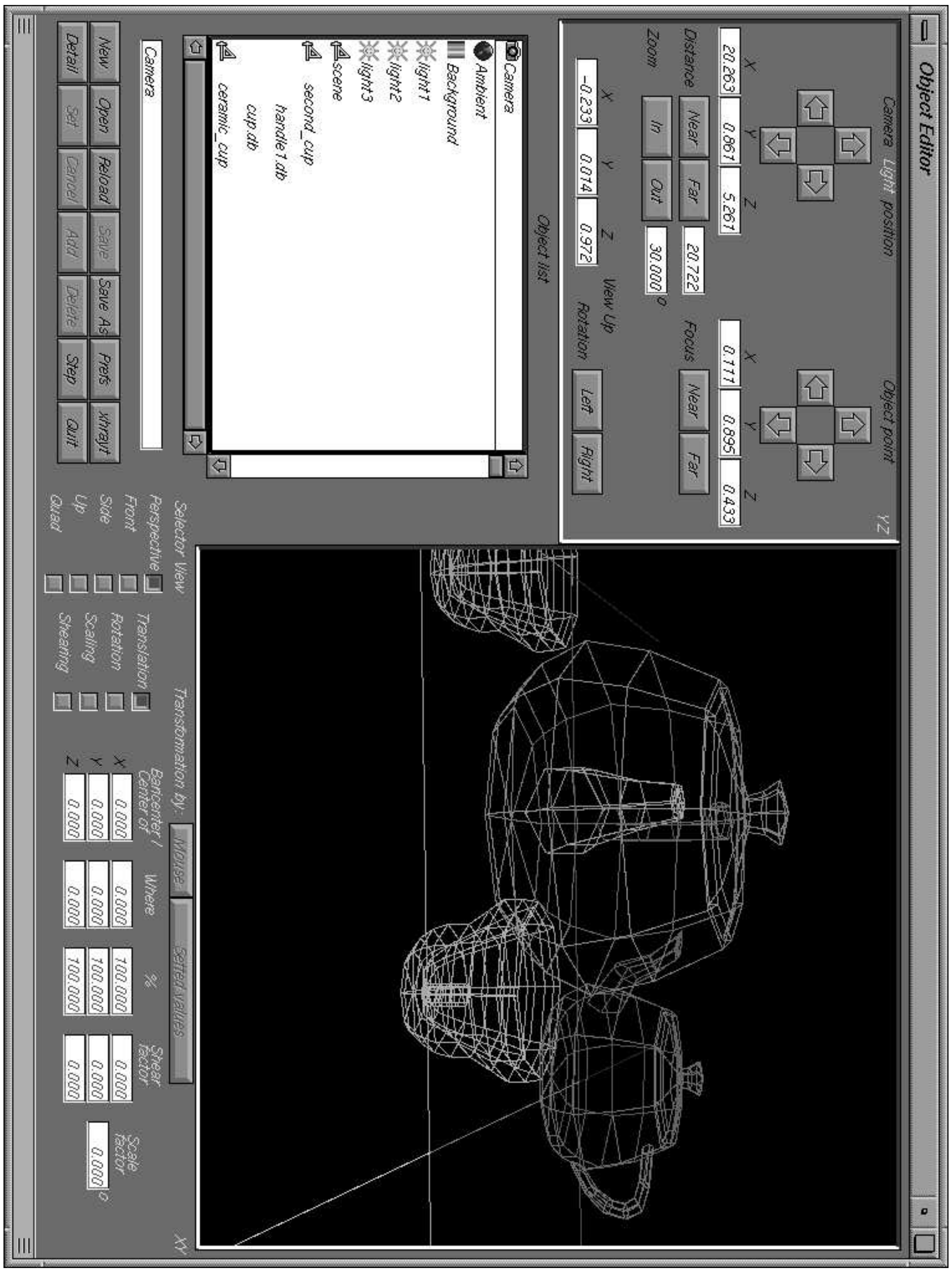
1.1.3 New button

This button functions in the same way as the **Open** button, with the difference that it requires the name of the directory to be used to contain the project. The camera, ambient e background items with default values immediately appear in the box object list (Not yet implemented).

As soon as an .md file has been loaded, or if you have started to create a scene, this is shown in the View sector under the default parameters of camera and the object list is abled. It allows you to select any item in the scene (see fig. 1.2). The hierarchical structure of the objects is shown by an icon near to every item and, indented on the left, you will see its sub-items. To select an item, click on it with the left mouse button. All and only the functions regarding that item will then become active. Its relevant textbox, as well as the bars, adds functionality enabling the user to select all the objects. The associated textbox serves to select an object by name.
The following paragraphs describe all the functions provided according to object type in the object list.

1.2 Camera

When selecting Camera in the Object list (this is the default selection that appears with the Object list sector), all the functions relating to it are abled in the *xcrayt* window (see fig 1.2). More precisely, the Control sector and



Camera Light position: X 20.263 Y 0.861 Z 5.261
Distance 20.722
Zoom 30.000 o
Object point: X 0.111 Y 0.895 Z 0.433
View Up: X −0.233 Y 0.014 Z 0.972
Camera

	Baricenter / Center of	Where	%	Shear factor
X	0.000	0.000	100.000	0.000
Y	0.000	0.000	100.000	0.000
Z	0.000	0.000	100.000	0.000

Scale factor 0.000 o

Figure 1.2: *xcrayt* panel, after a scene has been loaded

some of the buttons in the Buttons sector.

1.2.1 Control sector

The viewing direction is determined by the position of the camera and by a scene point. As can be understood from the label, the left four arrows (cursor) serves to position the camera, while the right serves to position the scene point. These two parameters are reported numerically in the textboxes below, from which they may also be varied. The X, Y and Z textboxes on

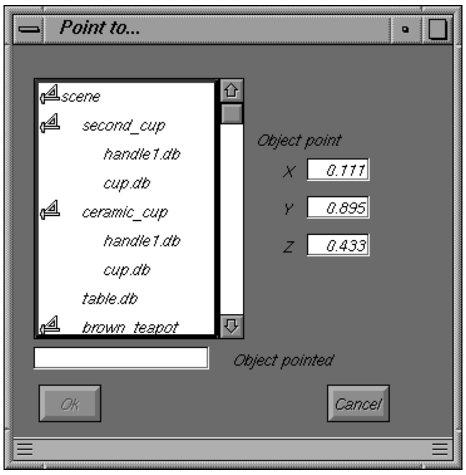


Figure 1.3: Select a scene point (object point)

the right have a further function in helping the user to choose a scene point. When they are pressed, a window appears (see fig. 1.3) allowing the center of gravity of an object in the scene (surfaces or objects consisting of surfaces) to be chosen as a scene point, or allowing the center of gravity co-ordinates to be inserted. The buttons present in the Control sector serve to vary the values of a fixed quantity (position, scene point and view volume).
Immediately below the cursors, next to the Distance label, there are two buttons and a textbox. The latter provides the distance between camera and scene point, while Near/Far moves the camera nearer to or further from the scene point. In order to position the camera at a fixed distance, use the textbox.
The **Zoom** button and the angle textbox allow the user to vary the view volume. Also in this case, the buttons vary the angle by a certain step (see **Step** button), while a precise value is inserted using angle textbox. If you click on this textbox, a list of the most common objectives appears (see fig. 1.4) that are contained in the file `camera.xclib` which you will find in the `xcmodel/lib` directory.
The data in the bottom left of the window, concerns the view up. The user

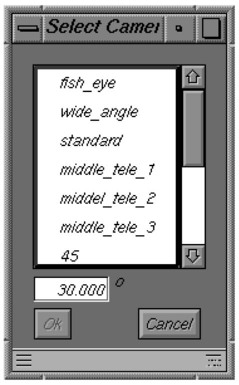


Figure 1.4: Select camera objective

has not been allowed a general setting, because this would be of little interest in the rendering. Sometimes this is used to make the viewed object seem rotated, while, in fact, it is the camera which is rotated. This parameter is often (0,0,1), indicating a perfectly upright camera with respect to the vertical axis. The rotation buttons, next to the above, that only vary the rotation of the frame (not the angle) are of greater utility.
Focus (Near/Far) has the same properties as Distance, except that it acts on the scene point by moving it away from or towards the camera.

1.2.2 Buttons sector

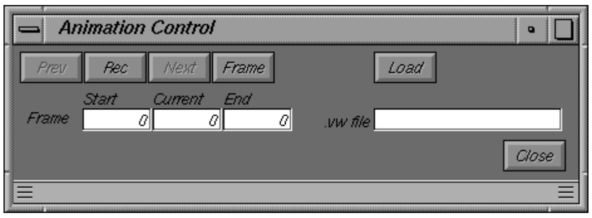
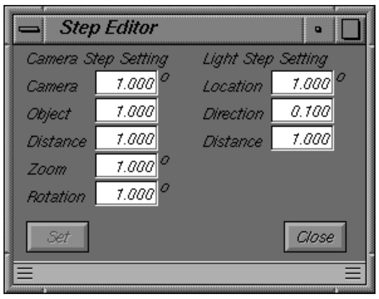


Figure 1.5: Load a set of .vw files

Detail button
The window of fig. 1.5 appears, allowing the user to set the parameters of the camera by loading a .vw file. If this file is loaded by a directory that is different from that of the present project, a copy is made.
This window also allows the user to load a list of different .vw files in order to make an animation.



Camera Step Setting		Light Step Setting	
Camera	1.000 o	Location	1.000 o
Object	1.000	Direction	0.100
Distance	1.000	Distance	1.000
Zoom	1.000 o		
Rotation	1.000 o		

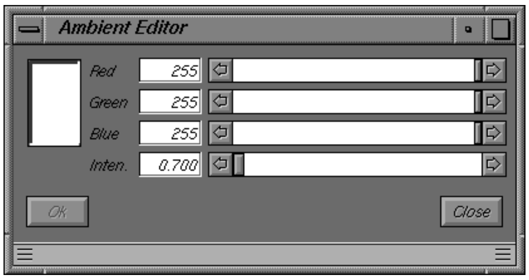
Figure 1.6: Step editor

Step button
This button opens a window (see fig. 1.6) enabling the user to vary the step to modify the camera position, the scene point and the lights. The values viewed are in degrees. In order to make the changes permanent, press Set. The default values will always be reset every time that the *xcrayt* program is run.

1.3 Ambient

If you select Ambient from the Object list, some buttons in the Buttons sector will be abled.

1.3.1 Buttons sector



Red	255
Green	255
Blue	255
Inten.	0.700

Figure 1.7: Ambient editor

Detail button
A window appears (see fig. 1.7) showing the present settings of the ambient light and enabling the colour to be changed; this variation is interactive.

1.4 Background

If you select Background in the Object list, some buttons in the Buttons sector will be abled.

1.4.1 Buttons sector

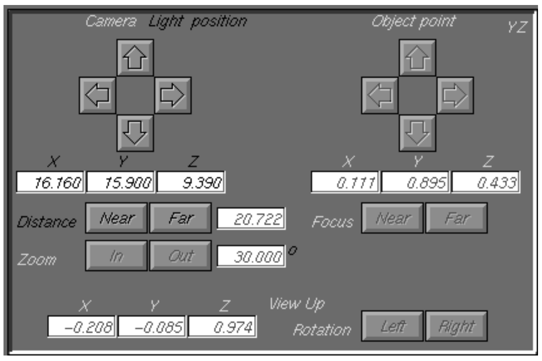
Detail button
A window appears similar to the one in fig. 1.7 showing the present settings of Background and enabling the colour to be changed and the background to be reflecting or not.

1.5 Lights

If you select a light source in the Object list some functions in the Control sector, and some buttons in the Buttons sector will be abled and the scene will be viewed in the View sector according to the parameters set on the light source.

1.5.1 Control sector

As can be seen on the label (see fig. 1.8), the left cursor is abled and determines the position of the light source, while the right cursor is unabled. The

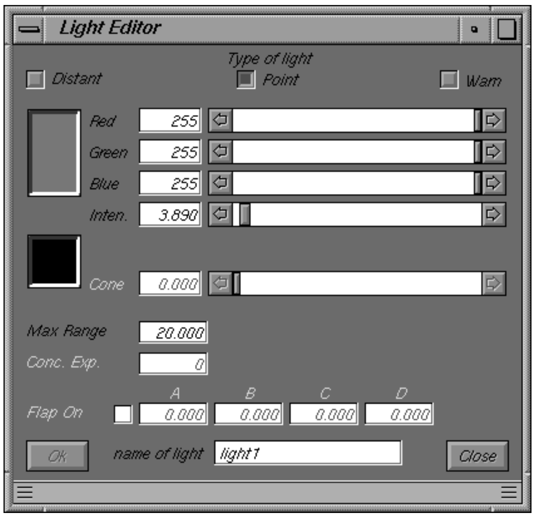


	X	Y	Z
Light position	16.160	15.900	9.390
Object point	0.111	0.895	0.433
View Up	-0.208	-0.085	0.974

Distance 20.722
Zoom 30.000 °

Figure 1.8: Control sector

position is numerically reported in the textbox below, which can be altered. Immediately beneath the active cursor, Distance is active, which moves the light source nearer to or further from the scene point using Near/Far.



Red 255
Green 255
Blue 255
Inten. 3.890
Cone 0.000
Max Range 20.000
Conc. Exp. 0
Flap On A 0.000 B 0.000 C 0.000 D 0.000
name of light light1

Figure 1.9: Light editor

1.5.2 Buttons sector

Detail button
A window appears, shown in fig. 1.9, which contains a lot of information about the lights: **Type of light**, color (**Red, Green** and **Blue**), **Intensity**, etc. The type of light selected determines which elements of the GUI will be active.

- A **Distant** light is a light source at an infinite distance from the scene on the direction defined by the (X, Y, Z) vector.
- A **Point** light is a light source in (X, Y, Z) with the intensity fading linerly to zero at the distance defined by **Max Range**; if the **Max Range** value is zero, the light intensity will not be dependent on the distant travelled.
- A **Warn** light is a **Point** light with a direction in order to aim a light at a particular area of an object. **Conc. Exp** defines the way in which light intensity decreases away from the aim direction; a larger value means that the light decreases quicker than for smaller value. **Cone** is an angular value used to restrict the path of the light; it define a cone surrounding the light direction. **Flap On** defines a plane equation $Ax + BY + CZ + D = 0$; the half space containing the light respect to the plane will be illuminated, while the other will not.

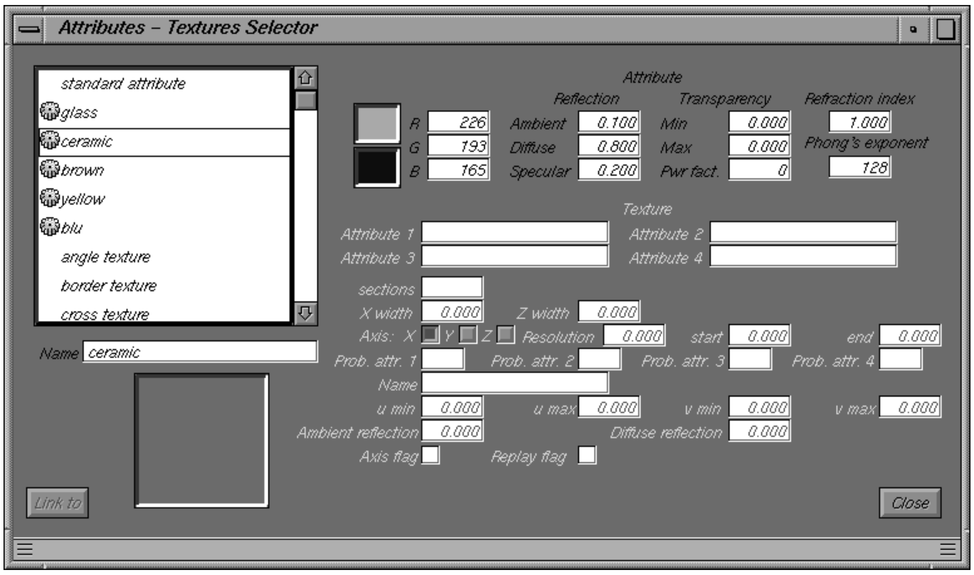
Step button
A window appears (see fig. 1.6) allowing the step changes of the camera

position, scene point and lighting to be modified. The values viewed are in degrees. In order to make the changes permanent, press **Set**. The default values will always be reset every time that the *xcrayt* program is run.

1.6 Objects

If an object item or one of its subitems in the Object list has been selected, some buttons in the Buttons, Selector View and Transformation sectors will be abled, and the scene will be viewed in the View sector from the point of view of the camera, showing the selected object (which will be drawn in white).

1.6.1 Buttons sector


Attributes – Textures Selector

standard attribute
glass
ceramic
brown
yellow
blu
angle texture
border texture
cross texture

Name: ceramic

Attribute

			Reflection		Transparency	Refraction index
R	226	Ambient	0.100	Min	0.000	1.000
G	193	Diffuse	0.800	Max	0.000	Phong's exponent
B	165	Specular	0.200	Pwr fact.	0	128

Texture

Attribute 1: | Attribute 2: | Attribute 3: | Attribute 4:
sections:
X width: 0.000 | Z width: 0.000
Axis: X [✓] Y Z | Resolution: 0.000 | start: 0.000 | end: 0.000
Prob. attr. 1: | Prob. attr. 2: | Prob. attr. 3: | Prob. attr. 4:
Name:
u min: 0.000 | u max: 0.000 | v min: 0.000 | v max: 0.000
Ambient reflection: 0.000 | Diffuse reflection: 0.000
Axis flag ☐ | Replay flag ☐

Figure 1.10: Link attribute to object

Detail button
This function allows you to match the selected object with a material. The materials to be used must already have been defined (see Attributes in the next section). Fig. 1.10 shows how to carry out this operation. The object will always be defined in a particular material, which, if unspecified, will be the default. The (unmodifiable) data concerning this material is provided in two sections according to type: at the top, if it is an attribute, at the bottom, if it is a texture. If the selected object is not a leaf (a surface) in the hierarchy, it is not possible to identify the material from which it has

been made (this is only possible by checking the leaves or the surfaces).
If you choose a texture, only some of the GUI fields will be abled (according to type). With texture mapping it is possible to choose and load an image from the texture directory through a file request; the image will be copied into the project directory and viewed in the square in the bottom left of the screen (see fig.1.10)(not yet implemented).

1.6.2 Selector View sector

This sector consists of 5 checkpoints (see fig.1.2); their selection allows a special scene to be viewed, or rather, a selected object in the scene to be viewed. The 5 views are:

- Perspective
- Front
- Side
- Up
- Quad

The Perspective view is the default. In each of these, the selected object will appear drawn in white, to be shown in the scene. The Quad view consists in the subdivision of the View sector into four parts showing the previous views.

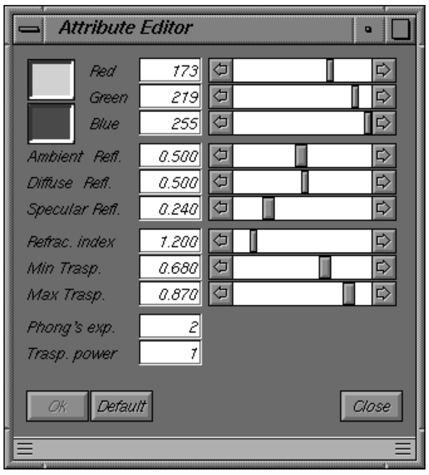
1.6.3 Transformation sector

This sector allows the selected object to be modified in terms of position (translation), scaling, rotation and share. For each modification, the user has to provide the fixed point for the transformation. Information about the center of gravity of the selected object, which will be assumed to be as before, unless specified, will help the user to do this. According to the chosen transformation (through the checkpoints) the user will be asked to insert the following parameters: the degree of rotation, the new position, the scaling factor, the shear factors, etc. For the scaling it is possible to obtain a modification by varying the percentage of the different dimensions. Once the data have been set, execute by pressing the button **Setted values**; interactive transformations are possible using the mouse (not yet implemented).

1.7 Attributes

If you select an attribute in the Object list, some buttons in the Buttons sector will be abled.

1.7.1 Buttons sector



Red 173
Green 219
Blue 255
Ambient Refl. 0.500
Diffuse Refl. 0.500
Specular Refl. 0.240
Refrac. index 1.200
Min Trasp. 0.680
Max Trasp. 0.870
Phong's exp. 2
Trasp. power 1

Figure 1.11: Attribute editor

Detail button
The window shown in fig. 1.11 appears, enabling the parameters for the selected attribute to be defined. These are the materials that can be associated with the objects in the scene; the refraction indexes are very important for their definition. By pressing the textbox relating to a Refraction Index the user can able a selection window containing a list of the most common materials. However, it is always possible to insert a generic material. Also in this case a library file has been created (refractionindex.xclib nella directory xcmodel/lib) in ASCII format, which can be edited by the user. For more information about the parameters defining an attribute see [DESCR99].

1.8 Other buttons in the Buttons sector

1.8.1 Set and Cancel buttons

The pair of buttons **Set** e **Cancel** allow a minimum level of undo/redo to be made on every object. These are immediately enabled, as soon as an object has been modified. The first shows the wish to make the changes permanent in the scene (in the memory but not on the file), while the second replaces the previous values in the object. If **Set** is chosen, both this button and **Cancel** will be unabled and the buttons **Save** and **Save As** will appear.

1.8.2 Reload, Save and Save As buttons

The **Reload** button allows the whole project to be reloaded without having to reopen file request again to repeat the selection. This is useful in cases in which the scene has been recompiled using the descriptor library, or you wish to start again from the last changes saved.
The **Save As** button is used to save the project on files; the model is saved in a file with an .md extension and the viewing parameters in a file with an .vw extension. It is possible to save one or both (see fig.1.12), and it is possible to choose an appropriate filename or directory. The **Save** button

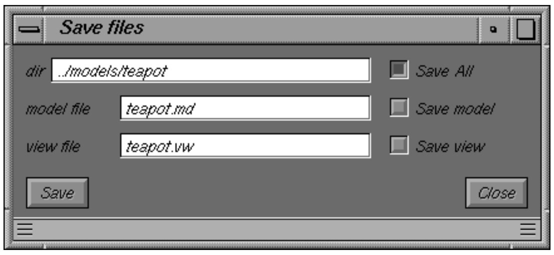

Save files
dir ../models/teapot
model file teapot.md
view file teapot.vw
Save All
Save model
Save view
Save
Close

Figure 1.12: Save As files

saves automatically both .md and .vw file with the chosen name and project directory.

1.8.3 Delete button

This option deletes an element from the scene, the one selected in the object list. It is not possible to delete Camera, Ambient or Background. If these are superfluous for a particular project, can be unabled by setting the intensity to zero, or by setting them as though they were black. As far as the other elements are concerned, the selected object will be deleted. In the case of the attributes it is not possible to delete an attribute of an object that is still being used. First you need to delete the object (or objects) that refer to it, or change its reference points. As far as hierarchy is concerned, this is a more difficult problem, since the data structure is tree-like. The user must be particularly careful when deleting the final "leaf" of a "node". In this case, the "node" will also have to be deleted to produce a new tree (not yet implemented).
The Delete operation is permanent, there is no Undo.

1.8.4 Add button

This option adds lights, attributes or objects. No other type of insertion is possible. For the first two, once a name has been chosen, the windows already viewed in fig. 1.9 and fig. 1.11 will be opened with their default values (standard attribute and white Warn light). This new attribute or light will appear at the bottom of the attributes and lights list. The objects behave differently. In fact, they have several types of insertion (not yet implemented):

surface add a simple surface (file .db or dbe) in the tree ;

object add an object (list of surfaces included in a file .obj);

copy of a leaf add a simple surface already presented in the hierarchy as a leaf;

copy of a subtree add an object already presented in the hierarchy as a subtree;

1.8.5 xhrayt button

This button executes the rendering engine. If it comes across a change in the project that has not yet been saved, the user will be asked to save the project; xhrayt will only run when the user has saved all the modifications. The window shown in fig.1.13 (front-end image of hrayt) is opened by xhrayt

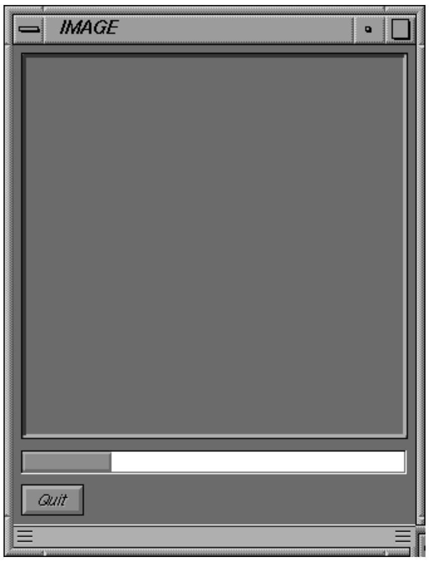


Figure 1.13: xhrayt execution

program, which then runs hrayt, the real ray-tracer of *xcmodel*; the bar shows the progress of the rendering. The system allows several types of rendering to be executed.

1.8.6 Prefs button

This button allows a series of parameters defining the ways in which the ray tracer must render the model to be varied interactively throught the window of fig. 1.14.

The parameters contained when the window is opened read by a file with

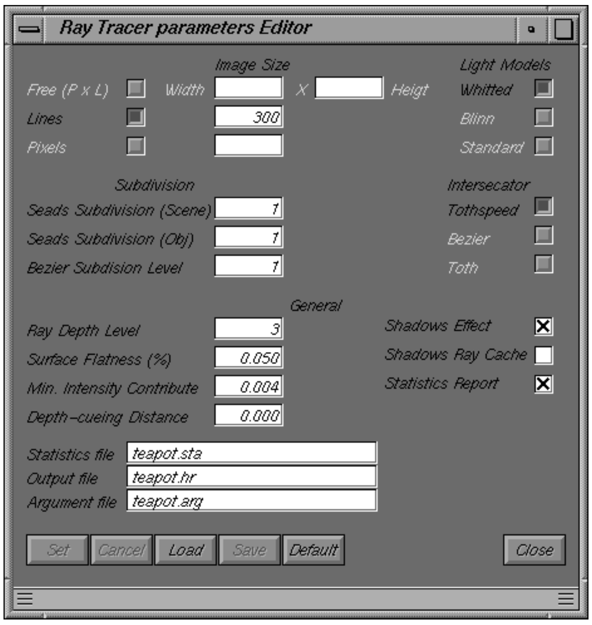


Image Size
- Free (P x L) ☐ Width ___ X ___ Heigt
- Lines ■ 300
- Pixels ☐

Light Models
- Whitted ■
- Blinn ☐
- Standard ☐

Subdivision
- Seads Subdivision (Scene) 1
- Seads Subdivision (Obj) 1
- Bezier Subdision Level 1

Intersecator
- Tothspeed ■
- Bezier ☐
- Toth ☐

General
- Ray Depth Level 3
- Surface Flatness (%) 0.050
- Min. Intensity Contribute 0.004
- Depth–cueing Distance 0.000
- Shadows Effect ☒
- Shadows Ray Cache ☐
- Statistics Report ☒

- Statistics file teapot.sta
- Output file teapot.hr
- Argument file teapot.arg

Figure 1.14: Render parameters

the same name as the project and with an .arg extension. If these files do not exist, the default parameters will be used. It is possible to load an .arg file with a different name from that of the project by setting the name and using the **Load** button. The parameters are:

- **Image Size** to set the dimensions of the image to be produced. The information consists in the number of lines and columns of the discretised grid of the screen in pixels or in the number of lines or pixels for a square grid.
- **Seads Subdivision (Scene)** and **Seads Subdivision (Obj)** to determine the number of divisions along each axis, in order to subdivide the scene space and the NURBS objects. The three-dimensional grids are represented by a SEADS data structure, which stands for Spatially Enumerated Auxiliary Data Structure.
- **Bezier Subdivision Level** to determine the subdivision level of the parametric space of each surface, as well as the basic one in Bezier

patches. Each of these patches is then evenly subdivided into 2^k patches, where k is the value of the object field.

- **Ray Depth Level** sets the maximum depth of the ray tree.
- **Surface Flatness** defines the tolerance with which a surface is considered flat in the Bezier subdivision process..
- **Min. Intensity Contribute** defines the value below which the contribution coming from a ray on the ray tree should not be considered.
- **Depth-cueing Distance** sets the maximum distance a surface point is viewed and then rendered.
- **Light Models** determines the lighting model on which to base the rendering of the scene.
- **Intersector** determines the type of algorithm intersector used by the ray tracer.
- **Shadows Effect** allows shadows to be enabled.
- **Shadows Ray Cache** allows caching of shadows to be improved.
- **Statistics Report** enables the production of a file containing statistical data about the execution of the ray tracer.
- **Statistics file** and **Output file** specify the names of the related files.

The buttons **Set** and **Cancel** respectively enable the new parameters or return to the previous ones. The **Save** button saves the parameters set in the specified .arg file. The **Default** button sets the default parameters.

CHAPTER 2

What is *xmovie*?

xmovie is the viewer of the *xcmodel* system [XCMODEL00]. It is self-contained and executable from the *xcmodel* console window. It is distributed with the archive **xcraytdev.tar.gz** (version in its development phase) and **xcraytusr.tar.gz** (executable version). Downloading and installation instructions are in [XCMODEL00].
xmovie is a viewer for images produced by *xcrayt* (or rather, by hrayt). It is therefore useful to view images that have previously been produced and saved, but, above all, to view animations made using *xcrayt*, which have been saved, photogram by photogram, but not viewed.
xmovie also allows the image format of *xcmodel*, that is .hr to be converted into a standard format, such as ppm.

It can be executed from the console window of *xcmodel*. *xmovie* appears with the window of fig. 2.1, which has the functions described below.

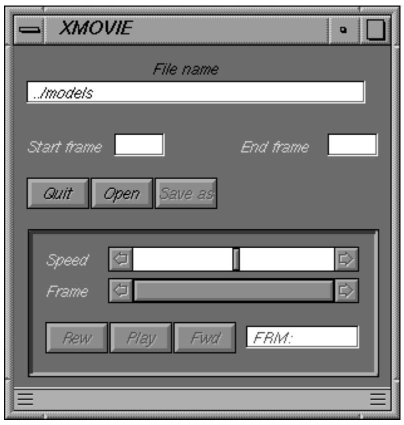


Figure 2.1: *xmovie* control panel

2.1 Quit button

This button allows the user to quit *xmovie*.

2.2 Open button

This button opens a file request to load a .hr or .ppm image file, or a file .hra containing a list of images. If an image file has been selected, it will be viewed as follows (see fig. 2.2). This image file can then be saved again using the **Save As** button with a different name and/or format (from .hr to .ppm or viceversa). If a list of images is selected, these will all be loaded in the memory and the sector of the window that manages the animations will be enabled. Using the **Rew**, **Play** and **Fwd** buttons it is possible, respectively, to rewind the film photogram by photogram, to view the whole animation at the speed chosen by the **Speed** bar, and to go through the animation photogram by photogram. **Rewind** and **Forward** can also be performed using the **Frame** bar. In this case the **Save As** button is unabled.

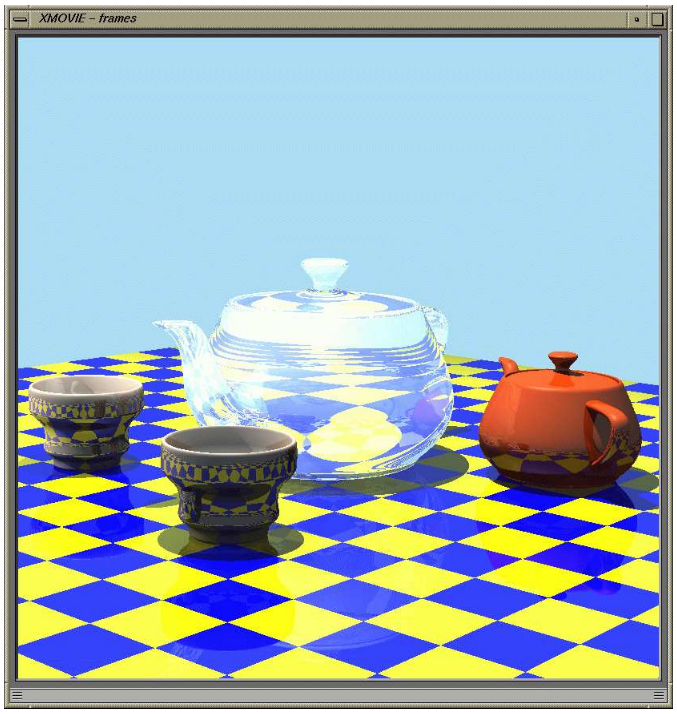


Figure 2.2: Rendered image

CHAPTER 3

Data file formats

In this section the syntax of each file format used by *xcrayt* is given and explained. The data files created or used by *xcrayt* are stored in xcmodel/models directory as default. The # character in the following is a comment to the data in the file.

3.1 Scene description

The following example file is xcmodel/models/gc_chess/gc_chess.md. gc_chess is the directory project, gc_chess.md is the scene description file. The extension .md identifies a scene model. The format is self-explanatory.

```
HEADER: torre 3 3 1 #scene name,N.lights,N.attrs+1,N.file surf
AMBIENT COLOR: 1.000000e+00 1.000000e+00 1.000000e+00 #ambient
AMBIENT INTENSITY: 7.000000e-01                          # ambient
BKG COLOR: 4.470000e-01 4.700000e-01 5.880000e-01  #background
BKG IS REFL: 0                                     #background
LIGHT N. 0                  # defines the first light of three
NAME: distant_light                                # light name
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                 # defines the second light of three
NAME: point_light                                  # light name
...
```

```
ATTRIBUTES N. 1               # defines the first attr. of two
NAME: scacco_bianco                            # attribute name
COLOR: 4.820000e-01 4.820000e-01 1.000000e+00     # RGB values
5.000000e-01                            # ambient reflection
5.000000e-01                            # diffuse reflection
4.000000e-01                             # mirror reflection
INDEX: 1.000000e+00 1.000000e+00          # refraction index
2                                         # Phong's exponent
0.000000e+00 0.000000e+00        # Min. and Max. transparency
0                                       # transparency power
ATTRIBUTES N. 2              # defines the second attr. of two
...
SURFACE N. 0               # defines the first surface. of one
NAME: Qtorre.db                                  # surface name
IS SURFACE: 1
NORMAL INSIDE: 0                       # surface normal versus
EXTENT:                                 # surface bounding box
0.000000e+00 0.000000e+00 0.000000e+00
2.000000e-01 2.000000e-01 6.400000e-01
HIERARCHY:                   # begin the hierarchy description
OBJECT TYPE: 1                                          # root
OBJECT NAME: torre                                 # root name
OBJECT TYPE: 3                                   # a left leaf
PRIMITIVE NAME: slice_4              # left leaf surface name
GEO SHAPE: 11          # surface geometric transform. to apply
TRANSFORM MATRIX TYPE: 1
TRANSFORM MATRIX:
-4.846628e-15 -1.000000e+00 0.000000e+00
1.000000e+00 -4.846628e-15 0.000000e+00
0.000000e+00 0.000000e+00 1.000000e+00
2.500000e-01 1.750000e+00 0.000000e+00
16
0
TEXTURE MATRIX TYPE: 1    #surf. material (texture and attr.)
TEXTURE MATRIX:
-1.864088e-14 -3.846154e+00 0.000000e+00
3.846154e+00 -1.864088e-14 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
marble_5_black.hr                                 # texture file
ATTRIBUTE N. 2                                # attribute applied
NURBS N. 0                            # index NURBS surface loaded
OBJECT TYPE: 2                           # a right son (node tree)
OBJECT TYPE: 3                                     # a left leaf
PRIMITIVE NAME: slice_3                 # left leaf surface name
...
OBJECT TYPE: 2                                       # right son
OBJECT TYPE: 3                                     # a left leaf
PRIMITIVE NAME: slice_2                  #left leaf surface name
...
OBJECT TYPE: 2                                       # right son
OBJECT TYPE: 3                                     # a left leaf
PRIMITIVE NAME: slice_1                 # left leaf surface name
...
OBJECT TYPE: 0                                    # a right leaf
```

3.2 View parameters

Contains the necessary information in order to define a 3D perspective view, that can simulate filming with a telecamera. These files have an .vw extension.

```
14.03 11.72 1.60                                  # camera point
0.20 0.80 0.45                 # scene point or object point
-0.00 -0.09 0.99                                       # view up
26.00                                              # view volume
1.00                                          # aspect ratio x/y
```

3.3 Ray tracer parameters

These parameters define the arguments of the ray tracer. These files have an .arg extension.

```
11                # number of arguments, follows the arg list
-L200
-e0.004000
-w
-c
-v1
-n1
-B1.000000
```

```
-S gc_chess.sta
-o gc_chess.hr
```

The arguments can be:

argument	signify
-B n	min flat ratio
-b	Blinn light model
-c	no shadow cache
-Dn	max 2^n subdivision in Bezier patches
-ddepth	max intersection tree depth
-ex	minimum intensity contribution
-Ln	image resolution
-ln	first line to be processed
-MB	Bezier Clipping intersector method
-MT	Toth intersector method
-nn	nurbs extent subdivisions
-o filename	image filename
-Pn	number of width pixels in the image; the height is deduced by the aspect ratio
-pn	first pixel to be processed
-rreach	computed intensity is proportional to the distance between the camera and the point considered. This is linearly scaled by a factor one at zero distance to a factor zero at the reach distance
-S filename	statistical file name
-s	shadow ON
-vn	scene extent subdivisions
-w	Whitted light model

3.4 Image file

This is a compressed, non-standard format for images, which can also be used for textures. These files have an .hr extension.

3.5 Standard image file

Standard image format common in Xwindow environments and known as ppm. This is a binary RGB file for true-color images.

3.6 Animation file

This contains the list, in ASCII format, of the frames (.hr or .ppm files) making up the animation.

```
10                                                # frame number
frame1.hr                           # follows the frames file names
frame2.hr
....
frame9.hr
frame10.hr
```

3.7 Statistics file

Statistics file for the image. This contains information about image generation. It is generated from *xcrayt* only if requested and, if it already exists, the information relating to the image in the rendering phase will be appended to the file. As an example, here is the Statistics file of the rendering of a scene.

```
GENERAL STATISTICS
------------------
run-length-encoded image on ............ :
../models/gc_chess/gc_chess.hr
image resolution (lines x pixels) ...... : 200x200
maximum intersection-tree depth ........ : 1
adaptive intersection-tree depth control : ON
minimum intensity contribution ......... : 0.004
lighting model ......................... : Whitted
depth cueing range ..................... : 0.0 - INFINITY
shadows ................................ : OFF
shadows cache .......................... : OFF
number of light sources ................ : 3
number of primitive objects ............ : 4
number of textured primitives .......... : 4
total number of patches ................ : 31
max uniform subdivision depth .......... : 0
min patch flatness ..................... : 1.00%

number of rays traced ..................... : 38000
total number of ray-extent checks ......... : 1710148
number of ray-object intersections ........ : 11463 (30.17%)
number of shadow ray-object intersections . : 0 (NaN%)

number of ray-patch intersections calls .... : 79600
number of tests resolved with second extent  : 19786 (24.86%)

Seads statistics : Scene seads
--------------------------------------------------------------
```

```
object tested per ray ......................... : 1.83
box tested per ray ............................ : 4.00

object tested per normal ray .................. : 1.83
object tested per shadow ray .................. : NaN
box tested per normal ray ..................... : 4.00
box tested per shadow ray ..................... : NaN
percentage empty voxel ........................ : 0.00%
average objects per noempty voxel ............. : 4.00
--------------------------------------------------------------

Seads statistics : Second level
--------------------------------------------------------------
object tested per ray ......................... : 1.75
box tested per ray ............................ : 31.00

object tested per normal ray .................. : 1.75
object tested per shadow ray .................. : NaN
box tested per normal ray ..................... : 31.00
box tested per shadow ray ..................... : NaN
percentage empty voxel ........................ : 0.00%
average objects per noempty voxel ............. : 31.00
--------------------------------------------------------------


preprocessing time .................... :  0h   0'  0.00''
ray tracing processing time ........... :  0h   0'  12.00''
total processing time ................. :  0h   0'  12.00''
```

List of Figures

Bibliography

[XCMODEL00] G.Casciola, *xcmodel*: a system to model and render NURBS curves and surfaces; User's Guide - Version 1.0, Progetto MURST: "Analisi Numerica: Metodi e Software Matematico", Ferrara (2000), http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[XCSURF00] G.Casciola, *xcsurf*: the 3D modeller; User's Guide - Version1.0, Progetto MURST: "Analisi Numerica: Metodi e Software Matematico", Ferrara (2000), http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[XCBOOL00] G.Casciola, *xcbool*: the object composer; User's Guide - Version 1.0, Progetto MURST: "Analisi Numerica: Metodi e Software Matematico", Ferrara (2000), http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[XTOOLS99] S.Bonetti, G.Casciola, *xtools* library; Programming Guide - Version 1.0, (1999) http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[DESCR99] S.Bonetti, G.Casciola, *descriptor* library; Programming Guide, (1999) http://www.dm.unibo.it/∼casciola/html/xcmodel.html

[FOVD90] J.D.Foley, A.VanDam, S.K.Feiner, J.F.Hughes, Computer Graphics principles and practice, II Edition, Addison Wesley (1990).

[GLAS89] A.S.Glassner, An introduction to ray tracing, Academic Press (1989).

[CASC98] G.Casciola, Metodi Numerici per la Grafica, Dispense C.d.L. Informatica, Università degli Studi di Bologna, (1998).