Virtual lights: a method for expressive visualization

Domingo Martín, Juan Carlos Torres

Dpto. de Lenguajes y Sistemas Informáticos, Universidad de Granada, Avda. Andalucía, 38, Granada, Spain
[dmartin,jctorres]@goliat.ugr.es

Abstract
There has been great interest in expressive visualization over the last few years. This kind of visualization is used in 2D animation, illustration, and schematic drawing. The flat shading and shape lines, which we call external and internal sketches, are all visual charasteristics of this type of visualization. We present an algorithm for obtaining the sketches from a polygonal model. The algorithm is based on the concept of virtual lights, which allows us to control the visualization of internal sketches in a flexible way.

1. Introduction

Over the last few years there has been a great interest in non photorealistic rendering, or expressive [1] or pictorical visualization. This type of visualization presents advantanges over the photorealistic approach for several applications, in which it is more important to describe than to reproduce the scene exactly. Examples of such applications are the schematic representation of objects, illustration, and 2D animation.

Our work focus on the production of images that look like those that have been drawn by hand, but at the same time trying to make the process as automatic as possible. The principal elements that compose an image with those chararasteristics are flat shading and shape lines, what we call external and internal sketches (Figure 1). External sketches are the silhouettes: the visual borderline between the visible and invisible parts of the object. Internal sketches are the rest of shape lines, which represent a difference in visual attributes, usually a shade or color. A 3D polygonal model has been used because it can be easily handled by a computer, and specialized hardware is available, even at PC levels, for visualizing it. This model allows some operations that are difficult to achieve in a 2D automatic drawing system.

Several papers have been presented over the last few years, describing methods and algorithms, using silouettes, contour lines, and stroke textures, oriented towards illustration[2, 3, 4] and animation[5]. A review of what is called expressive visualization is shown in the Lansdown’s paper[1].

In this paper, we present new methods developed for producing the sketches. We present a formalism that allows us to represent the common charasteristics of external and internal sketches: *virtual lights*. We have used this formalism for implementing a method for obtaining external and internal sketches. The system has been implemented using OpenGL.

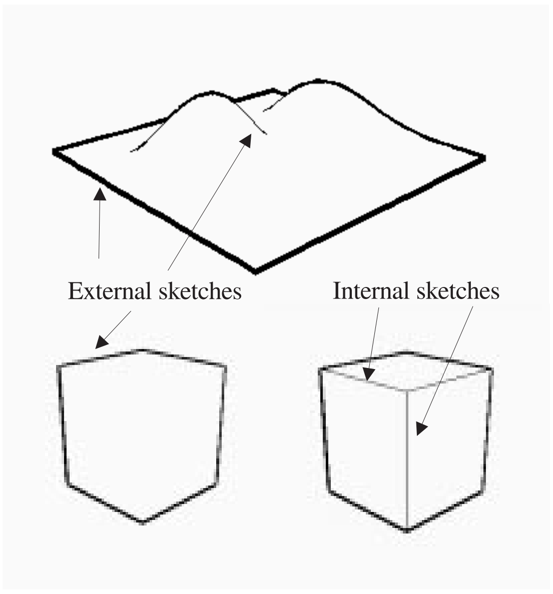

Figure 1: *Examples of external and internal sketches.*

2. *Virtual lights*: selection and classification of edges

Virtual lights is a model that allows us to define both external and internal sketches. Both types of sketches are the limits of a change of an attribute: visibility and color respectively. These attributes are computed depending on the geometry and the position of the observer for external sketches, and also depending on the lights for internal sketches. If we put a light at the same position that the observer, the silhouettes can be considered as the limit where the object is illuminated. In fact, the formulation used to compute the visibility of faces of a model is the same that the diffuse component of a simple illumination model. This is the main idea in *virtual lights*: shape lines, external and internal sketches, can be obtained using the same mechanism. Controlling external sketches, where and when they appear, has no sense (except for rare effects), but it is very interesting to control the internal sketches, because they give us complementary information about the form of the object.

A *virtual light* can be defined as a direction $\vec{v}$, or a position (x, y, z). In the first case, the *virtual light* is located at infinity, while in the second one it is nearer to the scene. *Virtual lights* are different to "normal" lights, because they do not contribute to the lighting of the scene. Another difference is that every object in the scene has its own set of *virtual lights*, not as usually occurs, where the lights are unique for all the elements of the scene. The position or orientation of *virtual lights* is defined in relation to the object that possesses them. Moreover, they have no intensity.

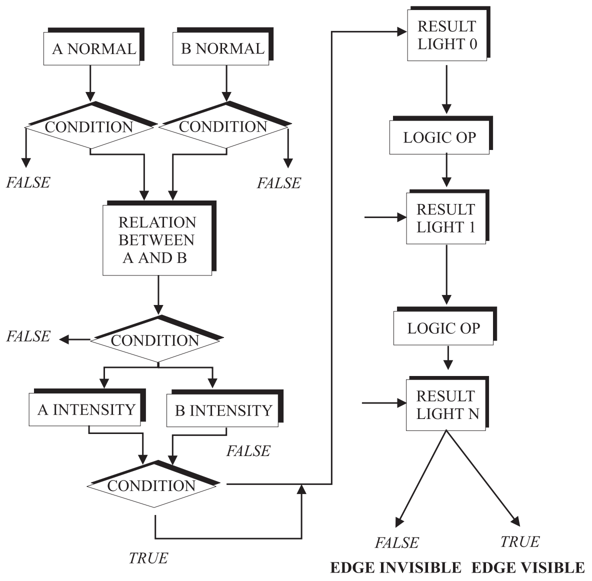

Figure 2: *Testing conditions with virtual lights. The diagram shows a detailed description of individual and global conditions for a virtual light.*

Every *virtual light* has associated one illumination model: diffuse or specular. These models correspond to the simple illumination model components, and they are defined as: $I = cos\theta$, for a diffuse *virtual light*, and $I = cos^n\alpha$, for a specular *virtual light*.

The model is a simplification of an illumination model composed by the Lambertian and Phong components. Because we are interested in relative changes of visibility or shade, the intensity of the *virtual light* and the reflection coefficients can be simplified. They are considered to be 1. The main difference between a diffuse and a specular *virtual light* is that the effect of a diffuse *virtual light* is independent of the observer's position whilst in the case of a specular *virtual light*, the sketches will be selected depending on the position of the *virtual light*, the orientation of the object and the observer's position.

The polygonal model that we use is a boundary representation based on triangles. The winged edge data structure is used for representing the objects. The edges of any object have an attribute for every defined *virtual light*, which can have a value that determines whether the edge is always a sketch, is never a sketch, or is dependant on the operations with the *virtual lights*. For example, this mechanism allows to eliminate external sketches from the union of two joined objects. As our 3D model is polygonal, sketches are closed and non closed sets of selected edges. We also call sketches, of both types, to individual edges when they are selected.

Now we can define how sketches are classified. Given a diffuse *virtual light* located at the same position that the observer, external sketches are those edges which has one face illuminated and the other face not. Every object must have the same diffuse *virtual light*. In order to define internal sketches the user must define some parameters that control, not only the type of *virtual light*, but when, where, and what edges will be selected, based in a difference of color. The conditions that are imposed concern the relative position of faces, individually and globally computed, and the computed difference of reflections. The process is shown in Figure 2. There is an example of where the *virtual lights* are located and how they affect the edges in Figure 3.

As each type of *virtual light* has associated an illumination model, it is sometimes necessary to combine the effects of the *virtual lights*. This is achieved by using a final condition that is applied to the individual results. This condition is a secuence of logical operations that relates the results of the egde to all its defined *virtual lights*. We can choose whether an edge will finally became a sketch if it is a sketch for *virtual light* 1 **OR** for *virtual light* 2, or when it is a sketch for virtual light1 **AND** *virtual light* 2, etc.

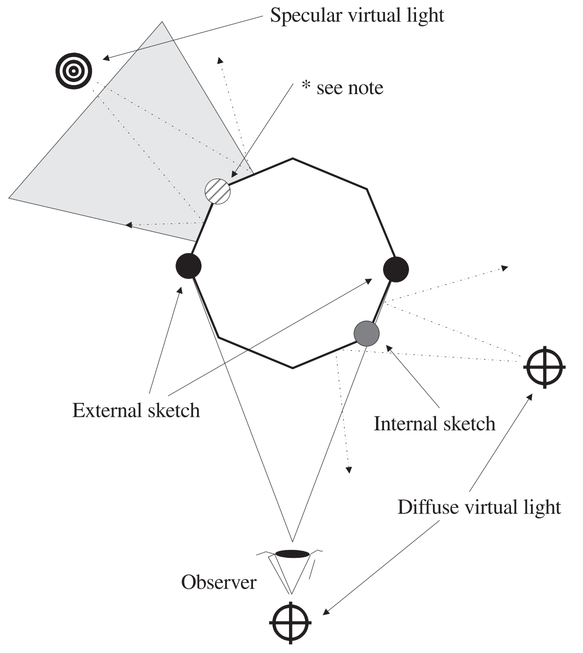

Figure 3: *Graphical description of virtual lights. * The edge will became an internal sketch when the observer enters the gray zone.*

3. The algorithm

The algorithm that allows us to obtain and visualize the sketches processes the edges of every object for every *virtual light*:

```
For every object
  {
  For every virtual light
    {
    Select and clasificate edges
    Compute the intersection between edges
    Form chains of edges
    }
  Visualization of chains
  }
```

The sketches are selected and classified from the polygonal model in the first step. The method of *virtual lights* is applied to the edges for obtaining external and internal sketches. In the second step, the intersections between the edges that are marked as external sketches must be done, because, in general, objects have more than one external sketch, and there are intersections between them. There is a change of visibility in these intersections (one sketch passes behind another). This process is not applied to internal sketches.

In the next step, chains of sketches are formed by edges that are of the same type and have a geometric continuity, that is, they share a vertex. This has the advantage of treating them as a unique element, a curve with its attributes. Otherwise, edges would be drawn individually with their own attributes.

In the last step, chains are drawn using the atributes of edges in a global form, taking into account that the changes of visibilty represent a change in the attributes.

4. Results

The system has been implemented using C++, and OpenGL for producing the images. All the charasteristics of the model are controlled by script files. We have designed several models that are used in producing 2D animation. They form a test bed that shows the capabilities of the system. The images of in Figure 6 are frames of an example which has 22000 faces and runs in real time with a Pentium II 300, 64 MB ram and Linux with Mesa with no hardware acceleration. Another example of the character is in Figure 5

5. Conclusions

We have presented an algorithm for producing images with a 2D appearance. The system uses *virtual lights* for obtaining external and internal sketches. This is a new formalism that allows us to treat both external and internal sketches in the same way. With this method, we can use a 3D model from which it is possible to obtain images with different charasteristics. We can use sketches to improve a real image, to present a scheme, to do 2D animation and illustration. Some examples of models with external and internal sketches are shown in Figure 4. Currently, we are working on improving the visualization process. We hope to achieve more flexible control over the shape of lines, especially thick and very thick lines, as illustration is mainly based on changing lines attributes and their shape or character".

Our current work is oriented towards the optimization of computing intersections. Although the current results are good, there are many possibilities for doing the algorithm faster, it would be even possible to make the computation faster by using a space partition, for instance.

References

1. J. Lansdown, S. Schofield. "Expressive rendering: A review of nonphotorealistic techniques", *IEEE Computer Graphics and Applications*, pp. 29-37, May 1995
2. W. Leister, "Computer Generated Copper Plates", Computer Graphics Forum, 13 (1), pp. 69-77, 1994
3. G. Winkenbach, D.H. Salesin, "Computer Generated pen and Ink Illustration", *Proceedings of SIGGRAPH 94*, pp.469-476
4. T. Saito, T. Takahashi, "Comprehensible Rendering of 3D Shapes", *Proceedings of SIGGRAPH 90*, pp. 197-296

5. D. Martín, J.C.Torres, "Alhambra: a system for production of 2D animation", *Computer Animation 99*, may 1999

6. L. Markosian et al., "Real-Time Nonphotorealistic rendering", *Proceedings of SIGGRAPH 97*, pp. 415-419

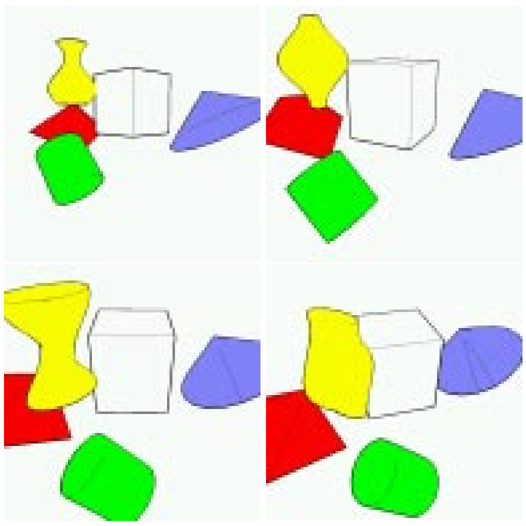

Figure 4: *Secuence showing several objects and the effect of their defined virtual lights.*

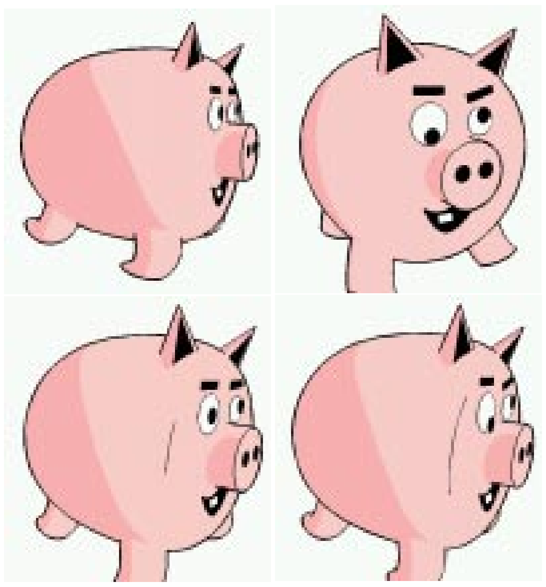

Figure 5: *Mad Pig, which has defined two virtual lights, one diffuse and one specular.*

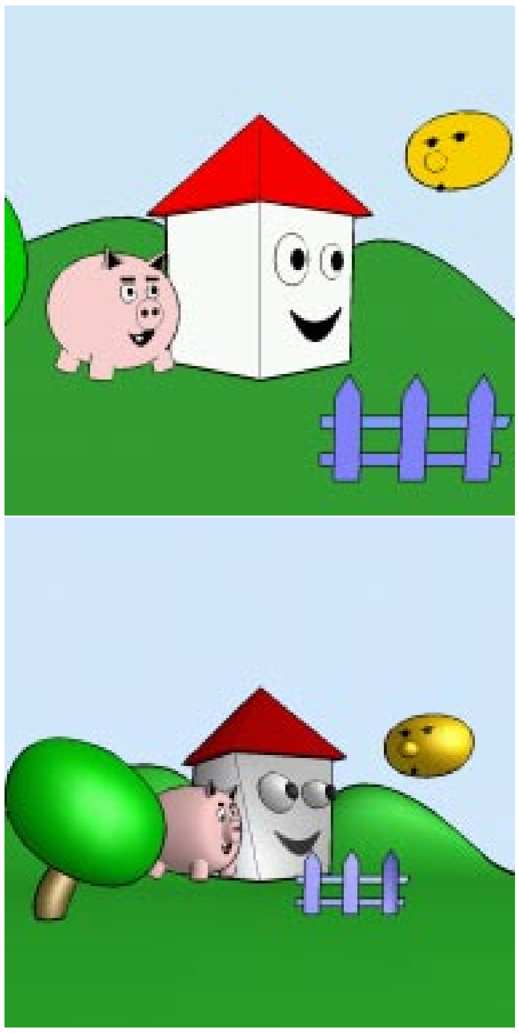

Figure 6: *Example with two frames of 2 animations with flat coloring and Gouraud shading.*