



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Liner Builder: A Tool for Building Geometric Models of Shaped Charge Liners

W.D. Henshaw

January 12, 2005

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Liner Builder: A Tool for Building Geometric Models of Shaped Charge Liners

William D. Henshaw

Centre for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, 94551
henshaw@llnl.gov

January 7, 2005

Abstract

This document describes the *Liner Builder*, an interactive graphics tool for building a geometric model of the conical liner that fits in the hollow cavity on the end of a shaped charge. This tool can also be used to load the volume of the liner with a distribution of spherical particles.

Contents

1	Introduction	3
2	Getting started	4
3	The Liner Builder Dialog	5
4	Example: a piecewise linear liner	7
5	Example: a piecewise quadratic liner	9
6	Example: a free-form liner	11
7	Defining a piecewise-linear liner cross-section	14
8	Defining a piecewise-quadratic liner cross-section	15
9	Defining a free-form liner cross-section	16
10	Building the volume of revolution from the cross-section curve	17
11	Loading a liner with spheres	17
12	Lighting curves - specifying the velocity of each particle	18
A	Appendix: The Nurbs Curve Builder	20

B Appendix: Features of the Overture Graphical User Interface	22
B.1 Using the mouse for rotating, scaling and picking	23
B.2 Using clipping planes	24
B.3 Setting the view characteristics	24
B.4 Changing the default appearance of the windows	26

Acknowledgments.

Bill Bateson and Dale Slone have contributed to the design of the *Liner Builder*. Dale has also contributed directly to the development of the *Liner Builder*, in particular the loading of the liner with spheres. The *Liner Builder* is built upon components of the Overture framework (including Rapsodi). Many people have worked on the development of Overture but Kyle Chand and Anders Petersson, in particular, have made important contributions to the components used by the *Liner Builder*.

1 Introduction

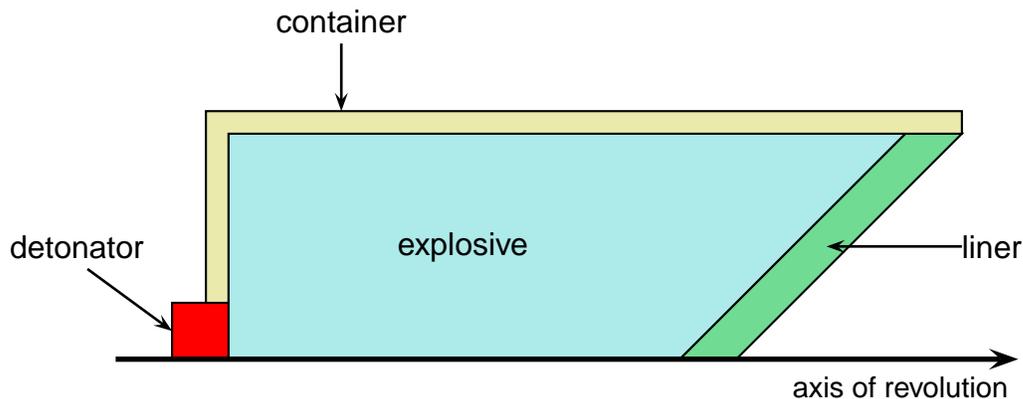


Figure 1: Basic geometry of a shaped charge. When revolved about the axis of revolution, the liner will form a conical shell.

This document describes the *Liner Builder*, an interactive graphics tool for building a geometric model of the conical liner that fits in the hollow cavity on the end of a shaped charge.

Figure 1 illustrates the basic geometry of a shaped charge. The shaped charge in this figure is assumed to be cylindrically symmetric with the axis of revolution shown. The liner is the thin piece of material that is located at the end of the shaped charge opposite to the detonator. The liner will typically be of the shape of a conical shell. When the detonator is set off, it will initiate a detonation that propagates through the explosive. The detonation wave will strike the liner material and accelerate it. The liner will then move and focus toward the axis of revolution, forming a high speed jet of material. The tip velocity of the jet can apparently exceed Mach 25 in air, although the flow velocity of the material remains subsonic. The liner of a shaped charge can take on many shapes such as hemisphere, cone, ellipse, tulip, trumpet, pyramid etc. The liner itself may consist of a thin layer of metal, plastic, ceramic or other similar material. Shaped charges have their largest use in the mining and oil well industries. For further details on shaped charges, see for example, *Fundamentals of Shaped Charges* by Walters and Zukas[1].

The basic steps in designing a liner with the the *Liner Builder* are

1. Create a two-dimensional curve defining the cross-section of the liner. There are various options available for defining this curve such as using a piece-wise linear curve, a piece-wise quadratic curve or a free-form curve.
2. rotate the cross-section curve to form a surface of revolution

Currently the liner shapes are restricted to be surfaces of revolution although it is envisioned that general three-dimensional shapes could be supported.

Given a liner there are additional steps that can be taken to load the liner with a collection of spheres to be used in a particle simulation of the liner motion. The spheres represent a model of the liner material.

1. define the properties of the spherical particles such at the distribution of radii; the volume and mass fractions in the loading, ...
2. *load* the interior of the liner with a distribution of spherical particles.
3. define the *lighting-curves* that define the initial velocity of the spheres to be used in a particle simulation of the liner motion.

The *Liner Builder* is part of the more general `rap` program. (The name `rap` is derived from the Rapsodi project – the name `rap` name will probably change in the future to something like `modelBuilder`). The `rap` program can be used to clean up, fix and modify CAD geometries as well as build new geometries.

2 Getting started

To start the *Liner Builder*, type `rap` to bring up the startup screen for `rap`. The `rap` program is usually found in the `Overture/bin` directory or in the `Dune bin` directory(?) Choose the button “`liner...`” to open up the *Liner Builder* dialog window. At this point the screen should look similar to figure 2. You are now ready to begin the design process.

Note that as commands are entered interactively, the command file `rap.cmd` is being saved with a text version of each command. This command file can later be given a new file name, edited with any text editor, and used to rerun the session.

Before designing a new liner it may be useful to first run through some examples. See sections 4, 5 and 6 for three examples.

The left mouse button is used for selecting buttons on the windows and dialogs. The left mouse button is also used for picking points on the graphics window. Clicking the right mouse button on any window will bring up any pop-up menu. Appendix B is a useful reference for understanding the behaviour of the buttons, mouse and view characteristics of the the graphics window.

3 The Liner Builder Dialog

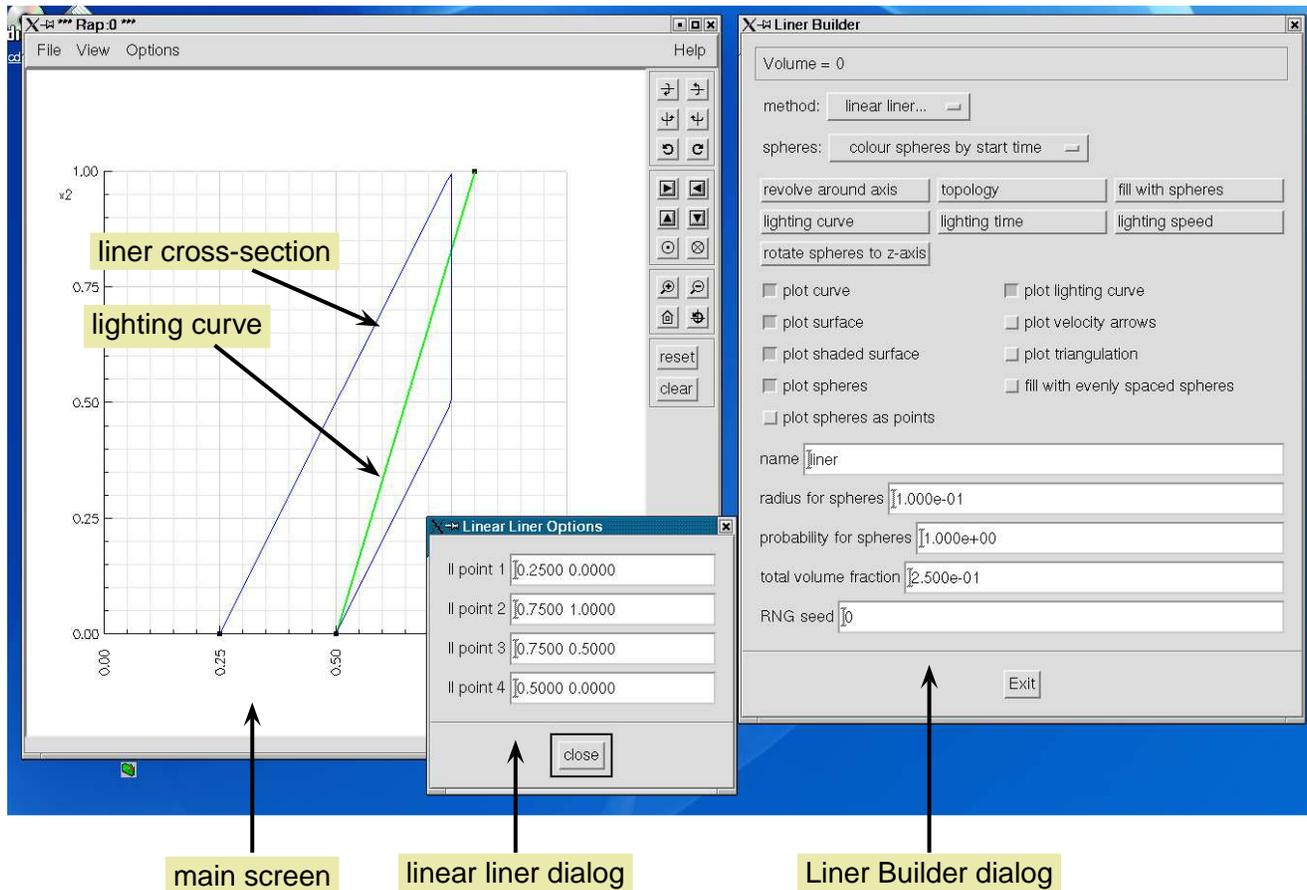


Figure 2: Liner Builder windows and dialogs. The main screen plots the results. The *Liner Builder* dialog controls the generation of the shaped charge liner.

The windows associated with the *Liner Builder* are shown in figure 2. The main graphics window displays the liner cross-section and the lighting curve. The main dialog shown controls the *Liner Builder*. A smaller dialog appears for inputting changes to the linear cross-section (since the method is currently set to “linear liner”).

Here are descriptions of the buttons on the Liner Builder dialog

method Choose the type of liner: “linear liner”, “quadratic liner” or “free form liner”. Selecting a new method will cause a new dialog to appear, specific to the liner type, from which parameters can be set.

spheres Choose the technique for colouring the spheres that fill the liner volume.

revolve around axis After the liner cross-section has been created, choose this option to build a surface of revolution. At the same time a triangulation covering the surface will be created. This triangulation is used for the rapid determination of whether a point is inside or outside the surface.

topology this option will recompute the triangulation. Normally this option is not directly used unless a poor quality triangulation is generated from the step “revolve around axis”.

fill with spheres After the properties of the spherical particles have been defined, choose this option to fill the liner with spheres.

lighting curve edit the lighting curve which determines the direction of propagation of the particles, see section 12.

lighting time edit the lighting time function which determines the time delay of the motion of the particles, see section 12.

lighting speed edit the lighting speed function which determines the speed of the particles, see section 12.

rotate spheres to the z-axis rotate the spherical particles so that the axis of revolution points along the z-axis. Use this option before outputting the sphere locations for use with some discrete simulation codes.

Here are descriptions of the toggle buttons

plot curve turn on or off the plotting the liner cross-section curve.

plot surface turn on or off the plotting of the surface of revolution.

plot shaded surface turn on or off the plotting the shaded surface on the surface of revolution. When the shaded surface is turned off, only the grid lines on the surface will appear allowing one to see inside the surface.

plot spheres turn on or off the plotting of the spheres (once they have been computed).

plot spheres as points turn on or off the plotting of the spheres as points (it is quicker to draw the spheres if they are just shown as points).

plot lighting curve turn on or off the plotting of the lighting curve.

plot velocity arrows turn on or off the plotting of velocity arrows emanating from each sphere. The velocity arrows indicate the initial velocity vector of the spheres.

plot triangulation turn on or off the plotting of the underlying triangulation that is associated with the liner surface.

fill with evenly spaced spheres for testing purposes, fill the liner with evenly spaced spheres.

Here are descriptions of the text fields

name Choose the name of the liner.

radius for spheres Enter a list of N different sphere radii, $R_1, R_2, \dots R_N$.

probability for spheres Enter the N probabilities, $P_1, P_2, \dots P_N$, for each of the radii specified in the “radius for spheres” text box.

total volume fraction Enter the total volume fraction (a real number between 0 and 1) of the liner volume that should be filled with spheres. As the volume fraction is increased it becomes more difficult to fill the volume with spheres.

RNG seed choose a seed for the random number generator that is used when loading the volume with spheres.

4 Example: a piecewise linear liner

In this example a linear liner is built and filled with spherical particles, see figure 3. The command file for generating these results is given here (file `linerLinear.cmd`):

```

1 *
2 * rap command file to build a shaped charge liner and fill it with spherical particles
3 * This example creates a piecewise linear liner.
4 * To run this example type:
5 *     rap linerLinear.cmd
6 *
7 liner...
8 * choose points on the piecewise linear liner
9   linear liner...
10  ll point 1 0 0
11  ll point 2 2.28284 2.28284
12  ll point 3 2.28284 2
13  ll point 4 .28284 0
14  close linear liner
15  pause
16 * create the surface of revolution
17  revolve around axis
18 * load the liner volume with spheres
19  radius for spheres .1 .05 .01
20  probability for spheres .2 .3 .5
21  total volume fraction 1.500e-01
22  RNG seed 97845
23  pause
24  fill with spheres

```

Note that comments in the command file are denoted by a leading asterisk. Also note that a blank line will be treated as the end of the command file. Thus, adding a blank line is one way to stop the program at a particular point in the command file.

This command file can be run using the command “`rap linerLinear.cmd`”. As the example runs the program will pause at various points (indicated by the `pause` statements in the command file). Choose `continue` to proceed with the next commands or choose `break` to break out of the command file.

To run the command file in batch mode with no plotting and no pausing use “`rap noplot nopause linerLinear.cmd`”.

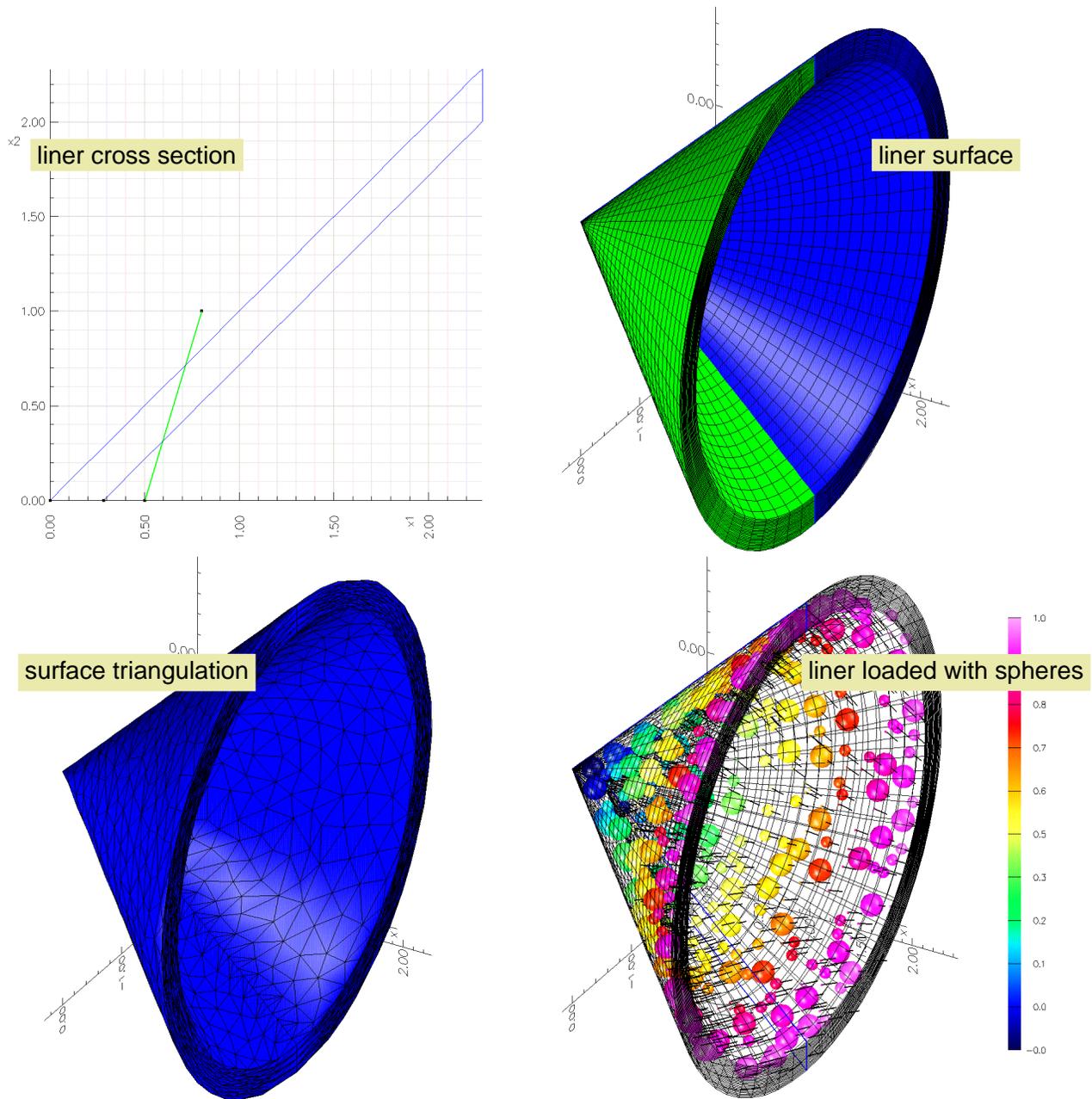


Figure 3: Results from running the `linerLinear.cmd` command file. Upper left: the liner cross-section is defined. Upper right: the surface of revolution for the liner. Lower left: the underlying triangulation of the surface. Lower right: the liner volume loaded with spheres, coloured by start time and with direction arrows shown.

5 Example: a piecewise quadratic liner

In this example a piecewise quadratic liner is built and filled with spherical particles, see figure 4. The command file for generating these results is given here (file `linerQuadratic.cmd`):

```

1 *
2 * rap command file to build a shaped charge liner and fill it with spherical particles
3 * This example builds a liner built from two quadratic sections.
4 * To run this example type:
5 *     rap linerQuadratic.cmd
6 *
7 liner...
8 quadratic liner...
9 * define points on the piecewise quadratc
10 ql point 1 0.2500 0.0000
11 ql point 2 0.3964 0.3536
12 ql point 3 1. .75
13 *
14 ql point 4 0.4000 0.0000
15 ql point 5 0.4732 0.1768
16 ql point 6 0.9000 .55
17 pause
18 *
19 * build the surface of revolution
20 *
21 revolve around axis
22 pause
23 *
24 * load with spheres
25 *
26 radius for spheres .05 .04 .03 .01
27 probability for spheres .1 .2 .3 .4
28 total volume fraction .25
29 RNG seed 4278
30 fill with spheres
31 *
32 pause
33 * plot spheres inside the grid
34 plot curve 0
35 plot lighting curve 0
36 plot shaded surface 0

```

This command file can be run using the command “`rap linerQuadratic.cmd`”. As the example runs the program will pause at various points (indicated by the `pause` statements in the command file). Choose `continue` to proceed with the next commands or choose `break` to break out of the command file.

To run the command file in batch mode with no plotting and no pausing use “`rap noplot nopause linerQuadratic.cmd`”.

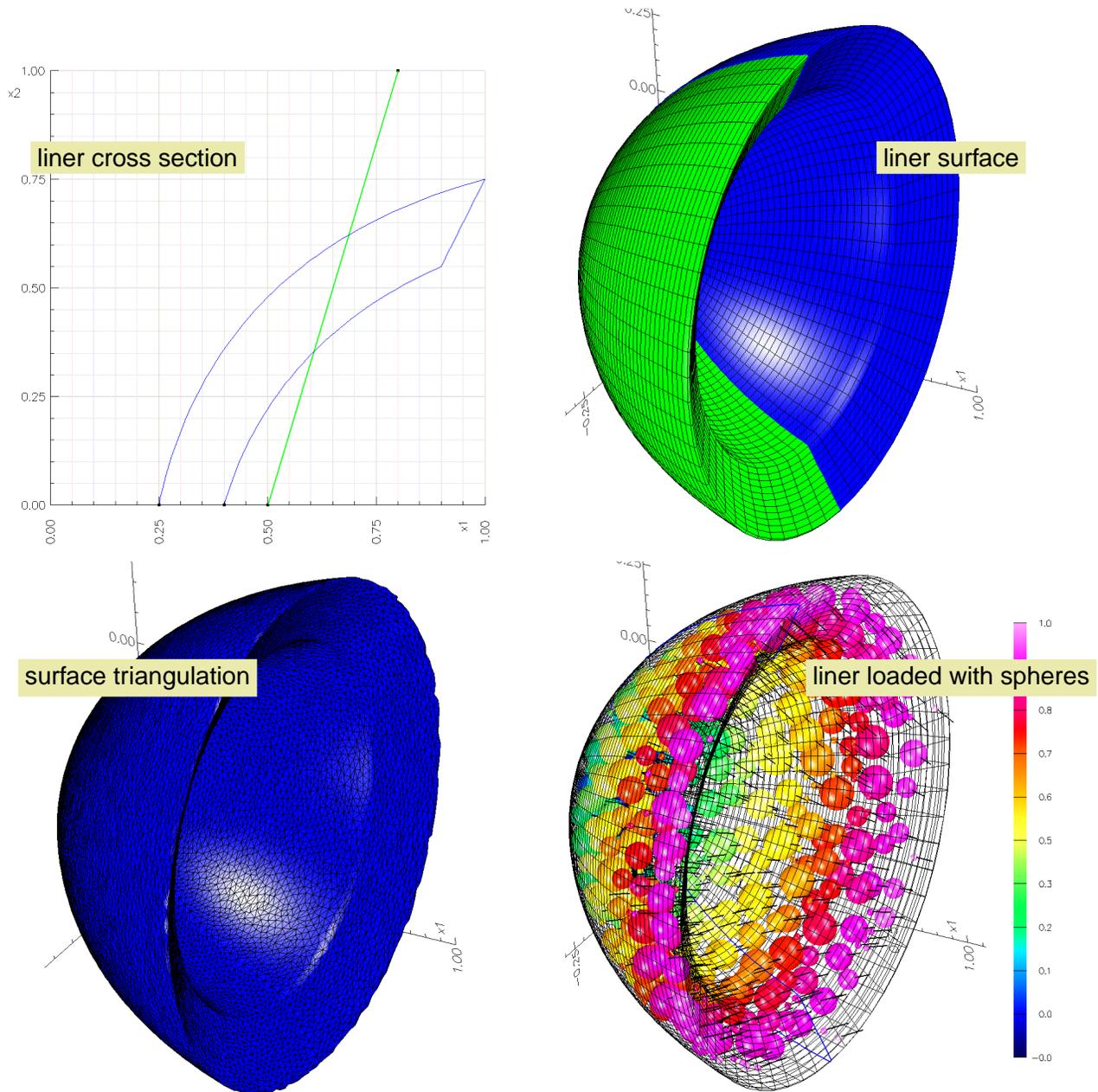


Figure 4: Results from running the `linerQuadratic.cmd` command file. Upper left: the liner cross-section is defined. Upper right: the surface of revolution for the liner. Lower left: the underlying triangulation of the surface. Lower right: the liner volume loaded with spheres, coloured by start time and with direction arrows shown.

6 Example: a free-form liner

In this example a free-form liner is built and filled with spherical particles, see 5. The command file for generating these results is given here (file `linerFreeForm.cmd`):

```

1  *
2  * rap command file to build a shaped charge liner and fill it with spherical particles
3  * This example will build a liner using a free form curve.
4  * To run this example type:
5  *     rap linerFreeForm.cmd
6  *
7  liner...
8  linear liner...
9  * build a free form liner consisting of three sub-curves
10 free form liner...
11 specify free form liner
12 * first build points (from which the sub-curves will be generated)
13 Mouse Mode Build Point
14     new point .2      0
15     new point 3.323187e-01 4.975356e-01
16     new point 7.024078e-01 8.494369e-01
17     new point 8.263645e-01 7.352899e-01
18     new point 6.796839e-01 6.050027e-01
19     new point 5.495047e-01 4.547871e-01
20     new point 6.026765e-01 3.258095e-01
21     new point 4.559958e-01 3.029903e-01
22     new point 4.083246e-01 1.587799e-01
23     new point 3.991570e-01 0
24 pause
25 * create the first sub-curve by choosing 3 points
26 Mouse Mode Interpolate Curve
27     point for interpolation 0
28     point for interpolation 1
29     point for interpolation 2
30     stop picking
31 pause
32 * create the second sub-curve
33     point for interpolation 2
34     point for interpolation 3
35     stop picking
36 *
37     point for interpolation 3
38     point for interpolation 4
39     point for interpolation 5
40     stop picking
41 *
42     point for interpolation 5
43     point for interpolation 6
44     point for interpolation 7
45     stop picking
46 *
47     point for interpolation 7
48     point for interpolation 8
49     point for interpolation 9
50     stop picking
51 *
52 * join the sub-curves to form one curve

```

```
53  assemble
54    assemble
55  exit
56  pause
57  * build the surface of revolution
58  revolve around axis
59  pause
60  *
61  * fill the volume with evenly spaced spheres
62  fill with evenly spaced spheres 1
63  radius for spheres .025
64  fill with spheres
65  *
66  plot curve 0
67  plot lighting curve 0
68  plot shaded surface 0
```

This command file can be run using the command “rap linerFreeForm.cmd”. As the example runs the program will pause at various points (indicated by the pause statements in the command file). Choose continue to proceed with the next commands or choose break to break out of the command file.

To run the command file with no plotting and no pausing use “rap noplot nopause linerFreeForm.cmd”.

The free-form cross-section curve in this example is created using the *Nurbs Curve Builder*, see section 9 for further details on building the free form liner and Appendix A for a description of the *Nurbs Curve Builder* commands.

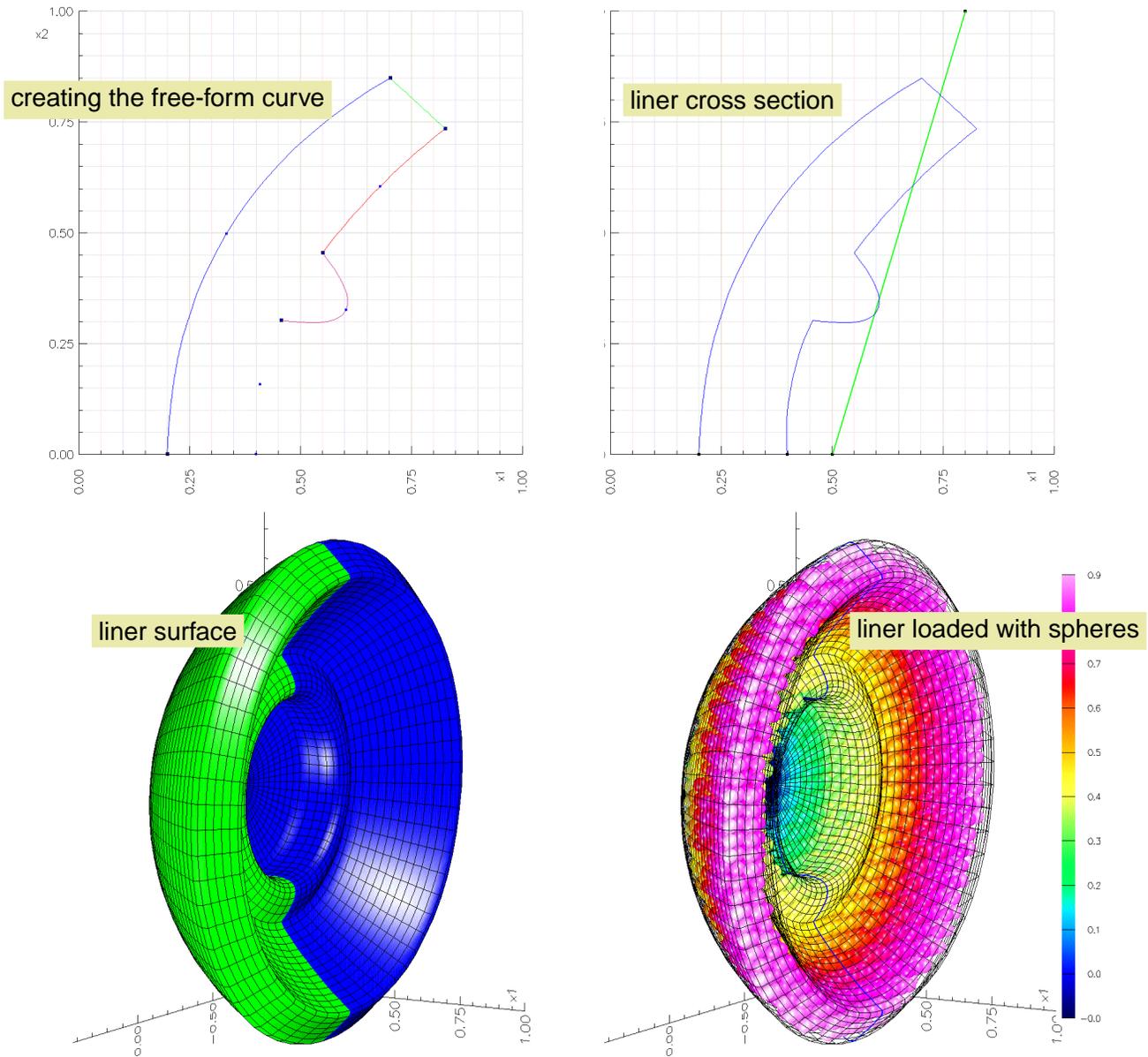


Figure 5: Results from running the `linerFreeForm.cmd` command file. Upper left: the liner cross-section is created using the NURBS curve builder. Upper right: the completed liner cross-section. Lower left: the surface of revolution for the liner. Lower right: the liner volume loaded with spheres, coloured by start time.

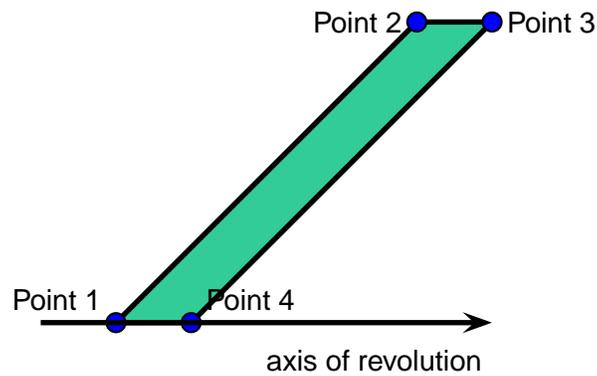


Figure 6: The cross-section of the piece-wise linear liner is defined by 4 points.

7 Defining a piecewise-linear liner cross-section

The linear liner is defined by four points are shown in figure 6. The coordinates of any of the four points can be altered. Points 1 and 4 should remain on the axis of revolution, $y = 0$.

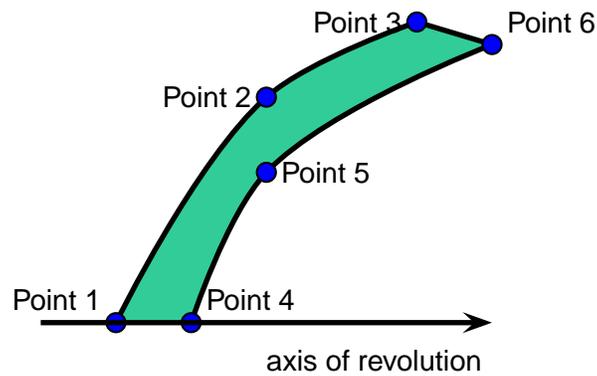


Figure 7: The cross-section of the piece-wise quadratic liner is defined by 6 points. A quadratic curve is interpolated through the points 1,2,3. Another quadratic passed through the points 4,5,6.

8 Defining a piecewise-quadratic liner cross-section

The piecewise quadratic liner is defined by 6 points as shown in figure 7. A quadratic curve is interpolated through the points 1,2,3. Another quadratic passed through the points 4,5,6. Points 1 and 4 should remain on the axis of revolution, $y = 0$.

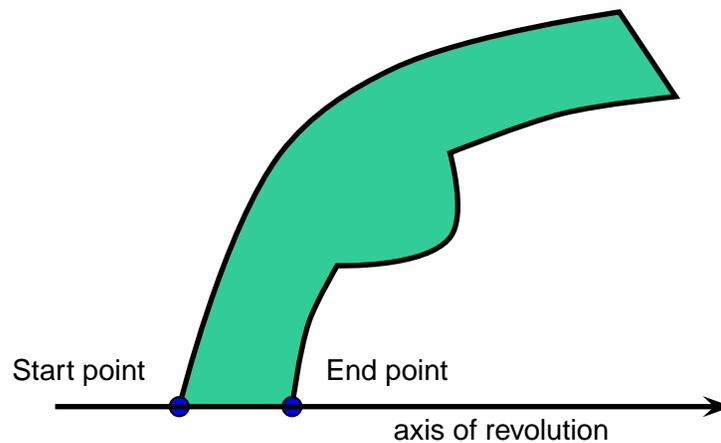


Figure 8: The free-form liner cross-section is defined by a set of sub-curves that are joined together to form a piecewise smooth curve.

9 Defining a free-form liner cross-section

The free-form liner makes use of a *Nurbs Curve Builder* to build a free form curve. A NURBS is a generalized type of spline (non-uniform rational B-spline). The free form curve is built from a set of smooth sub-curves that are joined together. The global curve is allowed to have corners where the sub-curves meet.

The basic steps in building a free form curve are (see the example in section 6)

- interactively choose (or manually input) a collection of points in the two-dimensional plane that will be used to form the sub-curves. If you have a lot of points to input you may want to manually edit one of the example command files.
- select a sub-set of the points to form a smooth sub-curve. The sub-curve will be created by interpolating the chosen points. Repeat this process for each sub-curve.
- Choose *assemble* to join the sub-curves into a single piecewise smooth curve. Note: even if only one sub-curve is built it is necessary to choose *assemble*.

Note that any points created on the axis of revolution should be placed exactly at $y = 0$. It may be necessary to manually enter these points or to edit and change the values in the command file if the point was created interactively.

For further information on the *Nurbs Curve Builder* see the Appendix A.

10 Building the volume of revolution from the cross-section curve

The “revolve around axis” button on the *Liner Builder Dialog* is used to build a surface of revolution from the liner cross-section curve. At the same time a triangulation covering the surface will be created. This triangulation is used for the rapid determination of whether a point is inside or outside the surface.

Geometric properties of the linear are computed from the triangulation. These properties are printed to the screen and include the enclosed volume, center of mass and the moments of inertia matrix (computed assuming a uniform density throughout the liner).

The triangulation should be automatically generated although sometimes errors may occur. Errors in generating the triangulation most commonly occur when the ends of the cross-section curve do not exactly touch the axis of revolution.

11 Loading a liner with spheres

The current specification for loading the liner volume with spheres is just an initial attempt. A more sophisticated approach is envisioned for the future.

Currently the basic steps to fill the liner volume with spheres are

1. specify a list of N values, R_i , indicating the radii of the spheres.
2. specify a list of N values, P_i , indicating the probability that a sphere will radius R_i will appear.
3. Specify the fraction of the volume that should be occupied by the spheres.

Choosing the “fill liner spheres” button will cause the sphere loading algorithm to be called. The current algorithm is rather primitive and can be slow even for moderate volume fractions. An improved algorithm is under development.

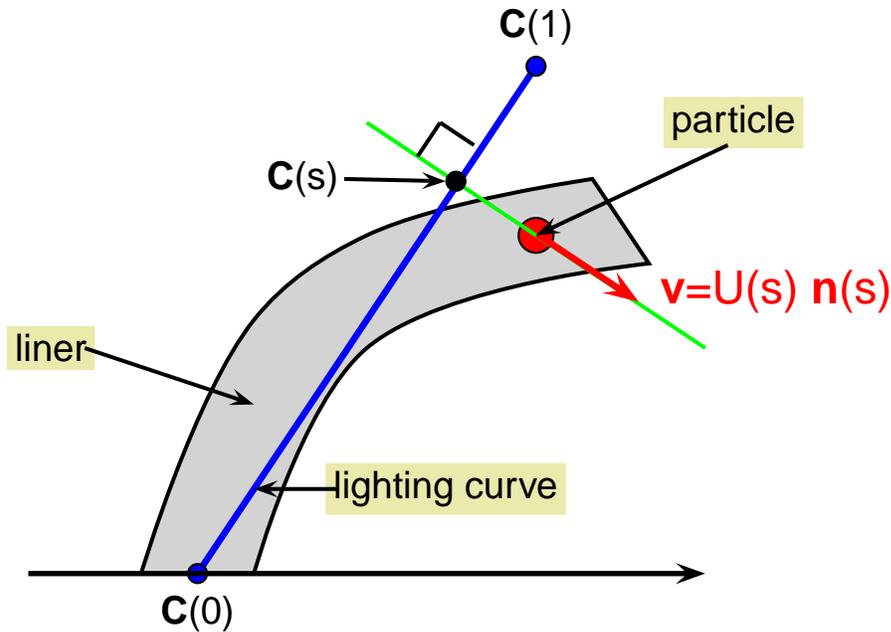


Figure 9: The lighting curve is $\mathbf{x} = \mathbf{C}(s)$, $s \in [0, 1]$. The lighting speed is $U(s)$. Any particle which lies along the normal, $\mathbf{n}(s)$ to the point $\mathbf{C}(s)$ is given a speed $U(s)$ and direction equal to the normal to $\mathbf{C}(s)$.

12 Lighting curves - specifying the velocity of each particle

Lighting curves are used to assign velocities to the particles in the liner. These curves are used to model the effect of the actual detonation hitting the liner material. There are three lighting functions, the *lighting curve* $\mathbf{C}(s)$, the *lighting speed* $U(s)$ and the *lighting time* $T(s)$.

Figure 9 shows how the lighting curves are used to assign a velocity to the particles in the liner volume. Let $\mathbf{x} = \mathbf{C}(s)$, $s \in [0, 1]$, denote the lighting curve. Let $U(s)$, $s \in [0, 1]$, denote the lighting speed.

To determine the velocity of a particle at position \mathbf{p} , first determine the value of s such that the line normal to the lighting curve at the point $\mathbf{C}(s)$ intersects the point \mathbf{p} . Denoting the normal to $\mathbf{C}(s)$ by $\mathbf{n}(s)$, then the velocity for the particle at the point \mathbf{p} is $\mathbf{u} = U(s)\mathbf{n}(s)$.

An additional *lighting time* curve, $T(s)$, defines when each particle begins to move. For the particle at position \mathbf{p} , the velocity would be zero for $t < T(s)$ and would equal $U(s)\mathbf{n}(s)$ for $t > T(s)$.

The lighting curves $\mathbf{C}(s)$, $U(s)$ and $T(s)$ are given default values. Each of these curves can be edited by choosing one of the buttons labeled “lighting curve”, “lighting speed”, or “lighting time”. The curve editor is the *Nurbs Curve Builder*, the same that is used to define a free form curve (section 9). The *Nurbs Curve Builder* is described in Appendix A.

NOTE: The lighting speed, $U(s)$ and the lighting curve $T(s)$ are really scalar valued functions of the parameter s ; In other words, U is a mapping from the unit interval onto the real numbers, $U : [0, 1] \rightarrow \mathbb{R}$. However, to make it convenient to graphically edit these functions, these two lighting functions are represented as parameterized curves. Thus $U(s)$ becomes the curve $\mathbf{U}(\xi) = (x_u(\xi), y_u(\xi))$ for $\xi \in [0, 1]$, while $T(s)$ becomes $\mathbf{T}(\eta) = (x_t(\eta), y_t(\eta))$ for $\eta \in [0, 1]$. Of course it only makes sense that the parameterized curves $\mathbf{U}(\xi)$ and $\mathbf{T}(\eta)$ be single valued functions of ξ and η . Given a point $\mathbf{C}(s)$, in order to compute $U(s)$ it is necessary to relate s to the parameter value ξ of the curve $\mathbf{U}(\xi) = (x_u(\xi), y_u(\xi))$. This is done by finding ξ such that $x_u(\xi) = s$ and then $U(s) = y_u(\xi)$. In other words, the “ x ” coordinate of the parameterized curves corresponds to s . Written compactly this becomes $U(s) = y_u(x_u^{-1}(s))$. Similarly the value of the lighting time equals $T(s) = y_t(x_t^{-1}(s))$.

In summary here is the algorithm for specifying the velocity of a particle that is located at the point \mathbf{p} :

1. Find s, α such that $\mathbf{C}(s) + \alpha \mathbf{n}(s) = \mathbf{p}$. Here $\mathbf{C}(s)$ should be the closest point on the curve to \mathbf{p} while $\mathbf{n}(s)$ is the normal to the curve at $\mathbf{C}(s)$.
2. Compute the speed $U(s) = y_u(x_u^{-1}(s))$ where $\mathbf{U}(\xi) = (x_u(\xi), y_u(\xi))$ is the parameterized lighting speed curve.
3. Compute the speed $T(s) = y_t(x_t^{-1}(s))$ where $\mathbf{T}(\eta) = (x_u(\eta), y_u(\eta))$ is the parameterized lighting time curve.

A Appendix: The Nurbs Curve Builder

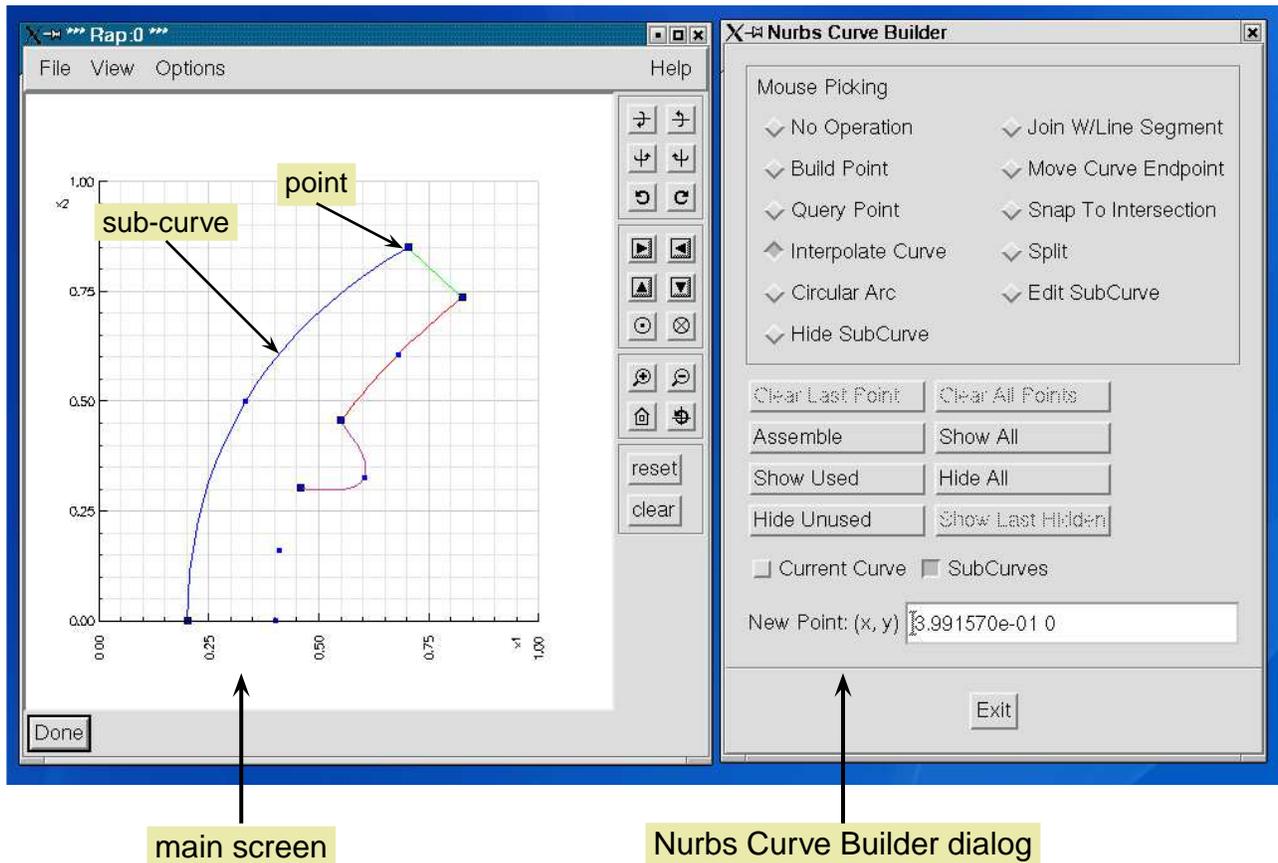


Figure 10: Creating sub-curves with the Nurbs Curve Builder. The sub-curves can be joined to form a piecewise smooth curve.

The *Nurbs Curve Builder* can be used to create and modify two-dimensional curves. The curves are represented as a NURBS (non-uniform rational B-spline) which is a generalized type of spline. The *Nurbs Curve Builder* dialog is shown in figure 10. The Curve Builder allows one to create a curve consisting of a set of smooth sub-curves that are joined together. The assembled curve is allowed to have corners where the sub-curves meet. Note that there are two types of curves plotted in the graphics window, the sub-curves and the assembled curve. Plotting of the assembled curve is controlled by the “Current Curve” toggle button.

The basic steps to build a curve are to first create a set of points and then choose a subset of these points to interpolate a sub-curve. Sub-curves are assembled into a single curve.

Build Points To build points select the “Build Point” mouse mode toggle. Points may now be chosen interactively by clicking with the mouse. Alternatively, the coordinates of a point can be typed into the text box, “New point (x,y)”, on the dialog window. It may be necessary to first set the *plot bounds* (see below) so that the plotting region is the correct size.

Interpolate Curve To create a curve from points select the “Interpolate Curve” mouse mode toggle. Points can be selected with the mouse. Choose done (from the lower left corner of the main window) when a sub-curve is completed.

Assemble Choose `Assemble` to join the set of sub-curves (where possible) into a single curve. `Assemble` should be chosen even if there is only one sub-curve.

Additional mouse modes are available including

Query Point With this mode on, selecting a point will cause the point label and coordinates to be printed.

Circular Arc build a circular arc from two points and a radius of curvature (choose a negative value for the radius of curvature to form the other possible arc that passes through the two points.)

Hide SubCurve hide a sub-curve (the hidden sub-curve will not be considered during the assemble step).

Move Curve Endpoint move the endpoint of one sub-curve to match the endpoint of another curve. First select a curve endpoint and then select a point near the end of another curve.

Join W/Line Segment Join the endpoints of two sub-curves with a line segment by selecting two points near the ends of the sub-curves.

Snap to Intersection If two sub-curves intersect, “`Snap to Intersection`” will join the sub-curves at the point of intersection and discard the extra segments. Select each sub-curve by clicking a point that lies on the portion of the sub-curve that should be retained.

Split split an existing curve into two pieces by selecting a point on the curve.

Edit SubCurve Enter the Nurbs editor to edit the properties of a sub-curve.

Here are descriptions of the buttons that appear in the *Nurbs Curve Builder* dialog,

Clear Last Point Remove the last point that was selected (when selecting points for “`Interpolate Curve`”, for example.)

Clear All Points Remove all selected points (when selecting points for “`Interpolate Curve`”, for example).

Assemble Join sub-curves (where possible) into an assembled curve (the current curve).

Show All Show all sub-curves and the assembled curve.

Show Used Only show sub-curves that are used in the assembled curve.

Hide all Hide all sub-curves.

Hide Unused Hide unused sub-curves.

Show Last Hidden Show the last hidden sub-curve.

The toggle buttons are

Current Curve plot the assembled curve if it has been created.

SubCurves plot the sub-curves.

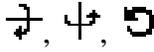
The text strings that appear in the *Nurbs Curve Builder* are

New point (x,y) build a new point by typing the x and y coordinates.

plot bounds Enter new plot bounds as x_{\min} , x_{\max} , y_{\min} , y_{\max} . Points can only be built interactively if the new coordinates lie inside the current plot bounds. Thus it may be necessary to increase the plot bounds before building new points.

B Appendix: Features of the Overture Graphical User Interface

This section describes some of the features and properties of the Overture graphical user interface which is illustrated in figure 11. Some of these features are

- User defined dialog windows that can contain pulldown menus with push or toggle buttons, option menus, text labels (for inputting strings), push buttons, and toggle buttons.
- Multiple graphics windows and a single command window with a scrollable sub-window for outputting text, a command line, and a scrollable list of previous commands.
- Rotation buttons  ,... which rotate the object on the screen about fixed x, y, and z axes (the x axis is to the right, the y-axis is up and the z-axis is out of the screen).
- Translation buttons  ,... which shift the object on the screen along a given axis. The two last buttons shift the object in and out of the screen, respectively. Since an orthographic projection method is used in the graphics interface, these buttons only change the appearance when clipping planes are used.
- Push buttons for making the objects bigger: , or smaller: , and a reset button:  to reset the view point, and a **clear** button to erase all objects on the screen.
- A push button  to set the rotation center.
- A **rubber band zoom** feature.
- Mouse driven **translate, rotate and zoom**.
- A pop-up menu that is active on the command and graphics windows. This menu is defined by the application (= user program).
- Static pull-down menus (**file, view, and help**) on the graphics windows. Here, the screen can be saved in different formats, clipping planes and viewing characteristics can be set, annotations can be made (not fully implemented), and some help can be found.
- Static pull-down menus (**file and help**) on the command window. Here command files can be read/saved, new graphics windows can be opened, the window focus can be set, and the application can be aborted.
- An optional pull-down menu (**My menu** in this case) that is defined by the application.
- Pushbuttons (**Pick 3D** and **Plot** in this case) that are defined by the application.
- A file-selection dialog box (not shown in the figure).
- The option of typing any command on the command line or reading any command from a command file. All commands can be entered in this fashion, including any pop-up or pull-down menu item or any of the buttons, **x+r:0, y-r:0, x+:0, y+:0, bigger:0**, etc. For the buttons, the :0 refers to the window number where the view should be modified, which in this case is window #0. Furthermore, when typing a command, only the first distinguishing characters need to be entered.
- Recording or retrieving a command sequence in a command file.

B.1 Using the mouse for rotating, scaling and picking

Rotations are performed with respect to axes that are fixed relative to the screen. The x-axis points to the right, the y-axis points upward and the z-axis points out of the screen. Rotations can either be performed about the centre of the window, or about a user defined point. This point can be set by first pressing the **set rotation point** icon on the graphics window and then clicking on the screen with the left mouse button. The rotation point can also be set by opening the “Set View Characteristics” dialog from the “View” pull-down menu (see section B.3 for details). Note that the centre of the window is changed with the translation commands **x+**, **x-**, **y+**, etc.

Typing a rotation command with an argument (on the command line), such as **x+r:1 45**, will cause the view in window number 1 to rotate by 45 degrees about the x-axis. If no window number is given with the commands, such as, **y-y 30**, the command will apply to window 0.

Typing a translation command with an argument, such as **x+:0 .25** will cause the view in window number 0 to move to the right .25 units (in normalized screen coordinates; the screen goes from -1 to 1).

Rubber band zoom: The middle mouse button is used to ZOOM in. Press the middle button at one corner of a square, drag the mouse to another corner and lift the button. The view will magnify to the square that was marked. Use ‘reset’ to reset the view.

Mouse driven translate, rotate and zoom: All these operations are performed with the SHIFT key down. To translate, you hold the SHIFT key down, press the left mouse button and drag the cursor; the plotted objects will translate in the same direction as the mouse is moved. To rotate the view, you hold the SHIFT key down and press the middle mouse button and drag the cursor. Moving the cursor left or right will rotate about the y-axis (the vertical screen direction) and moving up or down will rotate about the x-axis (horizontal screen direction). To zoom in or out, you hold the SHIFT key down, press the right mouse button and drag the cursor vertically. To rotate about the z-axis, you press the left mouse buttons and drag the cursor horizontally.

Picking (aka selecting): While clicking the left mouse button, you select an object and get the (x, y, z) coordinate of the point on the object where you clicked. The object that was selected is reported in the selectionInfo data structure, which holds the global ID number, the front and back z-buffer coordinates and the window and 3-D coordinates. The global ID number can be used by the application to identify the selected object.

It is also possible to select several objects on the screen by specifying a rectangular region. To do this, you press the left mouse button in one corner of the rectangle and drag it to the diagonally opposite corner of the rectangle, where the mouse-button is released. The program will draw a rectangular frame to indicate the selected region. When you are happy with the selected region, you release the mouse button. An imaginary viewing volume is defined by translating the rectangular region into the screen and all objects that intersect the viewing volume are selected. However, only the 3-D coordinate of the closest object is computed. We remark that the object with the lowest front z-buffer value is the closest to the viewer. Also note:

1. Objects that are hidden by another object are also selected by this method.
2. The selection takes clipping planes (see below) into account, so it is not possible to select an object that has been removed by a clipping plane.

The mouse driven features are summarized in table 1.

Modifier	Mouse button	Function
	left	picking
	middle	rubber band zoom
	right	pop-up menu
<SHIFT>	left	translate
<SHIFT>	middle	rotate around the x & y axes
<SHIFT>	right	zoom (up & down) and z-rotation (left & right)

Table 1: Mouse driven features.

B.2 Using clipping planes

The clipping plane dialog for each graphics window is opened from the ‘View’ pull-down menu on the menu bar in the graphics window. Figure 11 shows an example from the test program. The first clip plane is activated by clicking on the toggle button in the top left corner. After it is activated, we can look inside the cube and see the sphere. By dragging the slider bar for the clipping plane, the clipping plane is moved closer or further away from the eye. The direction of the normal of the clipping plane can also be changed by editing the numbers in the ‘Normal’ box.

B.3 Setting the view characteristics

The view characteristics dialog for each graphics window is opened from the ‘View’ pull-down menu on the menu bar in the graphics window. Figure 12 shows an example from the test program. NOTE: When entering numerical values into a text box, it is necessary to hit <RETURN> before the changes take effect.

Here follows a brief description of the functionality in this window:

Background colour: Select a background colour from the menu by clicking on the label with the current colour. In this example, the background colour is white. Changing the background colour will take effect immediately.

Text colour: Select a text (foreground) colour from the menu by clicking on the label with the current colour. In this example, the text colour is steel blue. Note that the text colour is used to colour the axes, the labels, and sometimes also the grid lines. However, only the axes will change colour immediately after a new colour is chosen. To update the colour of the labels and the grid lines, it is necessary to replot the object on the screen.

Axes origin: Click on the radio buttons to set the origin of the coordinate axes either at the default location (lower, left, back corner of the bounding box), or at the rotation point.

Rotation point: Enter the (X, Y, Z) coordinates of the rotation point. The rotation point will remain fixed to the screen during both interactive rotation with <SHIFT>+middle mouse button, and during rotation with the buttons $\mathbf{x+r}$, $\mathbf{y+r}$, etc.

Pick rotation point: After clicking on this button, set the rotation point by clicking with the left mouse button on a point on an object in the graphics window. NOTE:

1. Picking will only work in the window from which the view characteristics dialog was opened. If you are unsure which window to pick in, you can press the right mouse button and read the window number from the title of the popup menu.

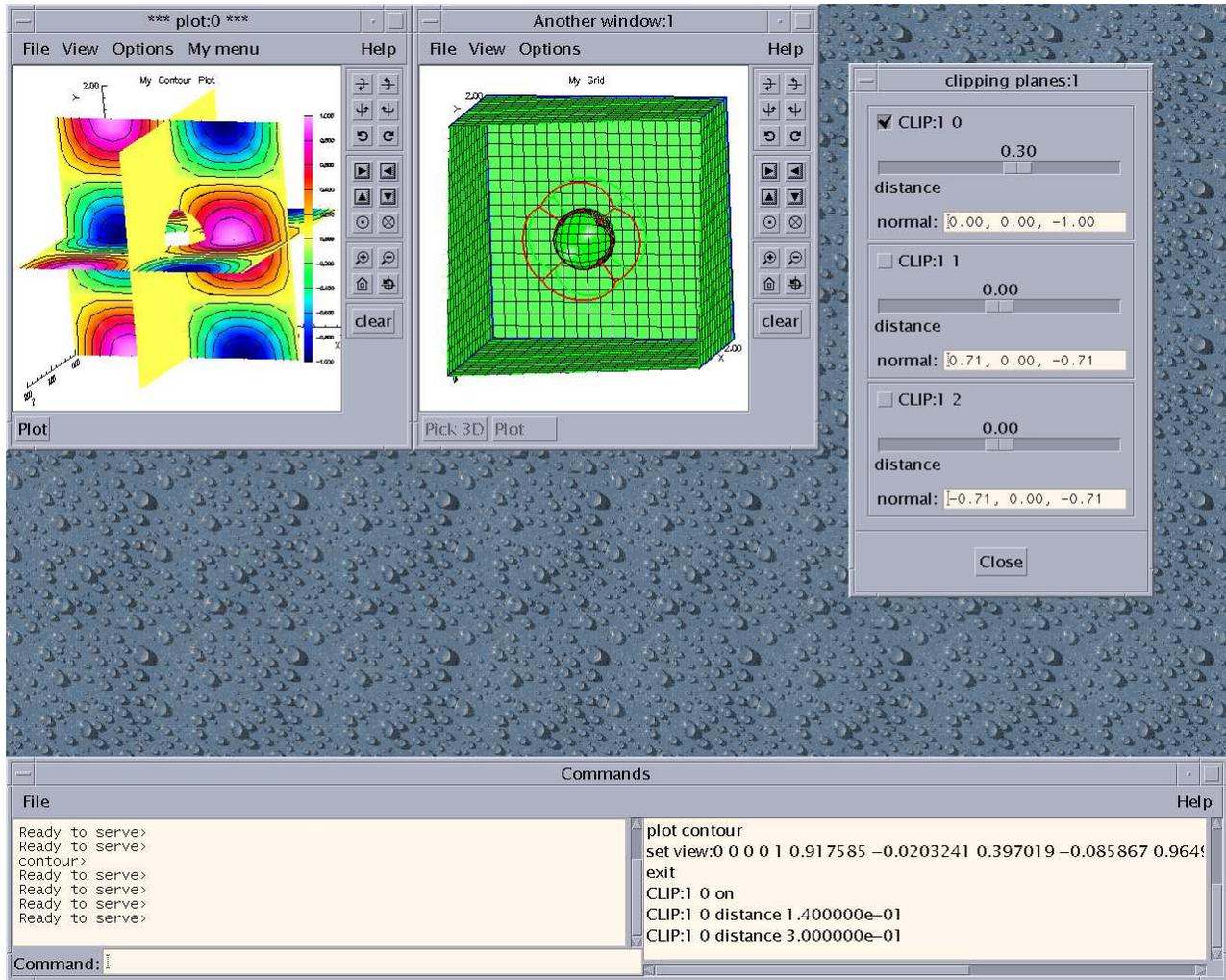


Figure 11: The Overture graphics interface. The clipping plane dialog window is also shown

Lighting: Activate/deactivate lighting in the graphics window.

Light #i: Turn on/off light source number i , $i = 0, 1, 2$. Turning off all light sources is the same as deactivating lighting with the above function.

Position (X, Y, Z): The location (X, Y, Z) of light source number i .

Ambient (R, G, B, A): The ambient colour (R, G, B, A) of light source number i .

Diffusive (R, G, B, A): The diffusive colour (R, G, B, A) of light source number i .

Specular (R, G, B, A): The specular colour (R, G, B, A) of light source number i .

X-colour material properties: The following properties characterize the default material, which is used when a X-colour is chosen for a lit object. The X-colours are distinguished from the predefined “special” materials in that only their ambient and diffuse reflections are defined, but not their specular and shininess properties. (For those fluent in OpenGL, this functionality is obtained by calling the function `glColorMaterial` with the argument `GL_AMBIENT_AND_DIFFUSE`.) Note that the reflective properties only influence objects that are lit. Also note that the objects need to be re-plotted in order for the changes to take effect.

The predefined materials are listed in table 2. Hence, any colour that is not in the table is considered to be an X-colour.

emerald	jade	obsidian	pearl
ruby	turquoise	brass	bronze
chrome	copper	gold	silver
blackPlastic	cyanPlastic	greenPlastic	redPlastic
whitePlastic	yellowPlastic	blackRubber	cyanRubber
greenRubber	redRubber	whiteRubber	yellowRubber

Table 2: Predefined materials.

Specular (R, G, B, A): The specular reflective property of the X-colour material. For example, by setting the first number to zero, you will get a bluish reflection on the green sphere in the example application. Note that the same effect could have been obtained by changing the specular colour of the light sources.

Shininess exponent: A number between 0 and 128 that describes how “narrow” the specular reflection of the X-colour material will be. A lower number gives a wider reflection.

B.4 Changing the default appearance of the windows

Default settings for some parameters can be changed through a file named `.overture.rc` in your HOME directory. An example of an `.overture.rc` file is

```
commandwindow*width: 800
commandwindow*height: 150
graphicswindow*width: 650
graphicswindow*height: 500
backgroundcolour: mediumgoldenrod
foregroundcolour: steelblue
```

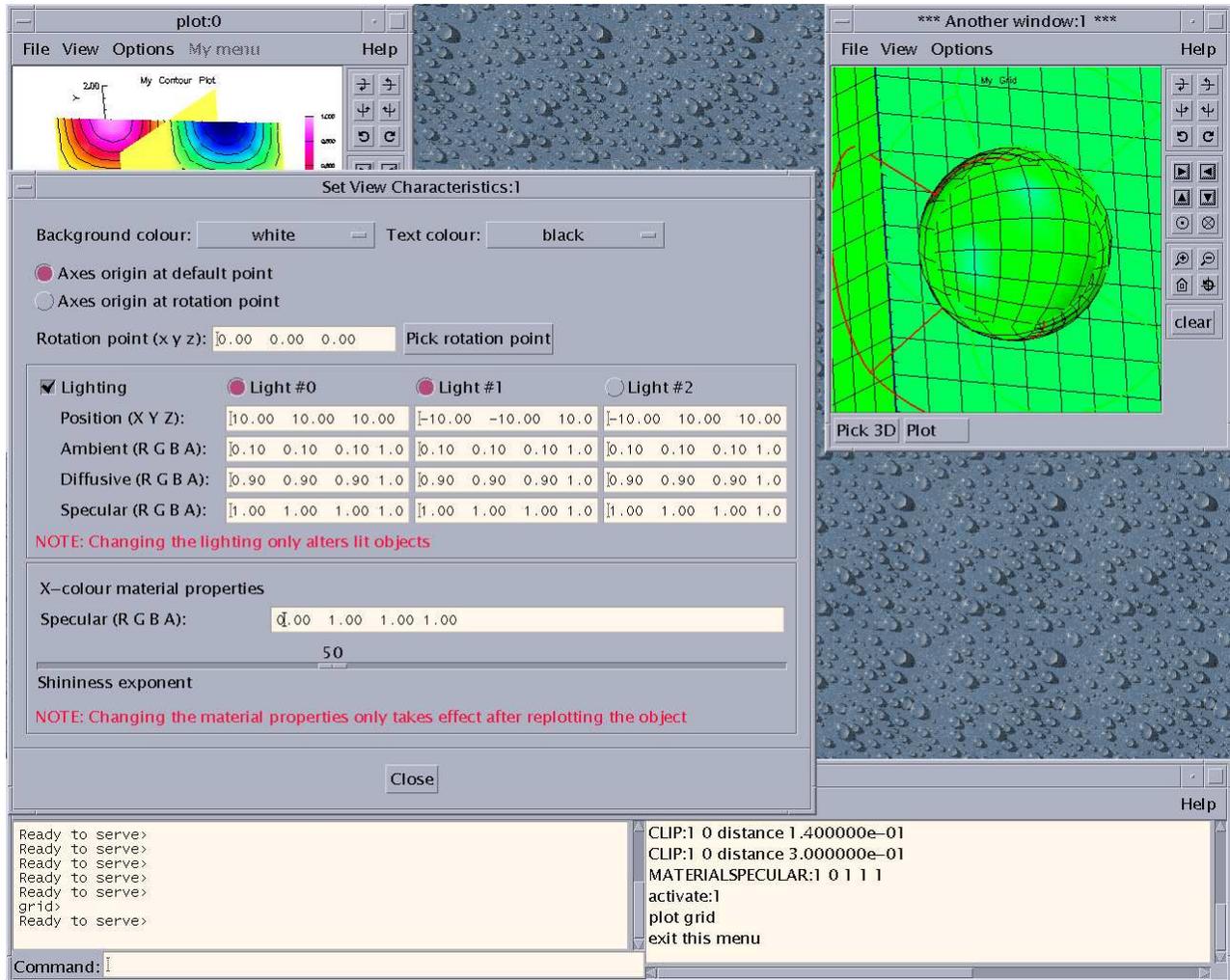


Figure 12: The view characteristics dialog window

The window sizes is specified in pixels. It is not necessary to specify both the width and height, and the default size is obtained either by omitting the command completely, or by setting the size to -1. The default foreground colour is black and the default background colour is white. If you change them, you must use one of the following colours:

black	white	red	blue	green
orange	yellow	darkgreen	seagreen	skyblue
navyblue	violet	pink	turquoise	gold
coral	violetred	darkturquoise	steelblue	orchid
salmon	aquamarine	mediumgoldenrod	wheat	khaki
maroon	slateblue	darkorchid	plum	

References

- [1] W. A. WALTERS AND J. A. ZUKAS, eds., *Fundamentals of Shaped Charges*, CMC Press, 1998.

Index

bigger,smaller,clear,reset, 22

mouse button
 translate, rotate and zoom, 23

rubber band zoom, 23