John Mardaljevic ***e-mail*** **jm@dmu.ac.uk**
Institute of Energy and Sustainable Development (IESD), De Montfort University, Leicester, UK

How to Generate Renderings Using Arbitrary-Shaped Clipping Surfaces

Background

The clipping planes feature in *Radiance* was a neat enhancement that allowed us to peek inside closed buildings without interfering with the transfer of light in the scene (see Section 3.2.2, p110 of the *Rendering with Radiance* book). In the current implementation (v3.14p), the clipping planes are exactly that - planar, flat in other words.

There may be occasions however when the user would like to generate a view using a non-planar surface. For example, say you have a scene with an 'L' shaped tower (green) and close by another tower (red), Figure 1. You would like to generate a single rendering showing the two re-entrant vertical surfaces under standard overcast sky conditions (i.e. to predict vertical daylight factors). A good view point for this would be that used for the rendering in Figure 1 - except of course the red building is in the way. For scenes like this it may be impossible to set parameters for a clipping plane that reveals both surfaces complete. The clipping plane drawn in Figure 1 takes out the obstructing tower at the expense of clipping off some of the green building.

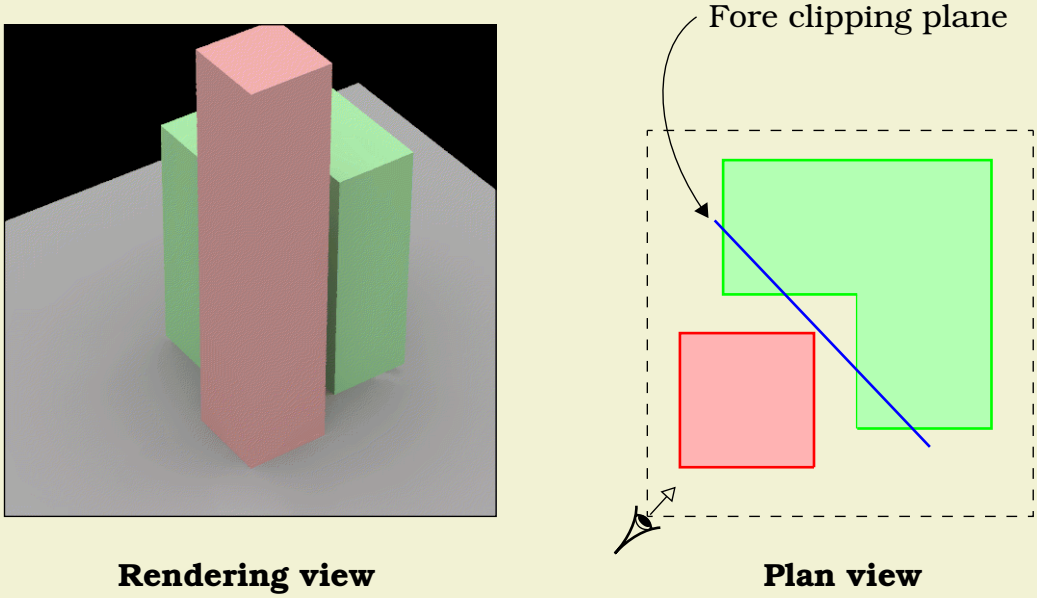

Figure 1. Scene and schematic showing clipping plane

It turns out that the user can (with relative ease) specify arbitrary-shaped surfaces to use as clipping 'planes' for tricky situations such as the one described above. The solution was worked out by Greg Ward during a brainstorming[1] session with the author and is described in the following Section.

1. Greg furnished the brain - the least I could do was to try the idea out and document it (JM).

A Neat Solution

The trick is to use a combination of **vwrays** and **rtrace** to generate the rendering *instead* of **rpict**. No 'built-in' clipping plane is used - **rtrace** doesn't have one. Instead, any surface of any shape can be made to act as a clipping 'plane' by using it to *intersect* with and then *restart* rays that originate from the desired view point, Figure 2.

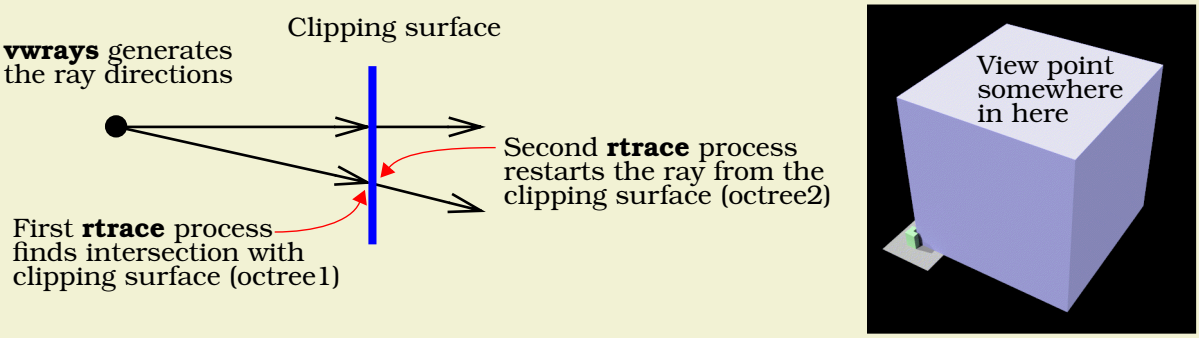

Figure 2. Schematic of pipeline process and rendering of 'clipping box'

Although it sounds a little fussy from the description given above, the individual processes can be linked together in a command pipeline as follows:

```
vwrays -fd -vf v1.vf -x 1024 -y 1024 | rtrace -w -h -fd -opd octree1 \
| rtrace [ambient options] -fdc `vwrays -d -vf v1.vf -x 1024 -y 1024` \
octree2 > clip.pic
```

The second instance of **vwrays** ensures that the correct pixel dimensions are used for images that are not square. For the scene in Figure 1 it was easy to generate a set of clipping surfaces using **genbox** and then transform the box so that vertical surfaces at one corner were between the red and green buildings. In fact, the clipping 'box' was made large enough so that - when in position - it encompassed the location of the desired view point (Figure 2). Not strictly necessary for a box where superfluous surfaces can be removed, just lazy. Note the contents of the octrees:

- Octree1 has the clipping box but does not contain the foreground (red) building. Ground plane and green building optional (actually the ground plane was included and acted as one of the clipping surfaces).
- Octree2 contains the 'real' scene, i.e. red and green buildings, ground plane, sky etc., but it does not contain the clipping box.

Results

The rendering generated using the clipping box is shown in Figure 3. The distribution in daylight illumination across the re-entrant surfaces is now easy to see. In another example, say you have a room constructed using just one outside

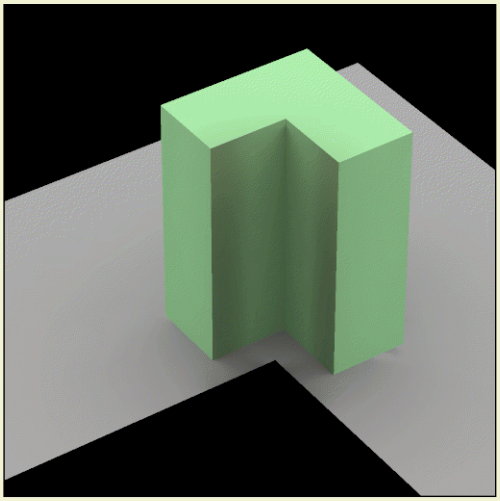

Figure 3. Rendering generated using 'clipping box'

surface. You can use the nearside visible surfaces to do the clipping. See the rendering of a simple box containing a light source with the nearside walls 'clipped away', Figure 4. Note, you don't even need to use two octrees for this.

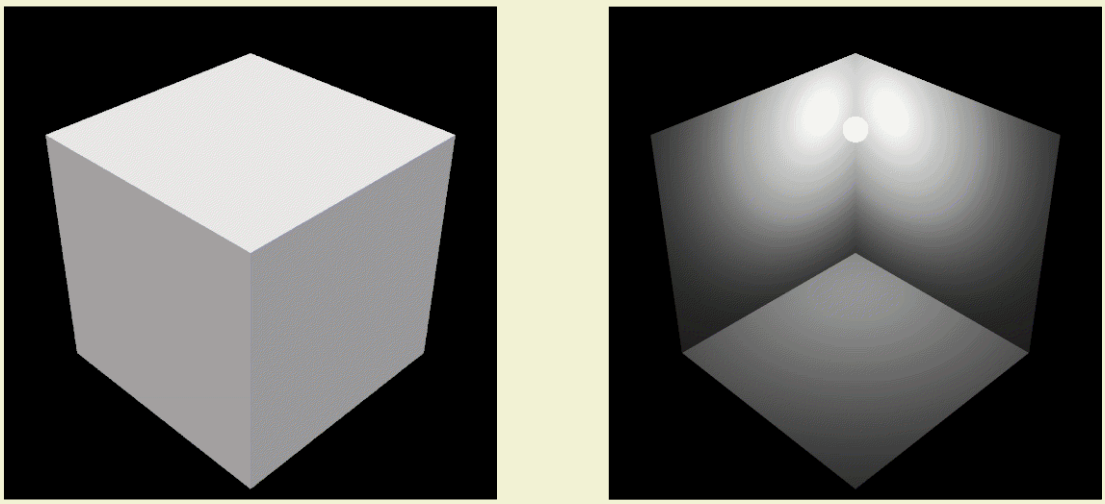

Figure 4. Box with light source

Verdict

An elegant solution to an admittedly niche problem. Moreover, a convincing demonstration of the power of the UNIX 'toolbox' model used by *Radiance*.