COMPUGRAPHICS 91

First International Conference on Computational Graphics and Visualization Techniques

16-20 September 1991 - "Villas de Sesimbra",
Sesimbra, PORTUGAL

PROCEEDINGS

VOLUME I

Edited by Harold P. Santo

Real-Time Impact Dynamics Simulation

Marie-Claire Forgue and Mikio Shinya

NTT Human Interface Laboratories
1-2356 Take, Yokosuka, Kanagawa, Japan
{forgue/shinya}@nttcvg.ntt.jp

Abstract
This paper describes a simple real-time impact dynamics simulation technique. It involves the use of a fast interference detection algorithm to identify the objects' faces that will interfere. A standard collision detection algorithm is then run only on the much smaller subset of surfaces that have been found to interfere; this can be done very cheaply. We then use physical laws to compute the trajectories of rebounding objects from the collision data. The application described here advances the field of impact dynamics by simulating realistic rigid-body motion in real-time.

1 Introduction

Incorporating physics into computer animation has so far lead to many interesting pieces of work, dedicated either to rigid bodies [3,4] or to deformable bodies [7], or both [8]. These dynamic approaches are successful in achieving realistic behavior (clearly the first priority), but they are too computationally expensive to be usable in practice. They allow neither interactive use, nor real-time simulation, either of which could be used to preview the results with simply rendered or wireframed models. This is in contrast with kinematic or keyframing techniques which do allow interactive feedback.

This paper presents a technique that achieves real-time simulation for physically-based modeling. Our approach involves detecting collision points first, and then, realistically simulating the motion of rebounding objects after impact. Experiments on impact dynamics simulation in [3] clearly demonstrate that collision detection takes up to 90% of the overall computational cost. Yet, the standard collision detection algorithm is not as computationally efficient as it could be: in section 2, we describe a real-time collision detection method which combines the use of a real-time interference detection algorithm [6] and a standard collision detection algorithm [2].

Once one has determined where collision between two objects will occur, one needs

to calculate the resulting trajectory. In section 3, we describe the physical laws involved in impact dynamics and show how they can be used efficiently.

Finally, in section 4, we describe our implementation and provide some experimental results, which confirm that our method allows real-time simulation.

2 Real-Time Collision Detection

Collision detection is generally time-consuming, particularly when a large number of objects is involved. Typically, the computational complexity of existing algorithms like Boyse's [2], for example, is in $O(n^2)$, where n is the number of polygons in a polygonal environment. This is not acceptable for the real-time dynamics simulation needed in computer animation.

In a different paper [6] , we proposed a new type of interference detection algorithm with the following advantages:

- $O(n)$ complexity in the total number of faces of objects,
- speed can be increased with standard graphics hardware,
- applicable to any renderable surface,
- simple and easy to implement.

Basic results from a standard collision detection algorithm, such as Boyse's, are the locations of collision points, and the normal vectors at these points. However, such information is not provided in full by our interference detection algorithm; we thus employ a standard collision detection algorithm, but only for the small subset of faces that do interfere.

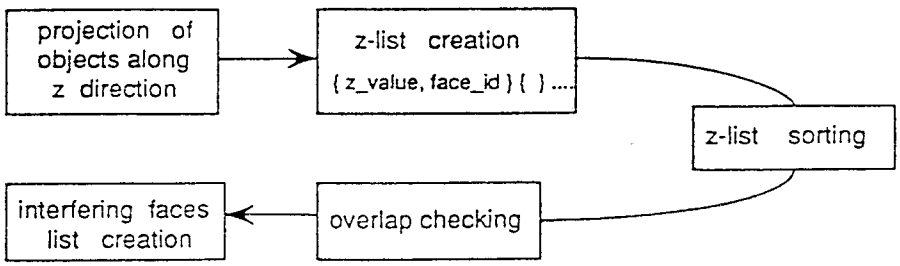


Figure 1: Flow-chart of the interference detection algorithm [6].

Let us briefly explain the real-time collision detection algorithm using the flow-chart given in figure 1. First, given an array of pixels called the z-list buffer, we project along a certain direction (z-direction in this case) and scan-convert each object. For each drawn pixel from the scan-conversion, we built a z-list containing pairs of z-values

and associated face identifiers (face_id's) couples: [*z-value, face_id*], [*z'-value, face'_id*]. Then, when all objects are processed, we sort the z-list against the z-values for each pixel drawn. Finally, if a sorted z-list contains pairs with dissimilar face_id's, interference is detected. An interfering faces list is then created for the z-list buffer.

This list of interfering faces is given to the standard collision detection algorithm. Let us describe the main features of Boyse's algorithm. Collision occurs in the following two basic situations: *(i)* a vertex collides with a face and *(ii)* an edge collides with another edge. In the first case, collision is detected when the trajectory of a vertex intersects the face of another object. In the second case, collision is detected when the trajectory of an edge intersects an edge of another object. In both cases, it can be considered as a face-edge intersection problem.

Objects move along trajectories. Though keeping track of movements can be approximated either as a linear interpolation or a bi-linear interpolation between initial and final positions of a moving object, we choose to decompose any general motion into a translation part followed by a rotation part. Boyse's algorithm allows collisions to be detected between a rotating vertex and a stationary face, and between a rotating edge and a stationary edge, which covers cases *(i)* and *(ii)*. Figures 2.b and 3.b illustrate the way the intersection is found between a circle and a face and between a cone and an edge. In case *(i)*, the impact point is determined as the intersection point between the moving path of the vertex of object A and the faces of object B (figure 2.a). In case *(ii)* (figure 3.a), it is set as the intersection point between the polygon swept by the moving edge of object A and the edges of the polygons of object B.

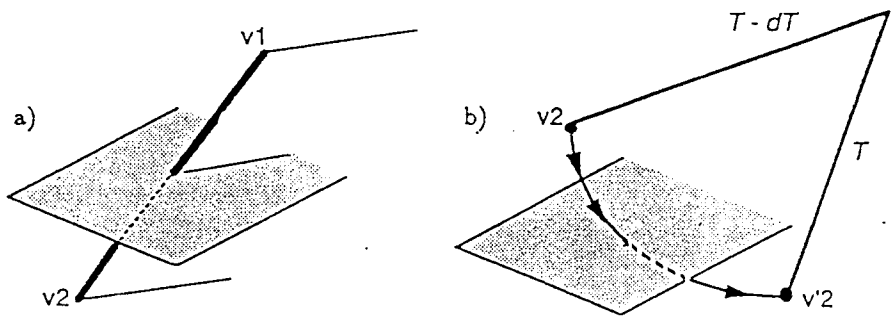


Figure 2: Boyse's case *(i)*

According to the type of contact obtained from collision detection, some rules are applied to obtain the normal vector at the point(s) of contact. In case *(i)*, the normal vector for a contact involving a face is the normal of the face at the contact point. In case *(ii)*, the normal to the contact point is the vector product of the normals of the intersecting edges. Some contacts are said to be indeterminate or degenerate [1],

like the vertex/vertex contact, and one must develop empirical rules to eliminate this indetermination [1,3].

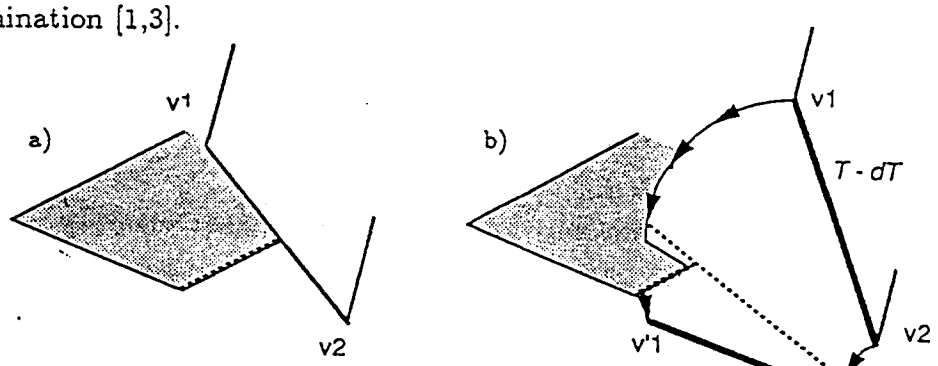


Figure 3: Boyse's case *(ii)*

Figure 4 summarizes the steps in our method. We first perform interference detection with a simple test that consists of comparing the minimum and maximum z values of the moving object with the maximum and minimum z values of the set of the other objects. In case of a positive result from this comparison step, we run the full z-list algorithm. With the produced list of interfering faces, we apply a standard collision detection algorithm using its vertex/face and edge/edge collision tests. To confirm the results and obtain full contact information (positions and normal vectors of the contact points), we apply a static interference check, that is a detection of intersections among objects in fixed positions.

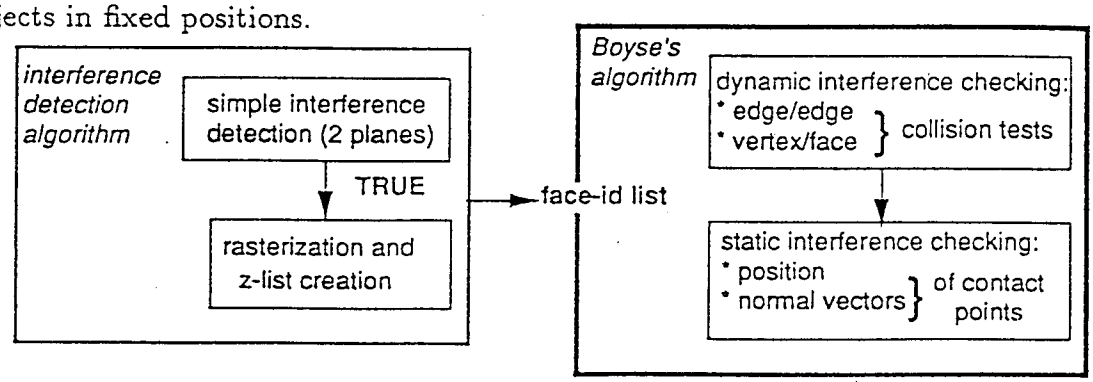


Figure 4: Synopsis of our collision detection method

3 Impact Dynamics

While collision detection is the main requirement for some applications, animation systems incorporating physics must calculate the direction and velocity of rebounding objects. This is the field of impact dynamics, which can be adequately modeled by Newton's laws. A good overview of the theory and equations involved can be found

in [4] and [3]. Below, we summarize the main concepts and results involved in impact dynamics.

Objects have the following physical characteristics: mass m_i, center of mass $\vec{c}_i$, moment-of-inertia tensor I_i, velocity (linear $\vec{v}_i$ and angular $\vec{w}_i$) and momentum (linear $\vec{p}_i$ and angular $\vec{l}_i$).

Dealing with the collision of rigid bodies must guarantee the conservation of linear and angular momentum of the rebounding objects. The change in linear momentum is equal to the impulse, noted $\vec{I}_{mp}$. If linear momenta is noted as $\vec{p}_i = m_i \vec{v}_i$, then the impulse on body A due to impact with body B is equal to the change in the linear momentum of A ($\vec{F}$ is the force):

$$\int_{\Delta t} \vec{F}\, dt = \vec{I}_{mp} = m_A\, (\vec{v}_{A\ end} - \vec{v}_{A\ init}) \tag{1}$$

The impulse on B is:

$$-\vec{I}_{mp} = m_B\, (\vec{v}_{B\ end} - \vec{v}_{B\ init}) \tag{2}$$

As for the change in angular momentum, the angular impulse on A is specified by ($\vec{N}$ is the torque):

$$\int_{\Delta t} \vec{N}\, dt = \vec{r}_A \times \vec{I}_{mp} = \vec{l}_{A\ end} - \vec{l}_{A\ init} \tag{3}$$

with: $\vec{l}_A = \mathrm{I}_A\, \vec{w}_A$.

Similarly, we get the angular impulse on B:

$$\vec{r}_B \times (-\vec{I}_{mp}) = \vec{l}_{B\ end} - \vec{l}_{B\ init} \tag{4}$$

$\vec{r}_A$ and $\vec{r}_B$ are the vectors set from the centers of mass of A and B respectively to the point of impact.

Elasticity of the collision is taken into consideration by means of a Newtonian coefficient, or coefficient of restitution, noted $\mathcal{E}$, whose values range from 0 (inelastic) to 1 (elastic). In terms of energy conservation, an elastic collision is equivalent to no loss of kinetic energy. Newton's rule [5] relates $\mathcal{E}$ to the relative velocities of two particles at the point of contact:

$$\frac{velocity\ \ of\ \ separation}{velocity\ \ of\ \ approach} = \frac{\vec{V}_s \cdot \vec{N}}{\vec{V}_a \cdot \vec{N}} = -\mathcal{E} \tag{5}$$

From equation (5), we can write the generalized Newton's rule for rigid bodies as the relation between $\mathcal{E}$ and the components, along the common normal vector $\vec{N}$ at the point of contact, of the relative velocity before and after the collision:

$$\frac{[\vec{v}_{A\ end} + (\vec{r}_A \times \vec{w}_{A\ end})] \cdot \vec{N} - [\vec{v}_{B\ end} + (\vec{r}_B \times \vec{w}_{B\ end})] \cdot \vec{N}}{[\vec{v}_{A\ init} + (\vec{r}_A \times \vec{w}_{A\ init})] \cdot \vec{N} - [\vec{v}_{A\ init} + (\vec{r}_A \times \vec{w}_{A\ init})] \cdot \vec{N}} = -\mathcal{E} \tag{6}$$

Assuming that objects in contact will not slide, i.e, there is enough friction to prevent it, we can write down the following equations featuring the orthogonal components of the velocity of separation, notated by u and v (they are perpendicular to the normal vector and their directions are given respectively by $\vec{V}_s \times \vec{N}$ and by $\vec{N} \times (\vec{V}_s \times \vec{N})$):

$$(V_s)^u = 0 \quad (7)$$

$$(V_s)^v = 0 \quad (8)$$

These last two equations hold given sufficient frictional force. From Coulomb's law, this implies that: $| (\vec{I}_{mp})^v | \leq \mu | \vec{I}_{mp}.\vec{N} |$ (with μ the coefficient of friction). If the non-sliding condition does not hold, equations (7) and (8) are not valid and must be replaced by assumptions on the orthogonal components of the impulse:

$$(\vec{I}_{mp})^u = 0 \quad (9)$$

$$(\vec{I}_{mp})^v = \mu | \vec{I}_{mp} \cdot \vec{N} | \quad (10)$$

In summary, it is possible to get the final linear and angular velocities for each object, as well as the impulse $\vec{I}_{mp}$, since all unknowns can be calculated with a complete set of equations: (1), (2), (3), (4), (6), and {(7),(8)} or {(9),(10)}. These equations can be solved by a Gauss-Jordan elimination method, for example. Considering the case of impact of a moving object with a stationary object, the system of equations is reduced to 9 equations of 9 unknowns (linear and angular velocity of the moving object, and impulse). Finally, if the relative velocity of the moving object is less than a small threshold, then the colliding objects can be considered to be in continuous contact.

4 Implementation and experimental results

We developed a simulation system using the collision detection method and the impact dynamics approach described above. Figure 5 presents a flow-chart of the system; notice that the most time consuming part of the computation is avoided in the case of non-colliding objects. In our implementation, all objects are clipped to the bounding box of the moving object and then scan-converted to the z-list buffer. Once interference is detected at a given frame, we use Boyse's collision detection algorithm only for the faces found in the interference list. An illustration of this is shown in figure 6. This is a shot of a head and a height field in contact, just before impact dynamics is applied. The colored faces are those contained in the interfering faces list. Since the number of faces in the list is expected to be small, the execution of the algorithm should be quite rapid.

We have run several experiments to validate the system's performance. All tests involved a single moving object colliding with a single stationary object. One of the resulting images is reproduced in figure 6. The two objects, the head and the height field, contain respectively 1875 and 3032 polygons. The animation was displayed on an IRIS 4D/25. For a 512×512 image resolution, the measured processing time is 0.24 *s/frame*. By increasing the number of polygons in the height field support, we get different execution times for the same test scene, which are shown in figure 7. Note that the display time clearly is the largest component of the total execution time. We thus can achieve real-time impact dynamics simulation as long as it is possible to display the image in real-time. Figure 8 gathers shots of other animated scenes. Total execution time and total number of polygons involved in each scene are indicated.

We adjusted the value of the Newtonian coefficient $\mathcal{E}$ for each scene such as the one featuring the expresso machine and the Mandelbrot field, so that all impacts would take place within the Mandelbrot field support. Modifying the value of $\mathcal{E}$, that is the elasticity of the impact, allows one to model different behaviors from the same colliding objects.

5 Conclusion

Real-time impact simulation becomes possible if one uses a simple interference detection method. By managing a list of interfering faces in a polygonal environment, we constrain the number of intersections tested. A standard collision detection algorithm is then invoked in order to get exact collision points, but only for the indicated faces. After collision detection analysis and static interference checking, impact dynamics is applied to produce the simulation. The approach has been implemented and its effectiveness in achieving real-time simulation of dynamic systems has been confirmed experimentally.

References

[1] D. Baraff. 'Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies'. In *Computer Graphics* 23 *(3)*, pages 223–232, July 1989.

[2] J.W. Boyse. 'Interference Detection among Solid and Surfaces'. *Communications of the ACM*, pages 3–9, January 1979.

[3] J.K. Hahn. 'Realistic Animation of Rigid Bodies'. In *Computer Graphics* 22 *(4)*, pages 299–308, August 1988.

[4] M. Moore and J. Wilhelms. 'Collision Detection and Response for Computer Animation'. In *Computer Graphics* 22 *(4)*, pages 289–298, August 1988.

[5] R.A.Becker. *Introduction to Theoretical Mechanics*. McGraw Hill Book Company, 1954.

[6] M. Shinya and M.C. Forgue. 'Interference Detection through Rasterization'. To appear in the *Journal of Visualization and Computer Animation* 2 (4), October 1991.

[7] D. Terzopoulos. 'Elastically Deformable Models'. In *Computer Graphics* 21 *(4)*, pages 205–214, August 1987.

[8] D. Terzopoulos and A. Witkin. 'Physically Based Models with Rigid and Deformable Components'. *IEEE CG&A*, November 1988.

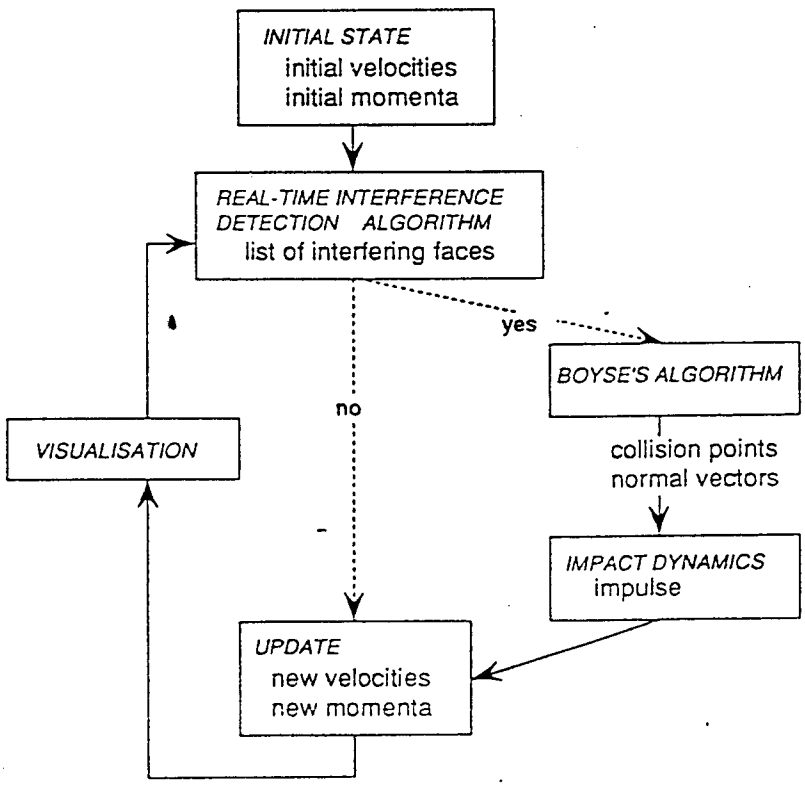


Figure 5: Impact dynamics simulation system

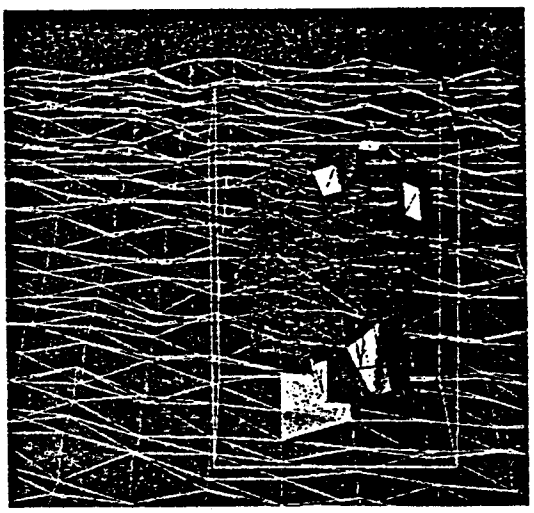


Figure 6: Interfering faces

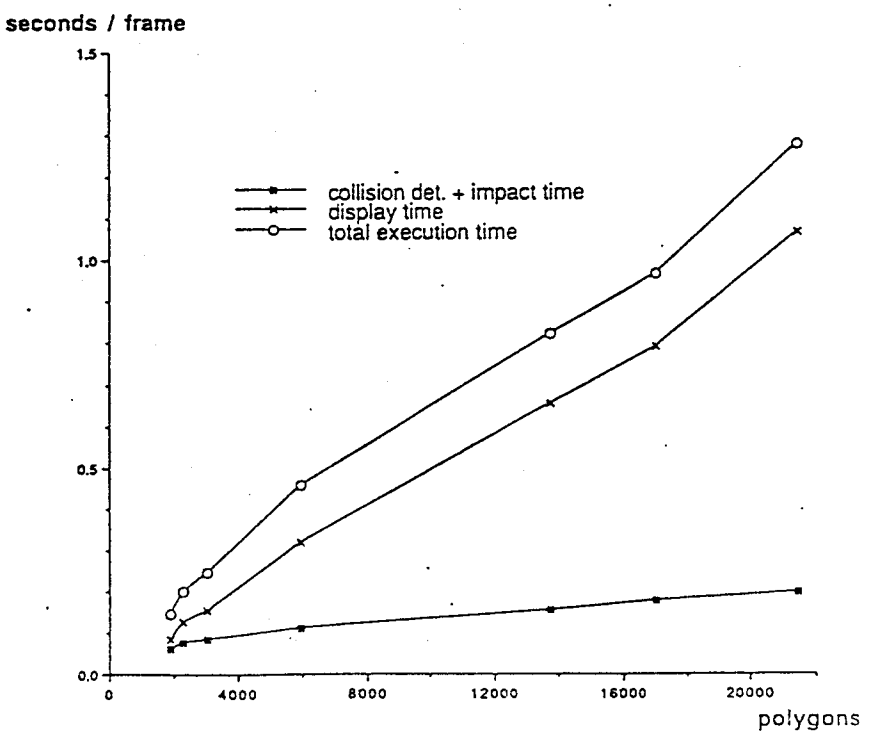
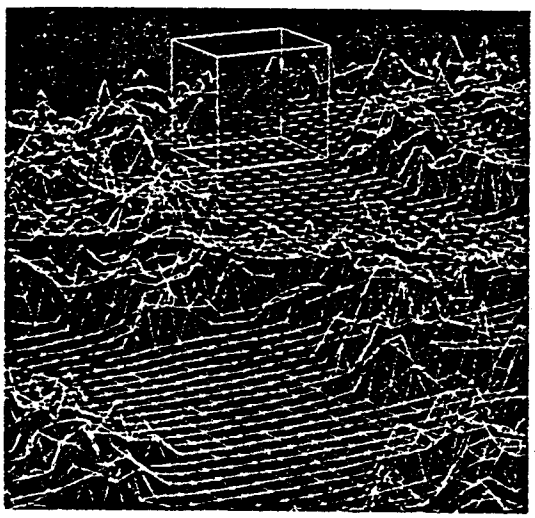
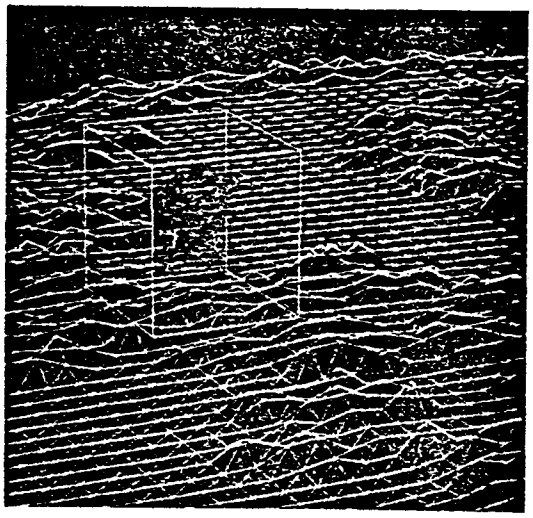


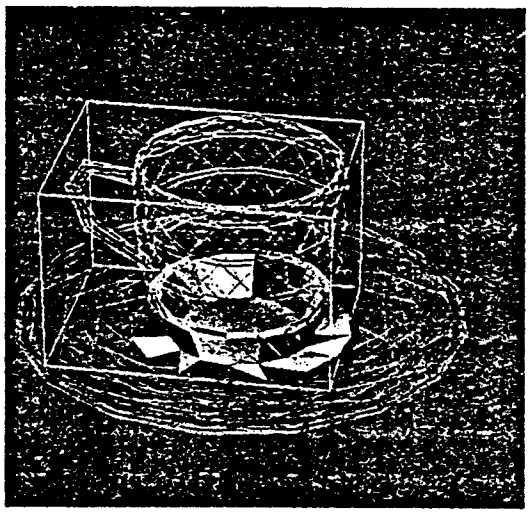
Figure 7: Execution times v.s. number of polygons



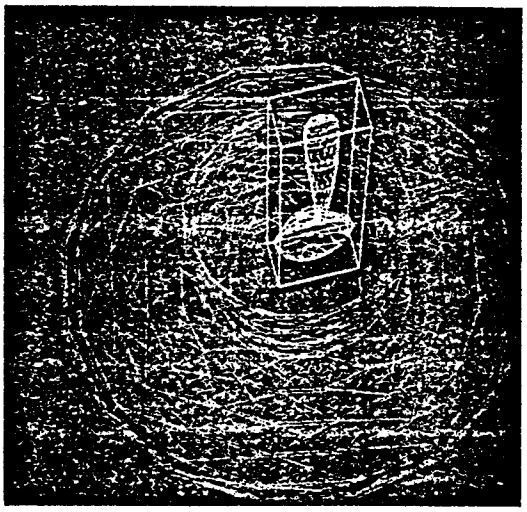
(576 + 7938) polygons: 0.82 $s/frame$



(1777 + 7938) polygons: 0.66 $s/frame$



(450 + 386) polygons: 0.35 $s/frame$



(836 + 512) polygons: 0.53 $s/frame$

Figure 8: Other tested scenes