Sced:
Constraint Based Scene Editing
User's Guide (v0.61)

Stephen Chenney

Basser Department Of Computer Science

1. INTRODUCTION

Sced is an editor for editing scene files for a variety of rendering programs which runs in a UNIX and X windows environment. It makes use of constraints to position and shape objects. Simple primitive objects can be created and there is full support for Constructive Solid Geometry (CSG), which allows for the definition of complex objects using set operations on simpler objects. Objects can be positioned in the scene, scaled, rotated, copied, moved etc. The current view of the scene can be dynamically changed in all its parameters. There is support for lighting and a camera in the scene. Objects may be previewed using a chosen raytracer. The program currently produces input files for POVRay, Rayshade and Genray, three raytracing programs.

2. PRELIMINARIES

2.1 Regions Of The Window

The window consists of 5 distinct regions.

On the left is a sequence of buttons - the command buttons. Most of the program's functionality is invoked via these buttons (some are menus). Menus have a small menu-like bitmap beside their label. In particular, the *edit* menu button appears below all the others. This menu will become sensitive only when there are objects on the list for editing.

Most apparent is the *view window* in which the scene appears. If the current program window is too small, the view window will be contained within standard X Athena scrollbars.

In the bottom left corner is the *apply* button, which is used for various purposes, most often to apply the text shown in the *text entry window* or to complete an operation.

The *prompt label* is beside the apply button at the bottom of the screen, and generally gives some indication of what is currently displayed in the *text entry window*.

Finally, the *text entry window* runs across the bottom of the screen. It is used to obtain optional text where appropriate. It is a standard Athena text widget, with all the editing functions that go along with that. Note, however, that the return key has been mapped to the apply button, so hitting return while in this window is the same as clicking on the apply button.

2.2 Command Button Functions

The following functions are available (listed according to the button they are on.)

- **File:** Load, Merge, Save, Export, Copyright, Quit
- **New Object**
- **CSG Window**
- **Object:** Edit, Copy, Delete, Name, Attribs, Dense Wire
- **Lights:** Create, Ambient
- **View:** Viewpoint, Pan, Lookat, Lookup, Distance, Eye
- **Window:** Zoom, Image Size, Save, Recall, Delete
- **Layers:** New, Add Objects, Merge, Display

- **Target**
- **Camera**
- **Preview**
- **Reset**
- **Maintain**

2.3 Selection

There are 2 forms of selection - object selection and point selection. Object selection is used to choose objects for editing etc. Point selection is used when a specific points are needed, such as for the entry of object reference points.

2.3.1 Object Selection To select an object, drag over any of its edges while holding down button 1. To deselect, use button 2. A single mouse click is equivalent to dragging out a small rectangle centred on the pointer.

Selected objects are shown highlighted. They will remain selected until they are deselected or they become invisible for some reason (such as through deletion or a layer display change). Objects are generally NOT deselected by any of the functions which manipulate them.

2.3.2 Point Selection When the program needs a point to be entered, a small red square will appear on the nearest available point to the mouse. A mouse click in this square will select that point. Selected points appear in blue. If there are no more points required, the scene will return to its previous state. Selecting a point a second time will deselect it. Selection doesn't happen until the button is released, so can be cancelled by moving off the highlighted point.

It matters which button you use to select a point. Button 1 will select the point by *reference*, button 2 will select as an *offset* and button 3 will select as an *absolute* point. Currently which button you use will only make a difference when selecting points for constraints, hence the full implications will be discussed in terms of the editing interface. Suffice to say here that button 1 will do for most purposes. If a point refuses to be selected it may be that the button you are using has been masked out. Try another button in this case. Points will also refuse to be selected if they fail to sufficiently describe the feature required. Points may be deselected by choosing the point again. All this will be explained in more detail in terms of editing.

If it is valid to do so, a prompt will also appear in the text entry box at the bottom of the screen. You can type in a 3-vector here which nominates the corresponding point in the world. A vector is a sequence of three real numbers in pretty much any useful number format (those accepted by sscanf). When you have typed in the vector (eg *1.0 -2.3 5.0*), either hit the return key or click in the **apply** button to enter the text. If an invalid sequence was entered the text will be ignored. If it is appropriate for the point to be an offset point, a dialog asking to select the point type to use. If offset points are invalid in this context, the point is selected as an *absolute* point.

2.4 A Word on Text Entry

Some dialogs requiring text entry have the return key mapped to the **Done** or **Apply** button, so that a mouse click is not required. In general, such dialogs only require the entry of one piece of text. One example is the Ambient light dialog. A beep will sound if the return key is used at an inappropriate time for other dialogs. The text entry window at the bottom has the return key mapped to the apply button.

2.5 Changed Scene

The scene is considered to have changed if any of the following happens:

— An object is edited, ie it is moved, scaled, rotated or its attributes changed.

— A light is created or edited.

— The camera is redefined.

— A CSG object is defined, deleted or modified.

— A CSG tree is modified in any way.

— Another file is merged.

If the scene has changed, certain functions will ask for confirmation of some sort, generally whether you wish to save the current scene first.

2.6 Cancel

Cancel buttons are provided for almost all dialogs. Choosing cancel will exit the dialog leaving the scene as unchanged as possible.

3. BASIC FUNCTIONS

3.1 File Functions

3.1.1 Load If the scene has changed, you will be prompted to save first. Choose save to go to the save dialog, or load to get the load box.

In any case, the load dialog will eventually appear. The file name may be entered at the top, or can be selected from those shown in the windows below. The windows show 3 directory's files, each directory is a subdirectory to it's immediate left neighbour. To select a file, click on the filename. To view another directory, click on the directory name (shown with a / following its name). If there are more files than will fit in the window, a scrollbar is available. Click on Load or hit *return* to load the file shown at the top.

Any existing objects will be destroyed upon loading the file, as will the lights (as if a Reset was done). Base objects are NOT destroyed, nor are cameras.

3.1.2 Merge Merge adds the contents of a file to the scene currently being edited, without changing any existing features. It is most useful for loading predefined CSG objects for use in the current scene.

The dialog for selecting a file to merge is identical to that for load.

3.1.3 Save A save dialog, similar to that for load, will appear. See load above for details. Enter a filename at the top, or select a file. You **will not** be warned about overwriting an existing file. Choose Save or hit *return* to save to the named file.

The scene is saved in a text format. Cameras, viewports, CSG trees and the like are all saved.

3.1.4 Export If no target raytracer has been selected, you will be asked to select one. A dialog box, as for load and save, will then appear. The procedure is identical. The file produced is suitable for use as an input file to the specified raytracer.

3.1.5 Copyright A copyright message is displayed.

3.1.6 Quit If the scene is unsaved you will be prompted, otherwise the program exits quietly.

3.2 New Object

A dialog is presented for selecting a new primitive object. The objects available are Cube, Sphere, Cylinder, Cone, Square and Plane. To select on object click on its picture.

Selecting the **CSG Object** button within the New Object dialog will cause another dialog to be displayed, this time showing all the CSG objects currently defined. Use the scrollbars to view the range of objects available. Note that the CSG Object button will be insensitive if there are no CSG objects defined.

The new object is placed at the origin of the world. It is also added to the list of objects awaiting editing. It will have a name somehow indicative of its type and the number of objects created so far. It's has no attributes by default.

Note that Plane objects have a small triangle sticking out of one side of the plane. This indicates the non-normal side of the plane. That is, the normal to the plane points away from the tringle. In POV, the triangle indicates which side of the plane will be filled.

3.3 CSG Window

The CSG Window is brought to the front (and created if necessary) and all the selected objects are transferred to that window for use there. Any objects removed in this way will be taken off any edit list they may be on. Edit constraint references will be maintained between objects that are selected, otherwise they will converted to offsets or absolutes. This prevents dependencies between objects in different windows, and ensures that CSG defining objects only have references within themselves.

For a complete description of the CSG object interface, see the later section *The CSG Interface.*

3.4 Object Functions

3.4.1 Edit This command places all the currently selected objects onto the list of objects waiting to be edited. Editing in this context refers to the movement, scaling and rotating of an object.

3.4.2 Copy A copy is made of each object selected, and the copy is placed on the list of objects ready for editing. The new object has a name prefixed by the object it was copied from. The new object will inherit everything from its parent, including the constraints. Note that it inherits shape and location, so will appear directly on top of its source.

3.4.3 Delete You are prompted before all the selected objects are deleted.

3.4.4 Name For each of the selected objects, you are prompted for a new name. The current name is provided as a default. Names are for your benefit only. They are used to refer to objects on the edit list, and are printed as comments in exported files, allowing you to find the actual instance in the raytrace input file. The program places no restrictions on uniqueness or any other aspect of names.

3.4.5 Attributes (Attribs) A dialog box for object attributes is presented. The defaults are those for the first object found on the list of selected objects. When completed, the attributes of all the objects on the list are changed. This is useful for changing the surface features of a large number of objects at once. The attributes available are:

— *Colour:* Enter normalised values ($0 \leq x \leq 1.0$) for red, green and blue components of the object's colour. This is the colour before any other effect is applied, ie all other surface effects modify this colour.

— *Diffuse Reflection:* The diffuse component of an object's final colour models how much light from a source it radiates back in all directions. A value between 0 and 1 should be entered here.

— *Specular Coefficient and Power:* These parameters are used to model the highlights that appear on an object's surface due to reflection of a light source. The coefficient determines how much of the incident light is reflected in this manner. The power determines how large the highlight will be - the smaller the power the larger but less bright the highlight. The coefficient should be a normalised value, the power can be anything positive.

— *Reflection Coefficient* This parameter controls how much light from other objects is reflected by object. It is reflection in the sense of mirror reflection. Enter a normalised value, but you should be aware that reflective objects take much more time to render.

— *Refractive coefficient:* This is the coefficient of refraction for transparent objects. For instance air is 1.0, glass is 1.5. This parameter will only have an effect if the object has non-zero transparency.

— *Transparency:* This controls how much light passes through the object. A value of 1.0 means that the object is totally transparent, 0 means opaque. The light passing through is refracted according to the refractive index of the object. Like reflection, transparency dramatically increases the rendering time for an object.

It is also possible to set all the attributes to the default, by clicking on the **Default** button.

The **None** toggle at the bottom controls whether or not atrributes will be exported with the object. That is, to actually apply the attributes to the selected bodies this toggle must be unset (**not** inverted). The None button is needed to control application of attributes to CSG objects. In POVray and Rayshade, attributes specified for individual objects within CSG trees take precedence over those specified when an object is

instantiated. So for a chair, you want the attributes of indivual components to be unset (None), so that you have control over the color at the top level. For a window, you want the glass to have its attributes set, so that regardless of what you specify for the color of the window as a whole, the glass stays clear. To achieve this effect, set the glass attributes to what you wish, and make sure that all the other parts of the window (the frame etc) have None for their attributes.

When all the attributes have been set, click on the **Done** button. The program attempts to parse all the values entered and set the attributes. If a given value could not be parsed the value of that particular attribute will remain unchanged.

3.4.6 Dense Wireframes The ruled objects (cylinders and cones) have the option for dense wireframes, which use twice as many generating lines, giving a much better approximation to the true surface. Use this command to convert selected objects to this dense wireframe form. Once converted, objects cannot be taken back because of constraint references.

3.5 Light Manipulation Functions

3.5.1 Create A new light is created at the origin, and placed on the editing list. Lights are edited just like any other object. The intensity of a light is controlled through its color attribute. Lights appear in the scene as golden circles with crosses through them. All light sources are point sources at this stage.

3.5.2 Ambient You are prompted for the ambient light level in the scene. The light colour should be entered as normalised r-g-b values. This light source will diffusely light all object in the scene, regardless of whether or not they are in shadow. Note that POVRay is unable to make full use of this feature, as it only allows for monochromatic ambient light. If only one value is entered, the program will assume it is an intensity value, and set each of the r-g-b values to that entered, giving white light.

3.6 View Manipulation Functions

A range of functions are provided for manipulating your view of the scene. There are several key viewing parameters:

— *Viewpoint:* is a vector from the notional centre of the world out to the eye. In words it controls which direction you are looking from.

— *Look At:* is a point in the world which is notionally the centre of the world for viewing purposes. This point will always appear in the centre of the viewing window.

— *Look Up:* is a vector defining which way is up. A line, parallel to this vector passing through the *Look At* point will always appear to be vertical on the screen, although it may be leaning into or out of the screen.

— *Distance:* refers to the distance from the *Look At* point to the abstract viewplane window you are looking through. Points lying behind the viewplane will have strange things happen to them, and most likely become invisible. This distance also influences the apparent size of the scene.

— *Eye Distance:* is the distance from the viewplane to the eye. This controls the amount of perspective in the scene - shorter eye distances relative to viewplane distances give more perspective. For parallel views, set this value to be something large relative to the viewplane distance. Eye distance also influences the apparent size of the scene - shorter eye distances make the scene appear smaller.

3.6.1 Viewpoint The program goes into change view mode, where most buttons are desensitized and the cursor is a cross. The cursor is theoretically lying on a sphere, centred on the *Look At* point with poles in the *Look Up* direction. The current viewpoint is a point on this sphere. Moving the mouse moves the viewpoint on the sphere in the following way:

— *Button 1 Motion:* The viewpoint is free to move to any point on the sphere. Moving the mouse up or down rolls the world in the corresponding way, while moving sideways rotates the world around the *Look Up* vector. Moving the mouse all the way to the top of the window will give the impression of looking straight up the *Look Up* vector, while moving all the way to the bottom will have you looking straight down it. Moving from one side of the window to the other will do a complete circuit about the world. I recommend you ignore this description and learn by using the interface.

— *Button 2 Motion* Moving the mouse with button 2 down only allows for circling the world, it prevents moving the view higher or lower in the the scene.

— *Button 3 Motion* This allows only for apparent motion up or down in the scene.

The current viewpoint is shown in the text entry window. You can specify a particular viewpoint (in vector format) here. If for some reason your text is unacceptable, the view will remain the same.

Click on the **Apply** button to complete a viewpoint changing session. Clicking on **Undo** in the edit interface will also complete the view change.

3.6.2 Pan This is like panning a film camera. It rotates the view around the current eye position, so the scene appears to slide past you. In other words, both the viewpoint and look at point are changing while the eye stays in the same position in the world. The same buttons control constrained motion as with a viewpoint change.

3.6.3 Look At When this item is selected the program goes into *point selection* mode, where you can enter a new point for the centre of the world. You can type in an arbitrary point if you wish. If the point entered in the text entry window is an offset point, the Look At point will be offset from its current location by the amount entered.

3.6.4 Look Up The program goes into *point selection* mode waiting for 2 points defining the new up direction. The *Look Up* vector will be the one *from* the first point entered *to* the second point. Each point can be arbitrarily entered. The vector just chosen may not necessarily end up being vertical, due to perspective effects, but a vector parallel in the world, passing through the *Look At* point, will be.

3.6.5 Distance As with viewpoint changing, most of the buttons are desensitized. The cursor changes to an arrow. Button down mouse motion will cause the distance to change. Pushing up moves you closer to the scene, while dragging down moves you further away. Each button has a different speed - button 1 is medium, button 2 fast and 3 slow. You can also enter a new distance in the *text entry window*. The apply button completes a session.

3.6.6 Eye Distance Similar to distance, the eye distance is adjusted by pushing or dragging the mouse. Note that moving the eye closer will appear to make the scene smaller due to perspective effects. Again you may enter a value and the apply button completes a session.

3.7 Window Manipulation Functions

Window functions act with the View functions to control the current view of the scene.

— *Zoom:* controls the magnification of the scene.

— *Image Size:* controls the size of the window in which the scene appears. This is the window contained within scrollbars. The command does not adjust the outer window size. Note that it is the image size values that are sent to the raytracer to control the rendered size of the image (although POV ignores this option). It is also the size used in camera calculations.

— *Save:* Saves the current viewport for later recall.

— *Recall:* Recalls a viewport.

— *Delete:* Deletes a saved viewport.

3.7.1 Zoom (Magnification) Function A dialog appears asking for the new zoom value. This controls the apparent size of the image on screen. Clicking on the *Apply* button applies the new zoom without closing the dialog, whereas *Done* applies the value and closes the dialog. *To Fit* causes the zoom value to adjust to the largest value such that all the objects are visible in the scene. This is mostly useful for finding "lost" objects or expanding the scene to fill the available space. The return key is mapped to the Done button.

3.7.2 Image Size A dialog box appears asking for new width and height values for the *view window*. Enter positive integers for each value, and choose *Done.* Note that if the image size you enter is smaller than the available area in the program window, you will get a window larger than you asked for. However when the file is exported the value will be as you defined it. If you want to see how big it actually is, resize the

program window until scrollbars appear around the view window.

The *To Fit* button sets the values to the current visible area, so there will be no scrollbars.

Cancel exits the dialog leaving the actual values unchanged, regardless of whether any of the values in the dialog have been modified.

3.7.3 Save You are prompted for a name for the current view, which is then saved to be available via the Recall menu item. All aspects of the view are saved, including image size and zoom.

3.7.4 Recall A list of currently defined views is popped up. Choose the view you want by clicking on it. The stored view will be reinstated. Those views that appear in the defaults file will appear, as will any you have saved. One special view, called *Camera* sets the view to match the currently defined camera, if it exists.

3.7.5 Delete A dialog appears with all the currently defied views. Choose the one you wish to delete and it will be removed from the list.

3.8 Layer Functions

The Layer functions provide control over which objects are visible on the screen. Conceptually, each object belongs to a layer. New objects belong to the *World* layer, unless they were copied or they have explicitly been added to another layer. At any given time, a set of layers is displayed, and those objects which appear in the displayed layers are visible. Objects in layers not displayed are not visible. Invisible objects are still exported and saved. An object can only belong to one layer at a time.

3.8.1 New A new layer is created containing the selected objects. You are prompted for a name (a default is provided). The objects are removed from their previous layer, and the new layer is displayed.

3.8.2 Add Objects You are asked to select a layer, to which the selected objects will be added. The objects just transferred take on the visibility of the destination layer.

3.8.3 Merge You must select two layers. The first chosen is deleted and all the objects from that layer added to the second selected. If the *World* layer is one chosen, the other will be deleted regardless of the order selected. The objects transferred take on the visibility of the destination layer.

3.8.4 Display A popup containing all the defined layers is presented. Those displayed are shown highlighted. Each label is a toggle controlling the display of that layer. When a layer is toggled, the scene is redraw immediately to reflect the change. Choose *Finish* when you are satisfied.

3.9 Target

This button allows for the choice of the target raytracer. This not only determines the format of the exported file, it also changes the dialog presented for camera definition.

3.10 Camera

A dialog for the entry of camera data is presented. The actual parameters prompted for is dependent on the target raytracer, and if no target has been selected it is prompted for first.

You can modify any parameter - each corresponds to a camera definition element for the target raytracer.

Alternatively, you can use the *To Viewport* button to match the camera to the current viewing specifications. This is the easiest way to define a camera.

A *Default* button is also provided, although the default settings for each of the raytracers are useless for anything other than the simplest scene.

When all is finished, choose the *Done* button, which will save the camera as defined. *Cancel* will close the dialog leaving the camera unchanged from its previous state.

3.11 Preview

You are asked to nominate a raytracer with which to preview the selected objects. A window matching the size of the current view window is used, with the current view specifications (NOT the camera). If no lights

have been defined, a light located near the eye is used, otherwise the defined lights are used.

The selected objects are exported to a temporary file then previewed using a raytracer chosen for the task. If the process fork succeeded a dialog will pop up indicating the file that the raytrace-r's output is going to, and asking for recognition.

The target raytracer is not necessarily used because it may be unsuited to previewing (Rayshade does not give a runtime view of the picture being traced and POVRay is very slow but gives a dynamic color view.) The full pathnames for each of the raytracers needs to be defined in the defaults file, or compiled in. It is also possible to specify options for the preview.

3.12 Reset

As the name suggests this command destroys (after confirmation) the scene. All objects, uncompleted CSG trees, lights and the camera are deleted and the screen cleared.

3.13 Maintain

The Maintain button is a toggle. It controls interactive constraint maintenance. If on, constraints will be interactively maintained as objects are dragged and edited. If off, constraints will only be maintained at the end of each object edit session. On is the better setting from almost all points of view, except refresh speed. Turn maintenance off if the number of dependent objects is large and updating is much slower than usual.

4. EDIT INTERFACE

4.1 Overview

Sced is a *Constraint Based* editor, in that objects are manipulated through constraints on their location, shape and orientation. Constraints are also used to remove the 3rd dimension from the world, to allow meaningful interpretation of a 2D mouse point as a 3D world point. Constraints may also be *maintained* which means that the relative position of objects may remain the same even when one of the objects is moved.

Sced works by constraining 3 features of each body:

— *The Origin* controls the *position* of the object. The Origin appears in Green somewhere in the scene, initially at the centre of the object. Moving the origin point moves the body with it, and constraining it constrains where the body is in space. The origin also forms 2 other key functions. It the the centre for all rotation of the body. That is, the origin will stay in the same spot as the body rotates around it. This lets you rotate about any point you like, which is very useful. The origin is also the centre for scaling the object. That is, the origin stays where it is, regardless how you scale the object. This means you can move the object while scaling, or scale about one corner of the object. The upshot of all this is that the origin stays where you put it, no matter what other operations you apply.

— *Axes* control the orientation or *alignment* of the object. They appear as red, green and blue lines sticking out of the origin. The red, or *Major* axis is longer than the green, or *Minor* axis. The blue, *Other* axis is the shortest. The Major axis always takes precedence over the Minor axis, and the Other doesn't control anything much. Scaling is along the axes. When the axes rotate, the body rotates with them.

— *The Scaling Point* controls *scaling* of the object. It appears in Red on one vertex of the body. When this point moves the object scales to keep the point at the same vertex. Scaling is about the Origin and along the Axes, so you can shear the body if you want.

The cursor will change depending on what feature the mouse is over. If the feature under the mouse is not open to interaction, the cursor will be a closed circle. The cursor will be a pair of arrows in the rotation region. It will be a diamond cross in the origin region and a sizing arrow in the scale point.

The features just described may be constrained in various ways. The constraint types available follow. The number in brackets is the number of points needed to define the constraint.

— *Plane (3):* The point is constrained to lie in the given plane, defined by 3 points on the plane. The plane is represented by a large rectangle with a cross through it.

— *Line (2):* The point lies on a line as defined by 2 other points on the line. A line is represented by a line segment.

— *Point (1):* Completely defines the location of a point. A circle is drawn around the point.

— *Midplane (2):* The point must lie on a plane equidistant from 2 points. This is effectively a plane constraint, but with a different method of specifying the plane, and is represented as a plane.

— *Midpoint (2):* The constraint defined is a point — the midpoint of the two selected.

— *Origin Line or Plane (1,2):* The constraint that results is a line or plane which passes through the current origin point and the point(s) selected. The line or plane will follow the origin around as it changes.

— *Scale Line or Plane (1,2):* The constraint is a line or plane which always passes through the current scaling point and the points selected.

— *Axis (2):* An axis for rotating about, defined to be parallel to a line joining 2 given points. It appears as an arc on the arcball sphere. The arc could be considered as the edge of a disc with the defined axis.

— *Axis Plane (3):* A plane whose normal is used as a rotation. It essentially defines an Axis in a different way, and is drawn as an axis.

— *Alignment Point (1):* A line from the current origin to another point which the constrained axis will point along.

— *Alignment Line (2):* A line joining two points which an axis is constrained to be parallel to.

— *Alignment Plane (3):* A plane which an axis is constrained to be perpendicular to.

Constraints are defined by choosing the type off a menu and then selecting points which define the constraint. To select a plane, choose 3 points you want the plane to pass through. To choose a line, select 2 points that the line will pass through.

The type of point used to define a constraint matters.

Reference Points (button 1) are attached to a particular object. As the object they are attached to moves, so does the defining point. Use this type of point to specify relationships between objects that you want maintained. When choosing a reference point, button 1 down will cause the object referenced to change colour. If this is not the object desired, move the mouse with the button down to cycle through other objects which may be referenced by that point. Button up inside the point rectangle chooses the currently highlighted object.

Offset Points (button 2) stay in the same place relative to the current object. Use these to specify scaling constraints which relate to the body being edited only. If offset points are entered via the text entry window, the vector entered is taken as the offset from the centre of the body. Offset points are also useful as origin line constraint specifiers, as they encode motion in a particular direction.

Absolute Points have a fixed position in space. Use them to position an object somewhere where you don't want it to move from.

Reference points are the basis for constraint maintenance. They allow the current object to react to changes in the object it references. There are limitations on what can be used as a reference point. In particular, reference points cannot lead to cycles. The system will not allow references to objects that depend on the current object. So you can't say object A is the same height as object B, and then say that object B is the same height as A. The system will refuse to select reference points from dependent objects, which are shown dashed during editing.

The Origin and Scaling points both have a set of available constraints associated with them. To actually constrain the point, choose one or more of the available constraints. The selected constraints are said to be active. More than one constraint can be active at any one time, and the system solves for a resultant, which is like having all the selected constraints active at the same time. Interactive rotation also has a set of available constraints, but it is only meaningful to have one selected at a time. The Major and Minor axis can

each have a constraint associated with them, but this constraint is chosen explicitly. New constraints are added to the available lists, but not made active.

Constraints are named for your convenience. When a new constraint is added you are prompted for a name. A default is provided. Once again names need not be unique or anything like that (but you may have problems telling them apart).

The whole system works by doing the following:

— Make the origin satisfy the origin constraints. This will position the object somewhere.

— Make the axes satisfy any alignment constraints. This will orient the object somehow.

— Make the scaling object satisfy any scaling constraints. This will scale the object.

If a point is effectively Plane or Line constrained it may be moved with the mouse in such a way that the constraint is maintained. To move a partly constrained point, push any button down on the point and drag it to where you want it to go. If you drag the Origin point the object will move with the mouse. It may also be scaled to maintain the scaling constraints. If you drag the Scaling point the object will be scaled, and the origin will stay where it is.

If constraints are chosen for a point that cannot be mutually satisfied, the special Inconsistent constraint results, and the point is fixed until the constraints are adjusted. An inconsistently constrained point has a cross drawn through it.

4.2 Position

It is the Origin that defines the position of the object. The origin may be moved in 2 ways:

— If partially constrained it may be dragged around with the mouse by clicking any button down on the point and dragging it to its new location.

— A constraint may be selected which, by forcing satisfaction, moves the origin to a new location. This is actually a more definite way of doing things, because it allows for precise positioning. The object will move by the **minimum** amount to satisfy the new resulting constraint. This means the point will move perpendicular to the constraint until it hits it.

If the Origin is Point- or Inconsistently-Constrained, or not constrained at all, it cannot be moved interactively. It may still move if something it depends on is moved, but more on that later.

4.3 Scaling

The Scaling Point defines the scaling of an object. It may be moved in 2 ways just as for the Origin.

An object can also be scaled in response to a Position of Rotation drag, if the drag results in the Scaling constraints being violated. This may seem dangerous but is in fact very useful behaviour which allows a particular point to be put somewhere and then kept there.

All scaling happens with respect to the Origin and the Axes. So scaling will never cause the Origin to move, and is always relative to the Axes. If you don’t change the axes this means that scaling is always along the object’s natural axes.

Scaling uses the ratio between the original distance to the origin and the new distance to determine how much the object should be scaled in each direction. If the original distance to the origin along any axis is 0, then a constraint is automatically enforced to stop you scaling in that direction. These are called forced constraints.

4.4 Rotation

Interactive rotation happens through a mouse drag. If the button goes down anywhere in the scene that isn’t the Origin or Scaling Point, it is assumed rotation is taking place. If a rotation constraint is active then the object will rotate about this axis and around the Origin. The interface is Shoemake’s[1] Arcball interface. On the screen there is a dashed circle centred on the Origin point. This defines the edge of an imaginary sphere centred in the Origin and with the radius indicated. When you move the mouse you are notionally dragging it around on this sphere, and the object moves underneath. The interface works best if the motion is

constrained, when you should try to drag somewhere near the indicated arc. Dragging around the outside of the dashed circle is the same as rotating about an axis out of the screen, which may be useful. As rotation interfaces go this is good, so take the time to play with it. For full details, implementation and all, see the paper or the Graphics Gems 4 book.

The scaling constraints are maintained at all times through a rotation, which can give some interesting results, particularly if the scaling constraint suddenly goes from inconsistent to satisfiable. Undo will always get you back.

4.5 Alignment

Alignment involves rotating the object so that one of the axes satisfies some constraint. To align an axis, choose the type of constraint from the Major or Minor axis menus, and define the points for the constraint. The constraint will remain active until you remove it by selecting remove from the same menu. Axes can be temporarily aligned with one of the interactive rotation axes by choosing Temp Align off the axis menu.

Alignment constraints will cause corresponding interactive rotation constraints to be enforced to prevent breaking the constraints. The constraints will be un-enforced when the alignment is removed.

4.6 Default Constraints

Three default constraints are provided for each of Position, Scale and Rotate. They are:

— *Position:* Three planes aligned with the world axis planes and passing through the Origin. The X–Y Plane is the plane defined by the world X and Y axes. The others are the Y–Z and Z–X planes.

— *Scale:* Three planes passing through the Reference Point and perpendicular to the Object Axes. Together they allow rectangular scaling of the object. The planes are named after the axes that lie in them.

— *Rotate:* Three axes aligned with the Object Axes.

The defaults all move and change with the object in order to maintain the conditions just described.

4.7 Controlling the Direction of Motion

If you wish to control the direction in which a point will move to satisfy a constraint, you can do so through Origin or Scaling Line or Plane constraints. Consider a situation where you want a cylinder to move along its axis line until it meets a plane which is not perpendicular to the origin. It you were to just choose the plane constraint, the cylinder would move perpendicular to the plane, not its axis. To force it to move along its axis, add an Origin Line constraint passing through the Origin (it must) and parallel to the axis. Choose this constraint first. This effectively says that the origin must remain on its current axis line. Now choose the plane constraint. The system will calculate where the axis line intersects the plane and move the origin there. If you think about it this is the same as moving the cylinder along its axis until it meets the plane.

The same technique will work to control the direction of scaling, this time through use of a Scaling Line or Plane constraint.

4.8 Maintenance

The network of reference points used to define constraints introduce dependencies of one object upon another, which in turn may depend on some other object.

Constraint dependencies are maintained whenever an object is manipulated, if the Maintain toggle is set, otherwise they will be adjusted when editing of the current object is finished or suspended. So if a block is defined to lie on a plane, by using the plane to define reference points for a position constraint, and the plane is subsequently moved up, the block will also be moved up to still lie on the plane. The block will move in the minimum amount to satisfy the constraint.

Dependent objects are shown dashed. These objects are the ones that may be affected by manipulating the current object. You cannot select reference points from among these objects because to do so would cause cycles.

The solving of dependencies may take a little while. All dependencies, whether active or not, are updated, so don't keep unnecessary constraints with an object. All defined constraints are stored and saved with an object, so delete the ones you don't want any more before finishing or suspending the edit object session.

If you don't want any maintenance, define all position constraints as absolutes, all scaling constraints as offsets and rotation as one or the other. Alternatively, deselect all constraints before you finish editing an object. INACTIVE CONSTRAINTS CANNOT AFFECT THE OBJECT.

When an object is updated by the constraint maintenance algorithm, its position is updated first, then its orientation and then its scaling. This is consistent with the way things happen above.

4.9 Forced Scaling Constraints

When the Origin and Scaling Point both lie in one of the 3 planes defined by the Object Axes, scaling perpendicular to the plane is impossible. In this situation the program will enforce the corresponding default scaling constraint so violation is impossible. You cannot deselect forced constraints. The only way to remove them is to change the Origin, Scaling or Axes such that the situation no longer exists. The program remembers which constraints it enforced, but if one is selected while also being forced the program will remember that you selected it, and leave it selected once the enforcement is unnecessary.

Rotation constraints are also forced if alignment constraints are active. The intention is to prevent interactive rotation from destroying alignment. The forced constraints are removed if the alignment constraints are.

4.10 The Dialog Box

When an editing session begins on an object the Edit dialog box is popped up below the window in which the editing is happening (you can move the dialog around at will, even minimise it). All the buttons in the main displays are deactivated. The dialog itself has lots of buttons, some of whose meaning should be intuitive from the previous discussion.

— *Finish:* This finishes a session. Several things are done:

- The object is removed from the edit list and put in the world list if not already there.
- All objects depending on this one have their constraints updated and reworked.
- The screen is redrawn and all the main buttons resensitized.

There is no return from a finish. An full undo really should to be provided.

— *Suspend:* This is the same as a finish, except that the object is not removed from the edit list. Use this it you intend to do more editing of the object in the near future.

— *Undo:* Undo provides undo right back to the start of the current session. Everything can be undone, including constraint addition, removal, selection and all drags of any type. In effect, undo winds the state back one step. Undo is also used to cancel point selection, cancel a view change and cancel constraint removal.

— *Redo:* Undoes an Undo.

— *Origin:* Allows the specification of a new Object Origin. The program goes into select point mode waiting for a point. You are free to select any point, or type one in. It doesn't matter what type of point you select. The object will be moved if necessary so that the new origin satisfies the Position constraints, and any scaling constraints will be updated as a result of such a move. Redefining the Origin is a good way to move an object. In the box on table example it is useful to have the origin on the bottom of the box somewhere. A change in Origin can also cause scaling constraints to be forced.

— *Scaling:* Allows a new Scaling Point to be defined. The point chosen must be a point on the body, but it doesn't matter which button you use to select it. The object will be scaled if necessary to satisfy any scaling constraints. A change in Scaling may cause scaling constraints to be forced.

— *Major:* Allows access to all functions related to the Major Axis:

- Redefine 1: You choose a single point and the axis is redefined to point toward it. The object doesn't rotate, rather the axis is moved relative to the body.
- Redefine 2: As above but you choose 2 points, and the axis is defined to be parallel to the line passing FROM the first point TO the second. Direction matters.
- Temp Align: Aligns the axis with the current active rotation axis, if one is active. It doesn't remember the constraint, hence is temporary.
- Align Pt: Align the axis so that it points toward a chosen point, This is a maintained constraint, so be careful with the point type (reference, absolute or offset).
- Align Line: Align the axis to be parallel with a line going from a first selected point toward another selected point. Again this constraint is maintained, and direction matters.
- Align Plane: Aligns the axis to be perpendicular to the given plane (you enter 3 points). This one's maintained too.
- Remove: Removes any constraints that may be active.

There is a label at the bottom indicating whether or not the axis is constrained.

- *Minor:* This does all the things outlined above but with the Minor axis. There is one other difference. Because the Major axis takes precedence, and the Minor axis must always be perpendicular to it, the rotations and alignments that result move as close as possible to the constraint you specify, but may not achieve it entirely. This may sound difficult but is actually useful behaviour.
- *Remove:* Lets you remove a constraint from one of the constraint boxes. The cursor changes to a skull and the program waits for your next move. If you select an **inactive, non-default** constraint, that constraint will be deleted. Other constraints cannot be deleted. Note that the Uniform scale constraint is not a default for the purposes of this function.
- *View* Allows for view changing without exiting the edit session. You can change the viewpoint, the lookat point or the zoom. This is a very good way to verify the position of an object or the orientation of a constraint. Use Apply or Undo to cancel the operation.
- *Position Box* All the available Position constraints are displayed here. Each is a toggle showing whether or not that particular constraint is active. To activate or deactivate a constraint simply click on the toggle. An Add menu is provided to add new position constraints to those available.
- *Scale Box* A similar concept to the Position Box. Everything works the same. Recall that some constraints will be enforced by the program at certain times, and these cannot be deselected.
- *Rotate Box* The available rotation axes are displayed here. Only one may be active at any given time. Apart from that, it's the same as the Position Box.

5. THE CSG INTERFACE

The CSG interface consists of a separate window in which CSG objects are created and edited. The window is accessed via the **CSG Window** button on the main display. The window is divided much along the same lines as the scene window, with the addition of an extra region below the View Window of a *CSG Tree Window* where the objects currently being manipulated are represented by small labelled buttons. Each button is a menu, allowing access to certain functions for that object. The available functions will be discussed later.

The relative size of the CSG Tree Window and CSG View Window can be changed by dragging the small square on the dividing line up or down. Scrollbars will appear in both windows if required.

5.1 CSG Window Functions

Many of the functions provided in the CSG window are identical to those in the Scene Window. The full list of functions available are:

- **CSG:** Modify Existing, Delete Existing, Close
- **New Object**
- **Object:** Edit, Move, Name, Attribs, Dense Wire
- **View:** Viewpoint, Lookat, Lookup, Distance, Eye
- **Window:** Zoom, View Size, Save, Recall, Delete
- **Layers:** New, Add Objects, Merge, Display

In addition, there is an Edit button as with the Scene Window.

For those functions which are also defined in the Scene Window, the behaviour is the same except that it works on the CSG View Window and objects contained therein.

5.2 Modify Existing

This function allows for the editing of an existing CSG type. For an object to be modified there must be NO instances of that object. If an object you choose to modify has instances, you may choose to edit a copy of the object. Note that instances of one object appearing in the definition of another are counted as instances.

Objects with instances cannot be edited because of constraints. In particular, the wireframe is likely to change, which ruins every reference constraint because the referenced vertex now means something else. There is also the issue of what to do with the instances and their own constraints.

A dialog is popped up (it's the same as the new object one) allowing the selection of a CSG object for editing. A warning will be issued if the object has instances and you may edit a copy or cancel the operation. If all goes well, the CSG tree representing the object will appear in the CSG Tree Window as another CSG tree. From this point it may be manipulated just like any other tree.

This button will be unavailable if there are no CSG objects currently defined.

5.3 Delete Existing

A dialog is presented allowing the choice of an existing object to be deleted. The object to be deleted cannot have any instances, for obvious reasons. Again remember that instances may be hidden in the definition of other CSG objects.

This function will be unavailable if no CSG objects are defined.

5.4 Close

This button causes the CSG Window to be closed. This only removes the window from the screen, it does not change the contents of the window, which will be available once the window is remapped using the CSG Window button in the Scene Window.

5.5 Tree Menu Functions

Each button displayed in the CSG Tree Window is a menu displaying functions applicable to that particular node in the tree. Which functions are applicable depends on the type of node and where it is in the tree. The full list of available functions is:

- *Display*
- *Move*
- *Attach*
- *Complete*
- *Preview*
- *Evaluate*
- *Reorder*

- *Break*
- *Copy*
- *Delete*

5.6 Display

This option toggles display of the tree in the CSG View Window. The objects in a displayed tree may be edited just as normal instances. There are a few things to note however. All that really matters is the position of the object relative to other objects IN THE SAME TREE. Once a CSG object is created it can be moved about just as any other object, so its position with respect to the origin is not important for that reason. However, it does matter where the object is relative to the origin, because the origin will become the notional centre for the created CSG object. That is, it will be the default object origin for position, and the default fixed point for scaling and editing. These points may be changed later, but it is still a good idea to keep the origin somewhere near where it is sensible to think of it as the centre. Also note that the size of the object is not too important, as the resulting object may be scaled later.

Removing an object from the display takes any of its instances of all of the lists, including the selection and edit lists. In other words it is only possible to operate on displayed objects.

Displayed objects are still affected by visibility, so it they don't appear, check the visibility threshold.

The Display option is available for root nodes. All or none of a tree is displayed. To display parts, break the tree then put it back together when finished.

5.7 Move

The move option is for reordering trees within the CSG Tree Window, NOT for moving a tree's instances in the world. This option mostly exists because it is possible for some objects to be off-screen, while attachment requires that the 2 objects to be attached both be on-screen. Off-screen objects can be accessed using the scroll bars, but it may not be possible to get both trees on the screen without moving one.

Once the Move option is selected, a small rectangle appears attached to the cursor. Click with button one when the cursor is in the position that you want the tree to appear. For instance, to move the 7th tree so that it appears after the 2nd tree, select move from the 7th tree's menu, then click with button 1 somewhere between the 2nd and 3rd trees.

The Move option is available for root nodes.

5.8 Attach

This operation is the essence of CSG, allowing 2 trees to be combined with a set operator.

To attach one tree to another, select the Attach item, then choose the node from ANOTHER tree that you wish to be the SIBLING of the attached tree. The node selected as the sibling must be from a different tree - you cannot attach a tree to one of its children. The node you attach will become the RIGHT child of a new node, and the node you selected as the destination will become the LEFT child of the new node. The new node will be attached to the parent of the destination node, or become a new tree if the destination was a root node itself.

To choose the destination, push button 1 down on the desired destination node, and another menu will pop up allowing you to choose the type of operation to apply, or cancel. Choose the appropriate option and release the button.

Note that the ordering of the nodes is only really important for the difference operator. The right node is taken away from the left node. Use the Reorder option to swap the ordering if it isn't right.

Regardless of the initial display state of the tree you are attaching, it will always assume the display state of the destination tree.

This option is available for root nodes.

5.9 Complete

The Complete option takes a CSG tree and turns it into a new CSG object. If the tree selected consisted of a single instance it will simply become a normal instance in the world. Otherwise the following things happen:

- All references and dependencies to objects NOT in the current tree are replaced by absolute (for Origin) or offset points.
- You are prompted for a name for the CSG object being created. Select Complete or hit return when you are finished, or cancel at this point.
- A wireframe is generated for the object. THIS MAY TAKE A WHILE. It could be as bad as O squared seconds, where O is the number of objects in the CSG tree you are creating. (The process actually is most dependent on the number of polygons making up the objects at each level in the tree, but this is roughly proportional to the number of objects.) Note also that Planes and Squares will give unpredictable results.
- You are presented with the resulting object and asked to specify a default Scaling Point for the object.
- The object is made available as a new CSG object.
- The tree is removed from the CSG Window.

The Complete option is available for all root nodes.

Note that no instances of the object are created, you must create instances through the New Object dialog, just as with any other object.

The wireframe generation procedure takes a while because it attempts to produce a realistic wireframe for the object which is not more complex than necessary. This procedure is also highly susceptible to numerical inaccuracy. Occasionally it produces wireframes which aren't quite right.

When asked to select a default Scaling Point, the option is also available to use a full wireframe for the object. The full wireframe consists of the union of all component object wireframes. This option is useful in cases where the intersecting set of objects is so small that the approximate wireframes do not intersect at all.

5.10 Evaluate

The Evaluate option is sort of a Complete then Attach operation. The subtree starting at the node selected is completed as above, then an instance of this completed object is attached in place of the subtree. It allows for tree simplification to some extent.

When you select this option, the processes involved are the same as for complete with the extra stage at the end of creation and attachment of an instance in place of the tree.

This option is available for non-root internal nodes in the tree.

5.11 Preview

The subtree of the selected node is previewed using a renderer you choose. The preview uses the current CSG viewport and window size, so it is best to have the object displayed to make sure it's visible. This option allows indeterminacy about the resulting shape to be resolved. It goes some way toward solving the problem of design using approximate representations.

5.12 ReOrder

When this option is selected the children of the selected node will be reordered. This is useful for Difference operators, where the ordering matters (right is removed from left), or for changing the appearance of a tree in the window.

This option is available for all internal nodes.

5.13 Break

Selecting Break will cause the selected subtree to be broken off its tree and made a tree by itself. The parent node of that selected will be removed, and the node's sibling put in the parent's place. The node and all its children will become a separate tree.

The Break option is available for all non-root nodes.

5.14 Copy

The copy button copies the selected node and all its children and adds them as a new tree. It replaces the Copy command on the Object menu. The new tree is not initially displayed.

All constraints in a copied tree that reference other nodes in the tree will be adjusted to reference the corresponding copies. This allows a completely constrained tree to be copied and then edited independently.

Any node may be copied.

5.15 Delete

This will delete the selected node and all its children, patching the remaining tree as appropriate. No confirmation is asked for, so be careful. This function is essentially a replacement for Delete on the Object menu.

Any tree can be deleted.

6. COMMAND LINE OPTIONS

-F filename

Load the specified filename on startup. The -F need not be specified. Sced first looks for the file in the current directory. If it can't find it there, it tries the scene directory specified in the defaults file. If it still can't find it, it prints a warning and continues.

-D defaults

Use default values found in the given file. If this option is not present the program will look for a file called .scenerc in the user's home directory. Failing this, internal defaults will be used. The format for the defaults file is described below.

-I WidthxHeight

Use a view window of the given size. This does not effect the size of the program's window (use -geom), rather it determines the size of the scrollable window in which the scene appears.

7. DEFAULTS FILE

At startup the program attempts to read a file containing default values for various things. If the -D option was specified on the command line, it will look there, otherwise it will look for a file called *.scenerc* in your home directory. If neither can be found, compile time defaults will be used.

7.1 Format

The following keywords with the accompanying arguments are accepted. If a keyword is not present in the file, that value will have the built in default. Spaces, tabs and newlines act as separators. Anything from a # to the end of line is taken as a comment. In the following list, keywords are in bold and arguments in italics. Note that the program is case sensitive.

MainViewport

Begins the definition of the main viewing parameters. Some or all of the following keywords should appear next in the defaults file.

LookFrom *x y z*

Set the *Viewpoint* vector to given vector, where x, y, and z are floating point numbers.

LookAt *x y z*
Set the *Look At* point to be the given point.

LookUp *x y z*
Set the *Look Up* vector the the given vector.

ViewDist *dist*
Set the viewplane *Distance* to the given value, which should be a positive value.

EyeDist *dist*
Set the *Eye Distance* to the given value, which should be positive.

Magnify *zoom*
Set the *Zoom* value to the given value, which should be a positive integer.

Screen *width height*
Set the *Image Size* to the given width and height, which should be positive integers.

Viewport *"name"*
Define a viewport which will be available for recall. The parameters that follow are all the same as for the MainViewport.

Genray *path options*

POVray *path options*

Rayshade *path options*
Set the pathname and options for the named raytracer. These values will be used for the preview command, they do not influence the exported scene. If you do not have one of raytracers, set the value to "", the empty string.

Directory *pathname*
Set the default directory for loading, saving and exporting scenes. This is simply the directory that is shown in the file selection dialog box, and is not any hard restriction.

Attributes
Set the default object attributes. Any or all or the following options can appear. If a particular option is not set, the value will retain the compiled in default. See the *Attributes* command above for a full description of each parameter.

Color *red green blue*
Sets the default object color, where red, green and blue are values between 0 and 1.

Diffuse *coef*
Set the default diffuse coefficient.

Specular *coef power*
Set the default specular coefficient and power.

Reflect *coef*
Set the default reflective coefficent.

Refract *coef*
Set the default coefficient of refraction.

Transparency *coef*
Set the default transparency value.

8. X RESOURCES

There are fallback resources for all the top level window sizes. Everything else uses the local default. If you wish to change things, particularly the font, colors and geometries you may do so through a resources file, the command line or compile them in. The font has a major influence on the relative size of buttons and some dialogs. You may prefer editing using white on black, and if you have a smaller screen resolution you

will most likely wish to change the default geometries. The geometries used by default were designed for at least 1024x768, but are best at bigger than 1152x900 screen resolution.

The widgets you may like to change, and their defaults are:

Sced.geometry: 800x600 — the main window overall size.

Sced.csgShell.geometry: 800x600 — the csg edit window overall size.

Sced.newObject.geometry: 800x600 — the new object selection box size.

Sced.csgSelectShell: 400x300 — the dialog for selecting CSG objects for new object or modify/delete existing.

Sced.csgShell.csgReferenceShell: 400x400 — the "Choose a Scaling Point" dialog associated with CSG completion.

Sced*mainViewWindow — the main view window. You may wish to change the "foreground" and "background" resources for this widget.

Sced*csgmainViewWindow — the csg view window. Again, you may wish to set the "foreground" or "background" resource.

Sced*csgReferenceView — the view window used to show the "Choose a Scaling Point" CSG completed wireframe. You might like to change the foreground and background.

Sced*font — if you want to change the font.

9. RENDERER SPECIFIC NOTES

9.1 POVRay

The patches supplied with the Sced distribution will need to be applied. They allow for the matrix keyword to specify object transformations. Squares are rendered as thin boxes.

The normal to a plane is important in POV files. To determine which side of a plane is filled, a small triangle is drawn. That is, the triangle points opposite to the normal toward the filled half-space.

9.2 Rayshade

Cones and Cylinders are rendered WITH endcaps.

REFERENCES

1. Ken Shoemake
Arcball: A User Interface For Specifying Three Dimensional Orientation Using a Mouse
User Interface '92 , 1992, pp 151-156.

CONTENTS