Abstract.

During the last ten years computer graphics has graduated from the laboratory into the public domain. In the last five years the computer animation industry has reached the hundred million dollar mark and is still growing. Presented here are some of the more important recent developments in computer graphics that have made this growth possible and how these advances benefit the simulation community.

Keywords *Graphics, Animation, Realism, Parallelism, Distributed processor, Man-machine interfaces.*

1. INTRODUCTION

Computer Graphics is a continually evolving field of computer science, in this paper we bring up to date the developments described in a previous SCS conference, see [Wyvill 85]. Graphics as a medium for interaction has gained much popular support with the development of user interfaces that rely on windows, pop-up and pull-down menus, icons, scroll bars, buttons etc. Environments for interacting with any application with little or no use of the conventional keyboard can be seen on many workstations. Powerful workstations such as the Sun and Apollo, personal computers such as Macintosh, Atari and Amiga offer the applications programmer a toolbox of facilities for manufacturing an application with a graphical interface.

There are basically three ways in which graphics can help the field of simulation:

- To enhance the simulation results
- To facilitate the debugging and production of simulation programs
- To provide an interactive dialogue with a running simulation

Langlois' SIMSEA [Langlois 84] is a good example of graphics used to enhance the results of a simulation. SIMSEA couples a general purpose graphics language with a tool specifically designed for simulation, namely the class **SIMULATION** of Simula. In SIMSEA the animation of a bank simulation might show simple stick figures queuing up at the wickets in the bank. The ANDES [Birtwistle et al 84] system takes a different approach and is an example of a graphical debugging and program development aid. ANDES animates the underlying mechanism of the simulation in addition to animating the behaviour of the simulation model. ANDES also allows the user to observe a simulation from several perspectives or views. These views cover the spectrum from overall pictures of the status of the whole model down to full details of the current data values and actions of an individual process. Users may supplement the pre-defined views of ANDES with their own views tailored to specific applications, such as those provided by SIMSEA. The third point was covered by Jean Vaucher in a panel discussion on future directions in simulation software at the 1984 SCS conference. Vaucher described the application of the video game paradigm to simulation. To allow a manager or non-programmer to experiment with a simulation model he must have an easy to use fifth generation style of user interface. Perhaps with an expert system to guide him, the manager must be able to try out different alternatives by using graphical operations such as picking and pointing or perhaps voice input to the model. The *manager* may be an expert in a field other than programming and he should receive graphical feedback in real time to give him a better understanding of the simulation model. An aircraft simulator is a good example of such a system designed for a very specific situation which has no keyboard inputs and good animated graphics output and a user whose expertise is not necessarily in programming. Since computer graphics has reached the stage where realism can be achieved at progressively smaller costs there are many opportunities to use this technology and enhance the reputation of simulation by producing systems which non-specialists can immediately find useful. There are many examples of where better graphics can considerably add to simulation, clearly Langlois' wire frame

animated figures painstakingly filmed frame by frame would be more convincing produced as full colour 3D shaded images in real time. Training in situations where danger to life is involved is another example. Beside pilot training there are other applications such as fighting forest fires or coping with oil and gas blow outs that could use realistic graphics driven from a simulation model to good effect. It is therefore important for the simulation community to be aware of what is happening in graphics and how far these developments have gone to provide the sort of support that will enable the production of real time high quality realistic animation.

2. MODELLING

Mainstream graphics research has been concentrating on image synthesis since the advent of cheap raster displays. The problem is to design some three dimensional object and render it as realistically as possible. If animation is desired then motion specifications must be associated with the object and its parts. Three dimensional model building uses a variety of primitives and various systems offer different ways of constructing objects. There are three commonly used methods of representing surfaces; *Polygon Meshes, Quadric Surfaces and Parametric Bi-Cubic Surfaces.* Although easy to manipulate and render, polygon meshes suffer from some problems. To achieve the appearance of a curved surface a normal averaging technique is used which produces a smooth looking interior mesh with some defects such as the polygon silhouette at the edges. This type of technique is useful for planar surfaces such as modelling buildings and there are many algorithms for manipulating and rendering polygons.

Quadric surfaces [Math 68] are represented by quadratic functions such as a sphere or cylinder and objects composed from these primitives. This technique is useful in applications where the world can be so described, for example modelling a world of spheres as in a molecule display program.

A more flexible modelling technique is to use a parametric bi-cubic patch. The shape of the patch can be altered by editing a set of control points. Unfortunately, unless a large number of points are used, moving a control point changes the shape over a large area. Recent work on beta splines [Barsky and Beatty 84] has produced a patch whose shape can be finely controlled using two extra parameters at each control point called *bias* and *tension.*

There has been a strong emphasis in the computer graphics research community on modelling natural phenomena. These objects divide into two broad classes, those that have well defined surfaces and those that are more amorphous in nature. For many objects of the former class the surface techniques described above are adequate. However other objects such as a ball hitting the ground, undergoes a shape change in response to its surroundings. These objects have been termed as soft [Wyvill et al 86]. None of the above techniques lend themselves to the description of soft objects. This class of objects includes fabrics, cushions, living forms, mud and water. Such objects can be represented as a surface of constant value in a scalar field over three dimensions. The object can then be thought of as a skeleton of control keys (points or lines) which are then covered by the surface. Depending on the function used to describe the field an individual control key point will be covered by a sphere, and a key set of 3 line segments give rise to an ellipsoid. As the keys are moved towards each other the surfaces merge. This technique is particularly useful when describing objects to be animated. The control keys can be moved independently and the surface added later. Thus objects can be moved and their shape can change.

Other objects which can be described as amorphous, fluids such as fire or smoke can be described by a variety of techniques. Scalar fields have also been used to describe amorphous phenomena by merging objects with simple geometries. Given a threshold value of the scalar function the object is defined by a 3D grid of cells. An object is placed in a cell where the function is greater than a threshold value. A cruder representation is obtained compared with the soft technique, but the objects can built in real time. This approach has been used in medical applications for the display of human organs from tomograms. A summary

of amorphous modelling techniques is given in [Upson 86].

Modelling amorphous phenomena is particularly useful for a class of simulations. For example simulating a forest fire on a mountain side to help train fire fighters uses objects such as fire and trees which do not lend themselves to the geometric surface representations. In such a simulation it is important to model these objects accurately since the more realistic the simulation, the better the effect of the training.

Mandelbrot [Mandelbrot 83] put forward the idea of *fractal geometry* in which he identifies the self similarity in natural objects. To obtain the desired complexity a fractal is non-deterministic, each generation being found from the last using a stochastic process. Unfortunately such processes generate a very large amount of data and it is difficult to fit such procedurally defined objects in an existing system which uses objects defined with a more cubist approach as surface patches and polygons. However recent advances using particle techniques by Reeves [Reeves 83] and others have had some success in generating realistic fuzzy objects. The genesis effect in the film *Wrath of Khan* [Smith et al 82] which shows a fire spreading across a planet followed by a fast flight over the surface populated with fractal mountains has become a landmark in computer graphics. Since then more work has been done on particle systems. The concept of Graftals [Smith 84] is an idea used to describe plants and trees. Fractals are strictly non-deterministic whereas with graftals a deterministic procedural approach is taken which adequately provides sufficient complexity. Using this idea Smith has produced some extremely realistic images.

Other recent work has tackled the problem of describing waves in water, two recent papers suggest ways of representing waves approaching and breaking on a sloping beach. Peachey uses a phase function which produces wave refraction and other depth effects, and a wave profile which changes according to wave steepness and water depth. Particle systems are used to model the spray and effect of obstacles placed in a waves path. The waves are finally rendered using small polygons to approximate wave patches defined by a parametric representation [Peachey 86]. Fournier and Reeves use a similar model based on a parametric representation of waves which also takes into account wave trains [Fournier & Reeves 86].

The trend in modelling techniques is to develop graphical techniques to match more accurate simulation of natural phenomena.

3. RENDERING TECHNIQUES, THE PRICE OF REALISM

To produce a high quality image on a relatively low quality raster display from some description of the model is known as rendering. This process is the subject of much research as there are many trade offs depending what sort of quality is required and how much available machine time there is to perform this operation. There are a number of basic problems:

- Project the objects in perspective and remove hidden surfaces.
- Shade the objects according to the position and colour of light sources.
- Account for reflections and refraction through transparent objects.
- On low resolution displays defocus the edges to reduce aliasing.
- In animation reduce temporal aliasing.
- Produce shadows.

Various techniques have been developed, although ray tracing seems to solve most of the above mentioned problems. Goldstein [Goldstein 71] describes the technique as " ... basically a simulation of the physical process of photographing an object." Some of these features, such as reflections and refractions can be included in other hidden surface algorithms, e.g. scan-line and area sub-division [Sutherland et al 74]. However, these algorithms treat such surfaces as special cases whereas ray tracing deals naturally with them. The final picture quality provided by ray tracing has to be paid for by a large amount of computation, proportional to the number of pixels in the picture (or more to accommodate anti-aliasing) times the logarithm of the number of surfaces in the scene [Rubin and Whitted 80].

Ray tracing works by reversing the physical passage of a light ray. Rays are traced backwards from the eye through each pixel into the surfaces representing the scene. When the ray encounters a surface, three things may happen. If the surface is matte then the light intensity at that point on the surface will be taken as the intensity of the pixel the ray passed through. If the surface is a (partial) reflector then a new ray is started in the direction of the reflection. The final intensity of this ray will make a (partial) contribution to the intensity of the pixel. If the surface is (partially) transparent then a new refracted ray is generated passing through the surface. Again it will make a (partial) contribution to the pixel's intensity. Depending on the surface all three of these effects may occur and their respective intensities will be added to give the final pixel intensity. More than one ray per pixel must be used if antialiasing is required. Whitted [Whitted 80] uses four rays per pixel increasing the computer time proportionally.

In practice greater realism is achieved by attention to fine detail, for example modelling shadows gives a good cue to realism but takes substantially more computer time since the hidden surface algorithm has to be repeated from the point of view of the light sources. An even better cue is given by modelling light sources with a finite width thus producing fuzzy shadows (the penumbra). This has been done by a variation to ray tracing by using thick cone shaped rays [Amanatides 84]. Unfortunately such attention to detail corresponds to larger amounts of machine time necessary to render the object and real time ray tracing is yet to be done.

Recent work has concentrated on improving the lighting model. In ray tracing lights are modelled as points in space, thus shadows are sharp and do not show umbra and penumbra. (Although this problem has been solved using sampling techniques indicated above). The procedure can only model intra environment reflections in the specular direction and each time a new view is required the calculation must be repeated. The radiosity method see [Greenberg et al 86] determines the global illumination of the environment independent of the viewer position. However although this approach produces accurate results and it requires an enormous amount of computer time to use this method, since each surface in the scene can potentially add to the diffuse illumination of every other surface.

High quality pictures can be produced without the cost by faking the effect of surface detail. Texture mapping plays an important part in enhancing the visual complexity of an image to give the impression of surface detail, without the cost of manufacturing a geometric model of that detail. It is usually achieved by using some coordinates of position on a surface as input to a texture function or as selectors in a texture table. Environmental reflection maps provide the effect of reflections without going to the expense of ray tracing. Bump maps provide the effect of raising bumps or depressions in a surface by perturbing the direction of the surface normals. [Blinn 78]. More recently Peachey [Peachey 85] has proposed using a three dimensional texture space. This has many advantages. For example, the surface of marble or wood-grain are a consequence of the way in which the surface is part of the continuous solid texture of the material [Perlin 85; Peachey 85].

4. ANIMATION

It has been pointed out [Thalman and Thalman 85] that in the last ten years much research has been devoted to the development of rendering algorithms, but little to the equally difficult task of defining complex motion. One of the reasons is the lack of suitable modelling primitives. In modern animation systems the three most commonly used modelling primitives are: polygon mesh, spline surface patches and quadric surfaces. These techniques are not well suited to represent motion of objects which change shape over time. Thalman also [Thalman and Thalman 85] introduces the concept of laws to define local and global methods for governing the way in which objects in his system move.

In animation we wish to simulate the motion of real objects. This motion control falls roughly into two classes: simulation and illusion. In order to look natural, any animation has to represent possible motion in the physical world. In this sense, the best animation is based on detailed simulation which takes into account the physical laws which govern motion. In such a simulation, a mathematical model representing these laws, produces the desired effect automatically. In many applications, it is not necessary for an animation sequence to follow such a physical model. All that is required is to convince the human eye that the motion is one that would be seen in the real world. Fairing in hand animation can be regarded as a crude attempt to use a few simple rules to do this. Such techniques attempt to model the visual effect we refer to in this paper as illusion. In a sense, any approximation, however crude, can be thought of as simulation.

There are several different approaches to the problem of defining the way in which objects can be made to move. At the lowest level many animation systems simply offer geometric transformations over time, or inbetweening of an object from one key position to another. Although these primitives may be built up hierarchically to obtain motion, to describe something as complex as a human movement simulation requires more sophistication. For example [Badler et al 80] and [Zeltzer 82] use a goal-directed approach. A high level command such as *walk to the door* can be given, to which the system will apply a suitable pre-defined gait. Some computational aspects of this technique can be found in [Korein and Badler 82]. Some work has been done on human facial animation of which [Parke 82] is the best known. A good summary of research in this area is given in [Badler and Smoliar 79], for a summary of the issues involved, consult [Calvert et al 82].

Recent work has been done on the animation of soft objects (mentioned under modelling above). Shape change can be achieved by moving the control points and adjusting the spatial relationship between the points in a controlled manner. Since the object can also change shape, motion specification for these models becomes more difficult to express than for rigid objects. To some extent this problem is offset when these objects are represented as an iso-surface in a scalar field, since one or more closed surfaces will be formed no matter how the control points are moved [Wyvill et al 86]

5. GRAPHICS HARDWARE

The two major bottlenecks in the speed of producing high quality graphics are the time taken to do floating point operations such as matrix multiplies and clipping operations and the time taken to render the objects. For example ray tracing may take anywhere from several minutes to several hours of computer time to produce a single frame. In film animation 24 frames per second have to be produced. For video, thirty frames per second; thus a few minutes of film can take weeks to generate. For a running simulation to produce high quality realistic results the problem becomes one of needing processing speeds that cannot be provided by even todays supercomputers. A glossy magazine provides picture quality of something of the order of 2000 dots per inch. A 512 by 512 pixel display gives only the order of 40 dots per inch. Assuming a 512 by 512 pixel array is adequate with sufficient anti-aliasing each frame has to be processed in 1/30th second. Whatever the method of representing objects to describe any realistic scene requires thousands of data values each of which has to be pushed through some sort of pipe line of transformation and clipping operations before being rendered. Even if

the data is processed through the geometry pipeline in a reasonable amount of time the rendering process must be fast enough to produce thirty frames a second.

Much research is going on in hardware to produce parallel and VLSI architectures capable of handling the problem of speed in computer graphics. A significant contribution in this area is the Clark geometry engine [Clark 82] which is a series of 12 specialised VLSI processors designed to do the calculations prior to rendering (matrix multiplies and clipping operations). More recently several hardware manufacturers have developed specialised graphics hardware. The 32-bit Texas Instruments 34010 graphics processing chip [Asal et al 86] combines both specialised graphics instructions, such as pixel block transfers, and general-purpose instructions. This makes it flexible for graphics applications, and also suitable for non-graphics applications. The chip is designed to retain the speed found in specialised hardware controllers, while giving the flexibility of a general purpose programmable processor.

One of the properties of ray tracing is that the rays are independent of each other and a particular ray can be computed in parallel with any other ray. Several groups are working on parallel architectures for ray tracing. For example at the University of Calgary a four processor machine has been built comprising of a two dimensional systolic array of M68000 processors. [Cleary et al 83]. In the processor array a number of independent processors are connected by high speed links (comparable to processor speeds). The links are confined to those processors which are physical neighbours. Because the processors run independently sharing data only over the links and they execute different instructions on different data, they can be classified as MIMD (Multiple Instruction Multiple Data) systems [Hockney and Jesshope 81]. The great advantage of such systems is that because there is no global buss or global shared memory the system can be very easily laid out with entirely local wiring for signal paths. Using a suitable algorithm which efficiently uses the local communication links they can provide a very cheap way of constructing significant computing power. Indeed if a two dimensional network of links is used then there seem to be no limiting factors other than cost to growing the system indefinitely although ray tracing performance begins to decline after a certain size array is reached. Software support for developing and testing different distributed algorithms on this machine comes from the JADE [Unger et al 84] distributed software prototyping environment. JADE provides the tools necessary to develop, monitor and debug distributed software of this nature. Amongst the many problems faced by such architectures is to adequately distribute the picture over the available processors. This task can be very complex for example in the case of a consisting of a parabolic mirror focusing the rays into a small area of the screen. This problem has been tackled by balancing the load using an adaptive subdivision algorithm [Dippe and Swensen 84].

The structure of graphics systems is another area which is receiving some attention especially when distributed algorithms must be found to take advantage of multi-processor machines. Earlier graphics systems held a linear *display list* of primitives which were continually refreshed by the display processor. Structuring this into a hierarchy can lead to many advantages. Such a structure is discussed in [Wyvill et al 84a] and [Wyvill et al 84b]. Besides economy of storing a model part only once and reusing it higher in the hierarchy the structure traversal can take place on a remote processor. The possibility of making each sub-tree a separate process is being examined in order to take advantage of parallelism. The Apollo 580 uses a similar hierarchical scheme as its display list [Henderson et al 86; Apollo 85]. The bottleneck however for high quality graphics will still be the rendering process for the forseeable future. Currently 3D animated graphics can be achieved on a small display area in real time by leaving out many of the subtler and slow to calculate realism cues such as shadows and specular reflections. There are also severe limitations on the quantity of data that can handled although again algorithms are being produced whose processing time tends to depend on the number of pixels rather than on the number of graphical objects [Cleary et al 86].

Several machines capable of real time rendering a limited number of objects (usually polygon based) are now available. The most exciting of these is Pixel Planes developed by Henry Fuchs at the University of North Carolina [Fuchs et al 85]. The machine uses a one processor per pixel architecture at 512 by 512 resolution. Thus a quarter of a million processors have been built using custom VLSI chips. Polygon descriptions are sent to the machine and each processor (pixel) decides if it is inside the current polygon. Real time animation at

comparatively high quality is thus possible.

6. GRAPHICS & ROBOT SIMULATION

Graphical simulations of robots are useful in at least two ways:

- as an aid to research, where the robot and its environment need to be easily changed
- as an aid to design of robots, robot systems, and robot tasks

Robots lack teachability and adaptability [MacDonald 84; MacDonald in press; Lozano-Perez 83; McLaughlin 82; Bonner and Shin 82; Hasegawa 82; Ambler et al 82; Rosen 79; Nitzan 79; Benati et al 80]. Advanced robot systems are limited to specific tasks and their associated environments, and may be difficult to extend [Waltz, 82]. New schemes for teaching and programming robots need to be demonstrated, establishing their consistency and implementability. Simulations avoid the difficulties of real demonstrations — being flexible and free from unimportant details — so long as the simulations are correctly validated. We have found it possible to animate a very simple robot simulation in real time, using a Sun 3 and its graphics processor. The robot comprises only a few polygons, and responds as quickly as a user can move a controlling mouse. Although crude, we expect this kind of animation to be suitable for robot teaching and programming experiments. The teacher is able to lead the simulated robot through a task interactively.

Robots are becoming more popular in manufacturing industries [ElMaraghy 86]. There is an increasing need for methods of aiding robot programmers. The normal industrial teach method is limited to effectively teaching fixed sequences, but modern robots are capable of performing much more complex tasks. Graphical simulations aid the design and development of robots by robot designers. Graphical simulations also aid robot users, enabling them to simulate robots as parts of other systems, and simulate robots performing individual tasks. The latter may become important as an aid to robot programming; the teacher may have more control of the simulation than of a real robot, facilitating the programming and debugging of a task.

Below are discussed some systems for animation and simulation of robots and general articulated figures.

[Elmarghy 86] describes a system for modelling and simulating two modern robots; the PUMA 560 and the ADEPT I. The system may be used for interactively simulating assembly tasks carried out by any open chain mechanism. An expert system under development is intended to produce task plans for a robot, given a specification of an assembly task.

[MacKay and Tanner 86] describe *Adagio*, a robotics simulation workstation under development. The workstation is a multiprocessor, multitasking system, the simulation being implemented as several tasks. Near

real time animation of a robot and environment is planned.

General methods of representing three dimensional articulated figures are addressed by [Cachola and Schrack 86], [Badler 86], [Ridsdale et al 86] and [Armstrong et al 86]. One system [Cachola and Schrack 86] enables the description of segments and joints, and the specification of motion in a structured programming language. Only open kinematic chains can be modelled. Figures can be defined hierarchically, for example a hand can be defined and then used in an arm definition. Motion can be explicitly specified for each joint for a set of key frames. These basic motions can then be combined and synchronized, forming complex motions of articulated figures. One example given is of a motion *my_walk*, which comprises synchronized motion of *walking_legs* and *swinging_arms*, where *walking_legs* is defined as a *walk* by a number of legs, and so on. [Badler 86] aims at understanding human motion, by simulating and animating human motion. Simulations verify representations proposed for human motion. A general approach to motion understanding would cover

- geometry, kinematics and dynamics
- the goals of movements
- the agents mode of behaviour (e.g. defensive vs threatening)
- any abstract signs made by the motion (e.g. the same motion can signify a touch, press or punch)
- relationships between a movement and other concurrent or synchronized movements

[Ridsdale et al 86] aim to develop motion descriptions of articulated figures, both at the detailed level and at the scene level. In the long term they propose using knowledge-based inference in a *director's apprentice*, a system that will learn from a human director and then be able to produce motions given scene level descriptions. The detailed level system has been used by a skating choreographer and a dance choreographer. The scene level system is expected to be used by film and theater directors. [Armstrong et al 86] describe a user interface to a human figure dynamics modelling system, intended to alleviate the difficulty of computing the torques and forces needed for a dynamic analysis for animation. They expect to be able to use the techniques to achieve real time animation on a network of four SUN 3 workstations.

[Drewery and Tsotsos 86] describe a prototype animation system that executes movements in 3D given English motion commands. A planning system forms a plan for a task goal, using a frame-based knowledge base. The objective is to enable task level description of animation, where the user does not have to specify details at the motion level.

[Wilhelms 86] describes an interactive graphical motion editor in which motion can be specified kinematically as positions over time, or dynamically as motions caused by forces and torques being applied to masses. While kinematic motion is more easily computed, dynamic motion is easier to specify when there are

complex interactions between objects.

7. CONCLUSION

During the last ten years computer graphics has graduated from the laboratory into the public domain. In the last five years the computer animation industry has reached the hundred million dollar mark and is still growing. Presented here are some of the more important recent developments in computer graphics that have made this growth possible and how these advances benefit the simulation community.

Graphics as a medium for interaction has gained much popular support with the development of modern user interfaces. Graphics can help simulation by enhancing results, facilitating debugging and program production, and providing an interactive dialogue with a running simulation. Since computer graphics has reached the stage where realism can be achieved at progressively smaller costs there are many opportunities to use this technology and enhance the reputation of simulation by producing systems which non-specialists can immediately find useful. It is therefore important for the simulation community to be aware of what is happening in graphics and how far these developments have gone to provide the sort of support that will enable the production of real time high quality realistic animation.

Acknowledgements

The *JADE* project and the Systems Research and Development Group (SRDG) at the University of Calgary have been particularly supportive of our work in distributed graphics. This work, *JADE* and SRDG are supported by the Natural Science and Engineering Research Council of Canada.

References

Amanatides, J. (1984) "Ray Tracing with Cones," *Computer Graphics (Proc. SIGGRAPH 84), 18* (3) July, 129-135.

Ambler, A.P., Popplestone, R.J., and Kempf, K.G. (1982) "An Experiment with the Offline Programming of Robots" in *Proc. 12th Int.Symp. on Industrial Robots; 6th Int.Conf on Industrial Robot Technology.*, pp 491-504. Paris, June.

Armstrong, W.W., Green, M. and Lake, R. (1986) "Near-Real-Time Control of Human Figure Models" *Proceedings, Graphics Interface'86* 147-151 Vancouver 26-30 May

Apollo Computer Inc (1985) "Programming with DOMAIN 3D Graphics Metafile Resource" Release 9.0, Chelmsford, Mass.

Asal, M., Short, G., Preston, T., Simpson, R., Roskell, D. and Guttag, K. (1986) "The Texas Instruments 34010 Graphics System Processor" *IEEE CG&A 6* (10) 24-39.

Badler, N.L. (1986) "Animating Human Figures: Perspectives and Directions" *Proceedings, Graphics Interface '86* 115-120 Vancouver 26-30 May

Badler, N., O'Rourke, J. and Kaufman, B. (1980) "Special Problems in Human Movement Simulation" *Computer Graphics 14*(3) July

Badler, N. and Smoliar, S. (1979) "Digital Representations of Human Movement" *Computing Surveys 11*(1) March

Barsky, B. and Beatty, J. (1983) "Local Control of Bias and Tension in Beta-Splines" *Computer Graphics (Proc. SIGGRAPH 83), 17* (13) July, 193-218.

Benati, M., Gaglio, S., Morasso, P., Tagliasco, V., and Zaccaria, R. (1980) "Anthropomorphic Robotics, Parts I and II" *Biol.Cybern., 38*, 125-150.

Birtwistle, G., Joyce, J. and Wyvill, B. (1984) "Andes an Environment for Animated Discrete Event Simulation" *Proc. UKSC*, Bath, Sept

Blinn, J.F. (1978) "Simulation of Wrinkled Surfaces," *Computer Graphics (Proc. SIGGRAPH 78), 12* (2) July, 286-292.

Bonner, S. and Shin, K.G. (1982) "A Comparative Study of Robot Languages" *Computer, 15* (12) 82-96.

Cachola, D. and Schrack, G. (1986) "Modelling and Animating Three-Dimensional Articulate Figures" *Proceedings, Graphics Interface'86* 152-157 Vancouver 26-30 May

Calvert, T.W., Chapman, J. and Patla, A. (1982) "Aspects of the Kinematic Simulation of Human Movement" *IEEE Computer Graphics and Applications* Nov

Clark, J.H. (1982) "The geometry engine, a VLSI system for graphics" *Proc. SIGGRAPH '82* July 127-133

Cleary, J., Wyvill, B.L.M., Birtwistle, G. and Vatti, R. (1983) "Design and Analysis of a Parallel Ray Tracing Computer" *Proc. XI Association of Simula Users Conference* Paris

Cleary, J.G., Wyvill, B., Birtwistle, G.M. and Vatti, R. (1986) "Multiprocessor ray tracing" (in press) *Eurographics*

Dippe, M. and Swensen, J. (1984) "An adaptive subdivision algorithm and parallel architecture for realistic image synthesis" *Proc. SIGGRAPH'84* July 149-158

Drewery, K. and Tsotsos, J. (1986) "Goal Directed Animation using English Motion Commands" *Proceedings, Graphics Interface '86* 131-135 Vancouver 26-30 May

ElMaraghy, H.A. (1986) "Kinematic and Geometric Modelling and Animation of Robots" *Proceedings, Graphics Interface '86* 15-19 Vancouver 26-30 May

Fournier, A and Reeves, W (1986) "A simple model of Ocean Waves" *Computer Graphics (Proc. SIGGRAPH 86) 20* (4) 75-84

Fuchs, H., Goldfeather, J., Hultquist, J.P., Spach, S., Austin, J.D., Brooks, F.P.Jr., Eyles, J.G. and Poulton, J. (1985) "Fast Spheres, Shadows, textures, transparencies, and Image Enhancements in Pixel-Planes" *Proc SIGRAPH '85 19* (3) 111-120

Goldstein, R.A. (1971) "The System for Computer Animation by 3-D Objects," *Proc. 1971 UAIDE Annu. Meet.*, Stromberg Datagraphix.

Greenberg, D, and Cohen, M and Torrance, K (1986) "The visual simulation of amorphous phenomena" *The Visual Computer 2* (5) 291-297

Hasegawa, T. (1982) "An Interactive System for Modeling and Monitoring a Manipulation Environment" *IEEE Trans.SMC, SMC-12* (3) 250-8.

Henderson, P., Bremser, C. and Lopez, A. (1986) "Understanding the DOMAIN Graphics Environment" Apollo White Paper, Apollo Computer Inc., Chelmsford, Mass.

Hockney, R.W. and Jesshope, C.R. (1981) "Parallel computers" Bristol, England, Adam Hilger Ltd.

Korein, J. and Badler, N.I. (1982) "Techniques for Generating the Goal-Directed Motion of Articulated Structures" *IEEE Computer Graphics and Applications* Nov

Langlois, L. (1984) "Simulation visualization with SIMSEA, a general purpose animation language" *Proc. SCS Conf. on Simulation in Strongly Typed Languages.* San Diego Feb

Lozano-Perez, T. (1983) "Robot Programming" *IEEE Proc, 71* (7) 821-41, Invited paper.

MacDonald, B.A. (1984) "Designing Teachable Robots" PhD thesis, University of Canterbury, Christchurch, New Zealand

MacDonald, B.A. (in press) "Improved Robot Design" *Transactions of IPENZ*, Elect/Mech/Chem section, Wellington, New Zealand.

MacKay, S.A. and Tanner, P.P.(1986) "Graphics Tools in Adagio, A Robotics Multitasking Multiprocessor Workstation" *Proceedings, Graphics Interface '86* 98-103 Vancouver 26-30 May

McLaughlin, J.R. (1982) "TRIG: An Interactive Robotic Teach System" Working Paper 234, MIT AI Lab, June. 53p.

Mandelbrot, Benoit. (1983) The Fractal Geometry of Nature W.H. Freeman and Company. (First Edition 1977).

Mathematical Applications Group (1968) "3D Simulated Graphics" Datamation Feb.

Nitzan, D. (1979) "Flexible Automation Program at SRI" *Proc JACC*, pp 754-9. Denver, June.

Parke, F.I. (1982) "Parameterized Models for Facial Animation" *IEEE Computer Graphics and Applications, 2* (9) 61-68.

Peachey, D. (1985) "Solid Texturing of Complex Surfaces" *Computer Graphics (Proc. SIGGRAPH 85) 19* (3) 11-20

Peachey, D. (1986) "Modelling Waves & Surf" *Computer Graphics (Proc. SIGGRAPH 86) 20* (4) 65-75

Perlin, K. (1985) "An Image Synthesizer" *SIGGRAPH 85, Computer Graphics 19* (3) 287-296

Reeves, W. (1983) "Particle Systems - A Technique for Modeling a Class of Fuzzy Objects" *ACM Transactions on Graphics 2* 91-108

Ridsdale. G., Hewitt, S. and Calvert, T.W. (1986) "The Interactive Specification of Human Animation" *Proceedings, Graphics Interface '86* 121-130 Vancouver 26-30 May

Rosen, C.A. (1979) "Machine Vision and Robotics: Industrial Requirements" in Computer Vision and Sensory-Based Robots., edited by Dodd, G and Rossol, L., pp 3-20. Plenum, N.Y., Symp. held at GM Labs..

Rubin, S. and Whitted, T. (1980) "A 3-Dimensional Representation for Fast Rendering of Complex Scenes," *Computer Graphics (Proc. SIGGRAPH 80)*, July, 110-116.

Smith, A.R. (1984) "Plants, Fractals and Formal Languages" *Proc. ACM SIGGRAPH '84* July 1-10

Smith, A.R., Carpenter, L., Cole, P., Evans, C., Porter, T. and Reeves, W. (1982) "Genesis Demo in Star Trek II: The wrath of Khan" Lucasfilm Computer Graphics Project for Industrial Light and Magic. June

Sutherland, W.R., Sproull, R.F. and Schumacker, R.A. (1974) "A Characterization of Ten Hidden-Surface Algorithms," *Computing Surveys, 6*(1), March, 1-55.

Thalman N. and Thalman D. (1985) "Three Dimensional Computer Animation: More an Evolution Than a Motion Problem" *IEEE Computer Graphics & Applications 5* (10) 47-57

Unger, B., Birtwistle, G., Cleary, J., Hill, D., Lomow, G., Neal, R., Peterson, M., Witten, I.H. and Wyvill, B.L.M. (1984) "JADE: A distributed software prototyping environment" *Proc. SCS Conf. on Simulation in Strongly Typed Languages* San Diego Feb

Upson, C. (1986) "The visual simulation of amorphous phenomena" *The Visual Computer 2* (5) 321-326

Waltz, D. (1982) "Artificial Intelligence" *Scientific American*, October, 78- 83.

Whitted, T. (1980) "An Improved Illumination Model for Shaded Display," *Communications of the ACM, 23* (6), June, 343-349.

Wilhelms, J. (1986) "Virya - A Motion Control Editor for Kinematic and Dynamic Animation"*Proceedings, Graphics Interface '86* 141-146 Vancouver 26-30 May

Wyvill, B (1985) "Current Trends in Graphics and Animation" *Proc. SCS Conf. on Simulation*, San Diego.

Wyvill, B, Liblong, B, and Hutchinson, N (1984a) "Using Recursion to Describe Polygonal Surfaces" *Proc. Graphics Interface 84*, Ottowa, June

Wyvill, B, Neal, R, Levinson, D and Bramwell, R (1984b) "JAGGIES: A Distributed Hierarchical Graphics System" *Proc. CIPS Session 84* Calgary, May

Wyvill G., Wyvill B. and McPheeters C. (1986) "Soft Objects" *Advanced Computer Graphics, Proceedings of CG* Tokyo 113-128

Zeltzer, D. (1982) "Motor Control Techniques for Figure Animation" *IEEE Computer Graphics and Applications* Nov