Cindy3D Project Documentation

Matthias Reitinger Jan Sommer

March 13, 2012

Contents

1 Project overview

1.1. Cinderella & CindyScript

Cinderella is a dynamic geometry software created by Ulrich Kortenkamp and Jürgen Richter-Gebert. One of its key features is the embedded functional scripting language *CindyScript*. It enables users of *Cinderella* to interact with geometric constructions in a programmatic way. Among others, *CindyScript* provides a rich set of methods for drawing two-dimensional geometry. However the display of three-dimensional objects is cumbersome, as one would have to fall back on the 2D drawing methods and implement algorithms like perspective projection and hidden surface removal by hand.

The recent release of *Cinderella 2.6* provides a new plug-in interface. This allows programmers to write *Java* libraries which expose *CindyScript* methods that can in turn be called from inside *Cinderella* constructions.

1.2. Cindy3D

In this context the *Cindy3D* project was born. The need for 3D drawing functionality in *Cinderella* was in fact one of the main reasons to create the new plug-in interface. The first version of *Cindy3D* was developed when the plug-in interface was still in a preliminary stage.

The main goals for *Cindy3D* were as follows:

- Free placement of geometric primitives in 3D space: points, segments, lines, rays, polygons, circles, spheres, and quad meshes
- Control over material parameters: color, shininess, transparency
- Customization of scene parameters: background, lights and camera
- High-quality rendering
- Interactive exploration of the scene

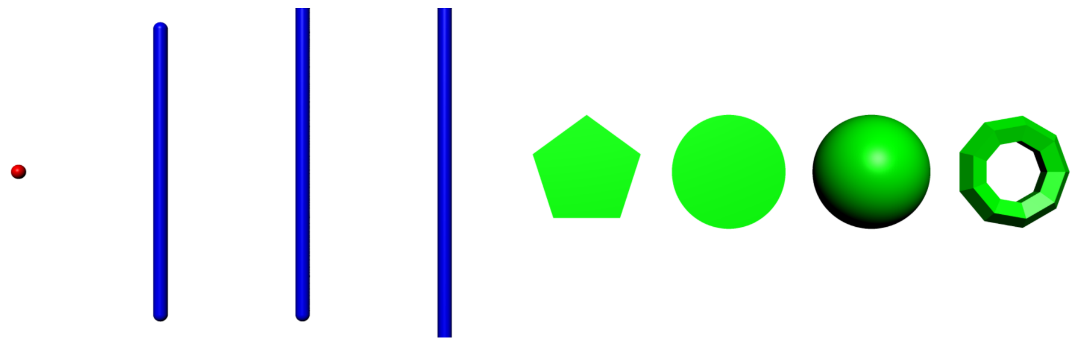
Figure 1.1.: Supported primitives. Point, segment, ray, line, polygon, circle, sphere, and quad mesh.

- User-friendly *CindyScript* interface:
  - Useful set of functions which are exposed to the user
  - Function names similar to the existing 2D drawing functions (if a corresponding function exists)
  - Sensible default parameters
- Support for all major platforms on which *Cinderella* runs: Windows, Mac OS X, Linux
- Hardware acceleration support

1.3. Results

We are confident that we met all the main goals as stated in the previous section. The following screen shots directly taken from a *Cindy3D* session illustrate the capabilities of the plugin. The *CindyScript* source code used to generate these images can be found in appendix A. For details on what can be done with *Cindy3D*, please refer to the user documentation in chapter 2.

1.3.1. Primitive types

Figure 1.1 shows the primitive types we support. They can be combined to produce more complex geometry like in the image on the title page, which consists of a thousand intersecting spheres. The quad mesh supports three different topologies (circular, cylindrical, toroidal) only one of which is shown in the figure.

1.3.2. Material parameters

Figure 1.2 illustrates the different material parameters which can be specified per primitive. The user can either set default parameters or define the parameters in each draw

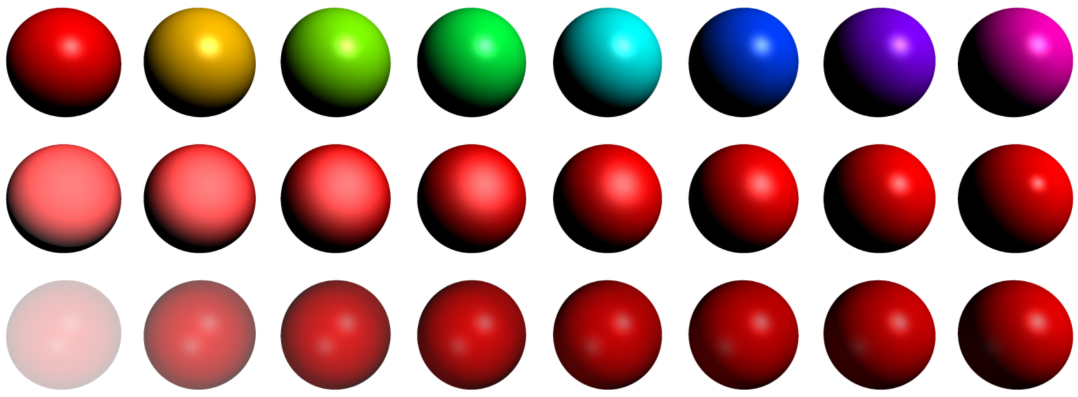

Figure 1.2.: Material parameters. Variations in color (first row), shininess (second row), and transparency (third row).

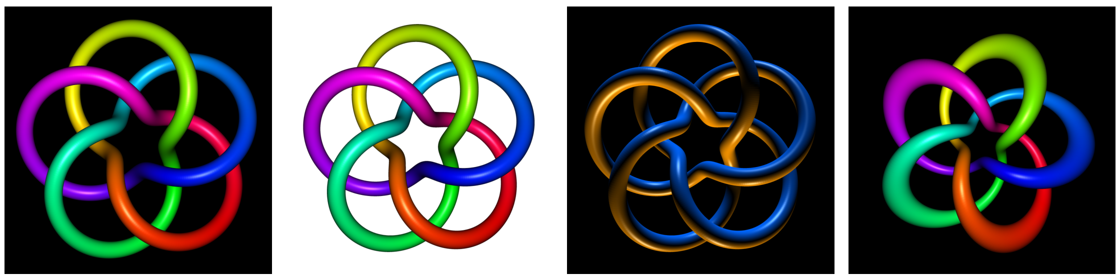

Figure 1.3.: Scene parameters. Changes in (b) background color, (c) light setting and (d) camera parameters are possible.

command with *CindyScript*'s modifier syntax. It's also possible to mix and match both methods and set a default material, overriding it with modifiers in some special cases.

1.3.3. Scene parameters

In figure 1.3 the same geometry is shown four times with different scene parameters. The background can be set to a new color (1.3b). In 1.3c the light setting has been changed to two directional lights coming from above (blue light) and below (orange light). Finally, 1.3d shows the effect of setting the camera's field of view to 120 degrees (the default is 45 degrees). It is also possible to position the camera freely in 3D space.

1.3.4. Rendering quality

The rendering quality of *Cindy3D* can be adjusted depending on the capabilities of the user's machine. Figure 1.4 compares different quality levels, ranging from the low-quality mode displaying triangular approximations of curved geometry (1.4a) to the

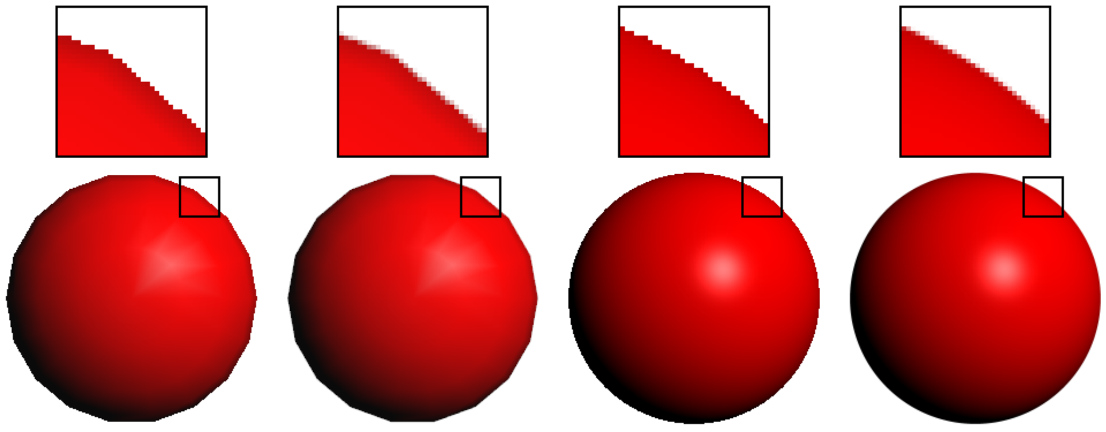

Figure 1.4.: Rendering quality. Different quality settings with low-quality (LQ) or high-quality (HQ) rendering and optional anti-aliasing (AA).

high-quality mode which uses ray casting to find exact intersections with the geometry (1.4c). Each of the two render modes can optionally be enhanced by anti-aliasing techniques to alleviate jaggy borders on object silhouettes (1.4b and 1.4d).

The high-quality mode in conjunction with anti-aliasing is able to produce images comparable to state of the art off-line rendering solutions at interactive rates. It is however more taxing on the hardware, requiring a fairly decent GPU to display complex scenes in the higher quality levels. The low-quality mode is intended for users with slower hardware and trades off image quality for faster rendering time.

1.3.5. Interactive exploration

The generated scene can be explored interactively by the user. The camera's position and orientation can be controlled with the computer mouse. It's possible to rotate, zoom and pan the camera to reach the camera angle you need.

1.3.6. User-friendly interface

We tried hard to make the *CindyScript* interface as user-friendly and consistent as possible. All *Cindy3D* function names end with `3d`, making it easy to spot *Cindy3D* commands in a piece of code. Users already familiar with the 2D drawing functions of *Cinderella* will notice that many functions like `color3d`, `size3d`, `draw3d`, `gsave3d`, etc. work just like their 2D counterparts `color`, `size`, `draw`, `gsave`, etc. A comprehensive command reference is provided as part of this document (section 2.3). Additionally a user guide is available to help new users get up to speed (section 2.2).

1.3.7. Cross-platform support

Cindy3D has been developed with platform independence in mind and has been tested on all major operating systems, i.e. Windows, Mac OS X, and Linux. It will probably also run on other platforms, provided that *Cinderella* supports it and *OpenGL* is available for hardware acceleration.

1.4. Future directions

There are some features we would have wanted in *Cindy3D*, yet didn't make it into this release for various reasons (like limited time or constraints in *Cinderella* at the time of development). We hope that these features can be implemented in further versions of *Cindy3D*:

- Coordinate system transformations (`translate3d`, `rotate3d`, `scale3d`)
- Embedding *Cindy3D* into the *Cinderella* drawing surface
- Applet support
- 2D function plots as additional primitive

2 User documentation

2.1. Installation

- Minimum system requirements (OpenGL version...)
- Install guide (with screenshots?)

2.2. User guide

- Understanding of *CindyScript* is assumed
- Scene management (begin3d, end3d)
- Coordinate system
- Primitive types
- Appearance (attributes, stack)
- Lights and shading model
- Examples
- Tutorial(s)?

2.3. Command reference

Link to HTML documentation or inline.

3 Developer documentation

3.1. Technical overview

- Java 6
- Used libraries: JOGL, apache-commons-math
- JAR packaging

3.2. Design overview

Insert fancy diagram here.

3.3. Rendering

- Explain raytraced rendering with proxy geometry
- Explain LOD

3.4. JavaDoc

Link to generated JavaDoc.

A

Example source code

List of Figures