Learning 3D Visualis a tion with MALUDA

João A. Madeiras Pereira and **Mário Rui Gomes**
Instituto Superior Técnico/Instituto Engenharia Sistemas e Computadores
Lisboa, Portugal.
{jap, Mrg}@inesc.pt

Abstract

This paper describes MALUDA system, which constitutes an important tool for teaching 3D-visualisation pipeline at the laboratory classes of the Computer Graphics course at Instituto Superior Técnico.
Keywords: *application tool, 3D visualisation pipeline, modelling, rasterization algorithms.*

1. Objectives

The traditional 3D-visualisation pipeline is one of the main topics to be taught in any introductory Computer Graphics course. The MALUDA system constitutes itself as an important complementary tool to learn about 3D visualisation. In fact, it is used at the laboratory classes of the Computer Graphics course at Instituto Superior Técnico, not only as a way to practice the use of a virtual camera model based-viewing pipeline but also as a test environment for rasterization algorithms developed by the students.

2. MALUDA main features

There are three main approaches a novice student in Computer Graphics can take to make effective use of MALUDA:

- Learning to create 3-dimensional scenes
- Understanding and observing different realism levels of image rendering
- Testing his own scan conversion strategy.
-

2.1. Three-dimensional modelling

A scene can be built up by using the NFF (Neutral File Format) language and/or the SIPP (Simple Polygon Processor) scene description library.

NFF [1] is a low level scene description language in the sense that it provides a minimal interface which allows a programmer to quickly create a scene by describing the geometry and basic surface characteristics of objects, the placement of lights, and the viewing frustum for the eye. There is no hierarchical modelling. An example of a NFF file is illustrated in Appendix A.1.
In addition to support the NFF scene description language, MALUDA also supports SIPP scene description library [2]. A SIPP scene is built up of objects (described in their local coordinates systems) which can be transformed with rotation, translation and scaling. The objects form hierarchies where each object can have arbitrarily many sub-objects. It allows also a scene to be illuminated by an arbitrary number of light sources of different types. In a SIPP scene is possible to create several virtual cameras, and then specify one of them to use when rendering the scene.

An important feature of MALUDA modelling capability is the possibility of merging a SIPP scene with one described in NFF language. Then, a user can ask either to render the resulting scene or to export it in NFF format (containing optionally only triangles).

2.2. Image Rendering

MALUDA provides the ability for creating 3-dimensional scenes and rendering them using either the internal scan-line z-buffer algorithm or an external rasterization algorithm provided by the user. By default, the internal rasterization engine is used in the rendering operation. This engine, through the use of the public domain SIPP library, renders images at different realism levels. Scene objects are rendered with either Phong, Gouraud or flat shading. An image can also be rendered as a line drawing of the polygon edges without any shading at all. The SIPP library also provides 3-dimensional texture mapping with automatic interpolation of texture coordinates. Simple anti-aliasing can be performed through oversampling. Transparencies are allowed and several shading functions (“shaders”) are available. Some examples generated by MALUDA by using the internal SIPP rasterization engine are illustrated in Appendix B.

2.3. Rasterization support

A major feature of this tool is the ability for a student to provide his own rasterization function. This makes it easy to experiment with various scan conversion schemes and to do special effects.

Two procedures can be used accordingly to the platform to execute the rasterization algorithm is relevant or not for the test. If execution platform is not important, the routine containing the rasterization code developed by the student can be integrated into MALUDA system to be invoked later. This procedure is meant to check the strategy and its performance whatever the machine and operating system. Or, the platform is important, like a parallel machine where we are interested to experiment several parallel schemes for the rasterization stage. In this case, MALUDA should be instructed to output an ASCII file containing device dependent coordinates polygonal database, which will be then used to feed the target rasterization algorithm. An example of this "flat" file is shown in Appendix A.2

3. MALUDA architecture

In figure 1, is depicted the MALUDA activity flow diagram

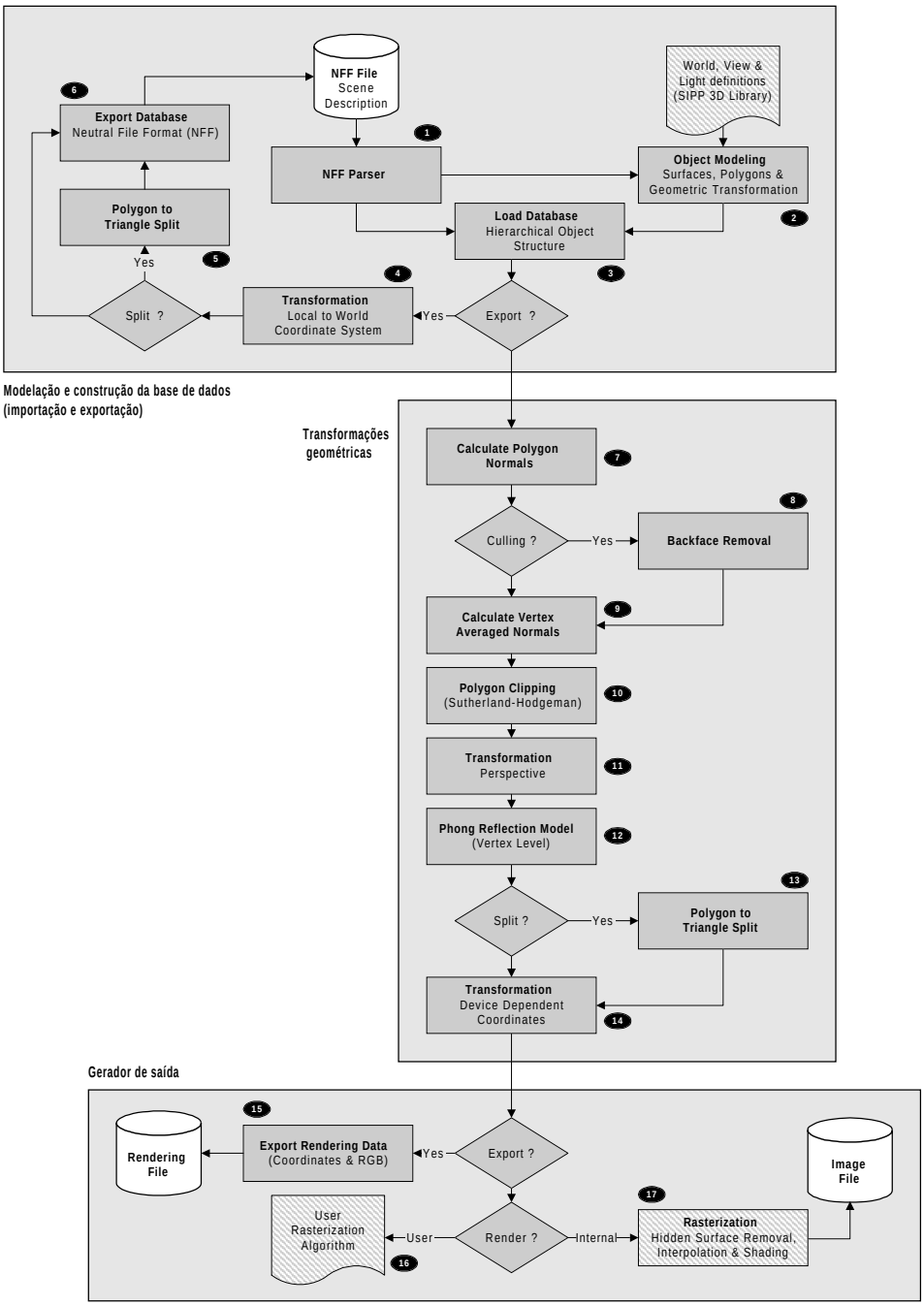


Figure 1 - MALUDA activity flow diagram

An analysis of it shows to a reader the most important modules and their associated functionality. Thus, accordingly to the MALUDA system description made above, we can distinguish the following functional blocks:

- Reader and parser of scenes created by NFF and/or SIPP languages
- Geometric transformer
- Output generator

At this point, is convenient to remind some important features of MALUDA and relate them with the numbered operations illustrated in Figure 1.

MALUDA can export the input scene as a NFF file format (World Coordinates) –operation 6– or as an ASCII polygonal "flat" file format (Device Dependent Coordinates) –operation 15. The system provides also the ability of image rendering by using either the internal SIPP rasterization engine –operation 17- or an external rasterization algorithm provided by the student –operation 16.

It is interesting to mention that when MALUDA is configured to act as image renderer, diagnostic and statistics informations are provided by the system. Appendix A.3 shows the information provided by the system when the default SIPP rasterization engine is being used.

References

[1] Eric Haines, "A Proposal for Standard Graphic Environments", in *IEEE Computer Graphics and Applications*, Vol 17, nº11, November 1987, pp 3-5

[2] J. Yngvesson, I. Wallin; "User's Guide to SIPP –a 3D rendering library", version 3.1, Public Domain Software via anonymous ftp from isy.liu.se (IP nº 130.236.1.3) in the directory /pub/sipp; March 1993

Appendix A – MALUDA Input/Output information

A.1 NFF File Format

```
v                Viewpoint definition;
from  1.02285  -3.17715  -2.1745
at  -0.004103  -0.004103  0.216539
up  -0.816497  -0.816497  0.816497
angle  45
hither  1
resolution 1024  1024
l  1.87607 -18.1239 -5.00042      light source position;
f  1 0.2 0.2 1 0 100000 0 0       surface parameters;
p  3          Polygon with three vertices;
-1  -1  1
-1  -0.9375  0.9375
-0.9375  -1  0.9375
p  3
-0.9375  -0.9375  1
-0.9375  -1  0.9375
-1  -0.9375  0.9375
p  4
12  12  -0.5
-12 12  -0.5
-12  -12  -0.5
12  -12  -0.5
f  1  0.9  0.7  0.5  0.5  45.2776  0  0          New surface;
s  0  0  0  0. 5          a sphere;
s 0.272166  0.272166  0.544331  0.166667
…
```

A.2 Device Dependent Coordinates Output File

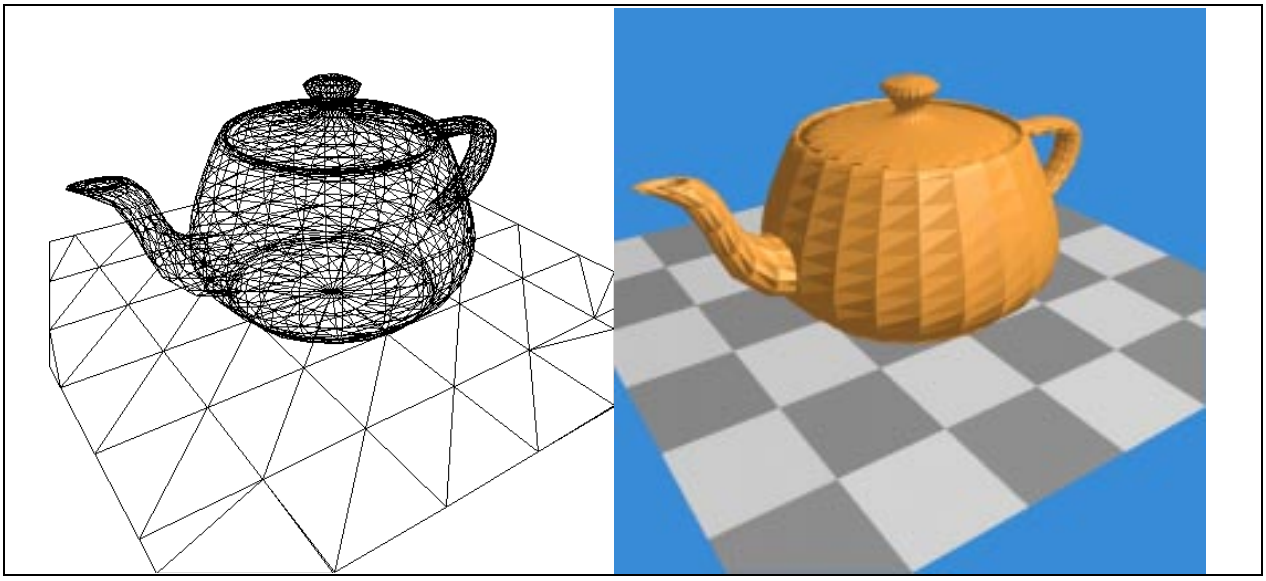
```
512 512 65535 255          Maximum resolution (x,y,z and colour);
120838                     Total number of polygons;
1                          Separator;
3 100 50                   Polygon with 3 vertices and Ymax/Ymin;
40 100 89 30 128 30        (x,y,z) and (RGB)
50 80 89 30 128 30
60 50 89 30 128 30
1
4 110 40                   Polygon with 4 vertices and Ymax/Ymin;
40 40 89 30 128 30
50 110 89 30 128 30
60 50 89 30 128 30
55 60 89 30 128 30
1
127                        End of file;
```

A.3 Diagnostic and Statistics information

```
NFF scene will be imported!
Pipeline: Parsing NFF file, wait...
Pipeline: Be aware that some SIPP and user defined view parameters
          will be override by NFF parameters!
Pipeline: NFF background loaded
Pipeline: NFF view parameters loaded
Pipeline: NFF light loaded (6.00 6.00 6.00) (1.00 1.00 1.00)
Pipeline: NFF surface with 1 polygons
(0.30,0.60,0.30,5,1.00,0.85,0.70)
Pipeline: NFF surface with 1 polygons
(0.30,0.00,0.20,5,1.00,0.61,0.41)
Pipeline: NFF surface with 145 polygons
(0.30,0.00,0.20,5,1.00,0.61,0.41)
Pipeline: NFF surface with 1 polygons
(0.30,0.00,1.00,5,1.00,1.00,1.00)
Pipeline: NFF surface with 145 polygons
(0.30,0.00,1.00,5,1.00,1.00,1.00)
Pipeline: NFF surface with 1 polygons
(0.30,0.00,1.00,5,0.02,0.67,1.00)
Pipeline: NFF surface with 145 polygons
(0.30,0.00,1.00,5,0.02,0.67,1.00)
Pipeline: NFF surface with 1 polygons
(0.30,0.00,1.00,5,0.21,1.00,0.01)
…
Pipeline: NFF surface with 145 polygons
(0.30,0.00,1.00,5,1.00,0.26,0.75)
Pipeline: Imported 129 surfaces with 9345 polygons, 0 spheres and 0
cones.
Pipeline: NFF parser done
Pipeline: Hither:1.00,Yon:15.00,
Resolution(X,Y,Z,Color):1024,1024,1024,512
Pipeline: Pipeline creating render data, wait...
Timer started           Geometric Transformations start;
Pipeline: Image file to be created: 'pipe.ppm'
Pipeline: Objects loaded: 3
Pipeline: Surfaces loaded: 129
Pipeline: Polygons loaded: 9345
Pipeline: Polygons ready to be rendered by SIPP!          Geometric
Transformations end;
Timer Stopped (Elapsed: 2.108s, User: 2.023s, System: 0.066s)
Pipeline: SIPP rendering, wait...
Timer started           Rasterization stage;
Pipeline: Render Mode: Gouraud Método de sombreamento;
Pipeline: SIPP rasterization done!
Timer Stopped (Elapsed: 112.693s, User: 108.058s, System: 1.053s)

Pipeline: Successful End
Pipeline: Polygons backfacing: 4864
Pipeline: Polygons totally clipped: 39
Pipeline: Polygons splited to triangles: 0
Pipeline: Polygons rendered: 4442
Pipeline: Releasing resources...!
Pipeline: Release complete
```

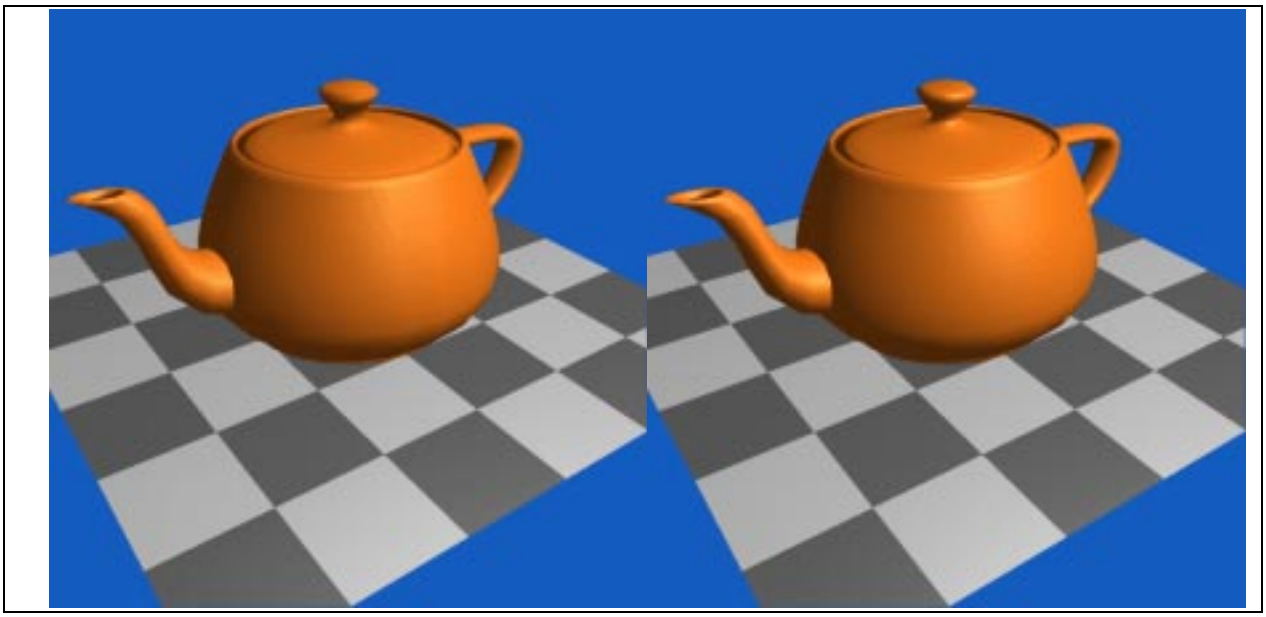
Appendix B – Different types of image rendering with SIPP rasterization



Teapot - *Wire-Frame* model

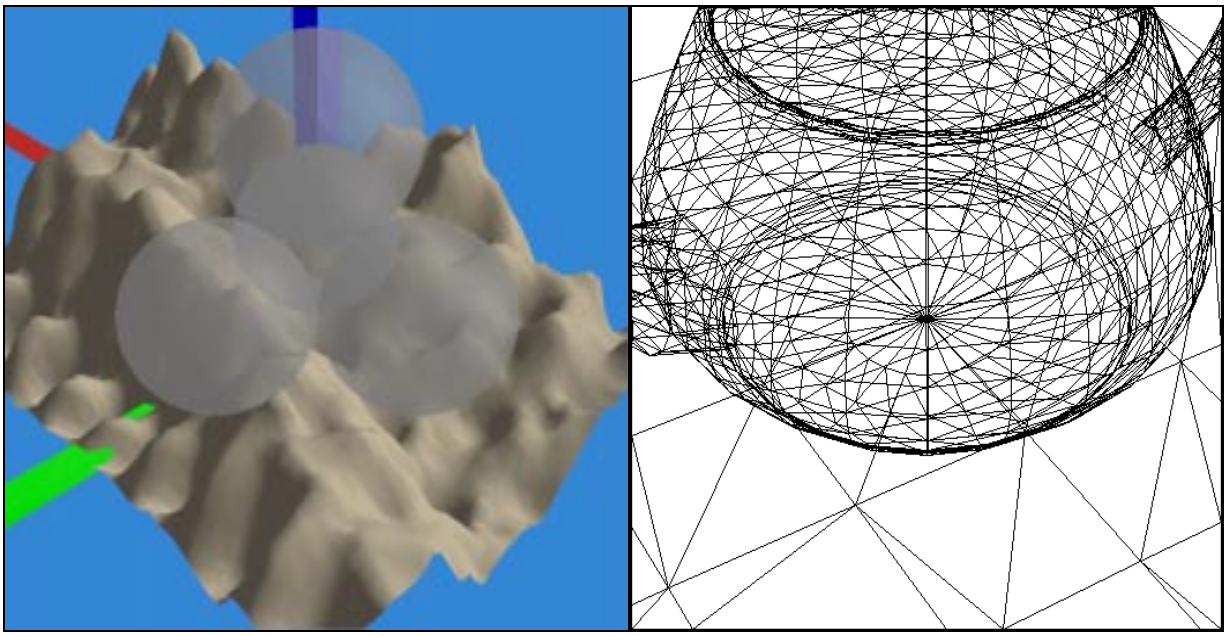
Teapot – Flat shading

Appendix B – Different types of image rendering with SIPP rasterization (cont.)



Teapot –Gouraud shading

Teapot – Phong Shading



Mount with transparent spheres

Teapot clipped by Sutherland-Hodgman