Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

A.A.M. Kuijk

Temporal issues of animate response

Computer Science/Department of Interactive Systems Report CS-R8960 December

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Temporal Issues of Animate Response

A.A.M. Kuijk

Centre for Mathematics and Computer Science (CWI)
Department of Interactive Systems
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
Email: fons@cwi.nl

Abstract

Due to increased capacities of personal workstations, graphical user interfaces become capable to offer natural human computer interaction. This results in animate response, i.e. natural transitions from one state into another. It is recognized that for optimal efficiency, such animate response should be "tuned" to the time frame of the user. It is not sufficient to speed up the response as much as possible: the temporal characteristics of the response should be based on human perceptual capabilities. This paper is a survey of notions related to animate response. Throughout the paper we will touch subjects which need consideration and/or further investigation.

CR Categories and Subject Descriptors:

I.3.3 [*Computer Graphics*]: Picture/Image Generation -- *Display algorithms;*

I.3.5 [*Computer Graphics*]: Computational Geometry and Object Modelling -- *Curve, surface, solid, and object representations;*

I.3.6 [*Computer Graphics*]: Methodology and Techniques -- *Interaction techniques*

Key Words & Phrases:

continuous motion, temporal anti-aliasing, implicit animation, direct manipulation, hierarchical data structures, progressive refinement.

Introduction

User interfaces are becoming more and more sophisticated due to the tempestuous evolution of both hard- and software. Sophisticated user interfaces support natural interaction and take into account features of human visual perception. This results in *animate response*, that is response where changes as invoked by the user are visualized by a smooth, natural transition from one state into another.

Motion plays an important role in numerous aspects of human vision. Eyes are in a constant state of movement to generate stimuli needed to see anything at all. Relative motion of objects is essential for three-dimensional perception: a change of viewing position generates motion parallax. The human mind is trained to interpret this motion parallax as a measure of distance. A person observing a three-dimensional object which he holds in his hand will almost automatically rotate it to generate relative motion and thereby get the best impression of the three-dimensional structure. The human vision system uses relative motion as a basis of grouping: organizing objects that belong together. By exploiting such acquired perceptual capabilities of the viewer, animate response can improve human computer interaction.

Why should we be concerned about temporal aspects of animate response, isn't it sufficient to simply let the system respond as fast as it can? If nowadays workstations are not fast enough, future systems will be, so why bother about it? The answer to this is that systems might never be fast enough: workstations may increase in capacity, but on the other hand applications become increasingly more complex. More importantly however, to let an interactive environment respond as fast as it can will almost never result in an adequate response. The system will quite often be to slow, but occasionally it may even be to fast.

We should continue to be concerned about temporal aspects of animate response: the system must be "tuned" to the time frame of the user. This tuning should aim at optimal interaction, on the one hand taking into account the limited temporal resolution of the display system and the human vision system, on the other hand taking into account ergonomic aspects: how to help the user anticipate to a new situation. The following two examples of such tuning may be illustrative. They can be found in a type of user interface commonly known as the desktop metaphor, a simple two dimensional environment.

Not all systems are capable of supporting real-time dragging of a window with all of its content. An elegant solution found is to leave the window with its content where it is and let the user drag a simplified representation of the window: its boundary rectangle. Due to this simplification, the system is able to give appropriate feedback to the user.

The visible action upon closing a window, i.e. clearing the screen, can be done very fast. Here, a system can take more time and smoothly lead the user to the new situation by an implicit animation: a shrinking boundary rectangle moving towards a disk or folder icon. During this animation, both the system and the user have time to update their administration.

For animate response in three-dimensional dynamic environments more sophisticated techniques are needed, but basically they are of the same nature: adapt to the users time frame. In this paper we will address some issues related to this.

The outline of the paper is the following. First we will be more specific on what we consider to be animate response. Next, we will visit an area in computer graphics which is closely related, the area of computer animation. Computer animation is an area where the knowledge of how to adapt to the human vision system is essential. Following this we will discuss aspects of visualization of motion given the limitations of a real-time environment. From this, we will see that hierarchical data structures are needed so as to be able to trade time against quality. Therefore, several aspects of hierarchical data structures are discussed. The last section will mention some issues which need further investigation.

Animate Response

As mentioned in the introduction, animate response, is the visual response of a system to changes invoked by the user. It is characterized by a smooth, natural transition from one state into another. It is useful to distinguish between the following two situations: *implicit animation*: the user interface automatically generates intermediate states and *direct manipulation*: the user generates intermediate states by manipulating objects or entities of objects. These two categories differ in who is in control over the transition phase, a difference which has direct implications on temporal aspects of the interaction.

Implicit animation

In the case of implicit animation, explicit specification of a new state implicitly specifies the transition. Although the user interface may allow the user to specify certain transition parameters, the transition itself is autonomously generated by the system. As the visualization is not part of a time restricted feedback loop, the temporal requirements are somewhat liberal.

Implicit animation is a sophisticated type of response, which can be employed by traditional command language interaction: objects are manipulated by means of explicit commands, issued for example by typing or by menu selection. Implicit animation merely helps to "guide the eye", to make the transition appear more natural. It reduces the computational effort of the interaction process, evidently not on the side of the system but on the side of the user. For this, some

intelligence on the system side of the interface is required. Based on the type and amount of the change, the system should be able to determine how to best visualize the transition for the user.

Implicit animation does not have to be restricted to generate a natural transition between two "key-frames", one may also think of more abstract transitions as in the example of closing a window as mentioned in the introduction. In the first state the user sees a window, in the final state the user sees an icon and the abstract transition is used to inform the user which icon relates to the window. Characteristic of abstract transitions is that an abstract relation between two by nature different entities can be visualized.

As an example of non-abstract implicit animation, think of a user looking at an image of a complex molecule. It would help him understand the whereabouts of all the atoms of that molecule if the command "rotate 90 degrees around the z-axes" does not result in a mere presentation of the new situation, but would result in a simulation type of response: an animated sequence of a slowly rotating molecule. Even if this rotation is visualized using a simple wireframe image, the relative motion of the individual atoms during this rotation would give the viewer the depth perception needed to fully comprehend the three-dimensional structure. It may be clear that this simulation type of response needs temporal tuning: it should not take too much time else the user will be annoyed, on the other hand it should not be too fast else the user will miss essential information.

As mentioned before, setting of transition parameters may allow the user to specify the speed, image quality or other aspects of the transition. Sophisticated systems would automatically adapt to the user's skill and preferences. It should be noted that although the temporal requirements may be somewhat liberal, too long transition times would distract rather than guide the user.

Direct Manipulation

Direct manipulation is a very natural and powerful interaction mechanism: the user is able to manipulate a visible representation of a virtual model as if it is a real object which can be seen and touched. The user can manipulate entities of that model by means of input devices like a mouse, a light pen, a trackball or more exotic devices such as a data-glove. These manipulations have to be visualized as a part of the feedback loop between input device, visualized entity, user and again input device. The user will handle the input device based on what he sees. As a result the time restrictions are severe.

Direct manipulation has already proven to be successful in a well known type of user interface: the desktop metaphor. This type of user interface became such a successful entrant in the office environment mainly because a virtual two dimensional environment was created which was a metaphor of an environment already familiar to the user: his desk. It allowed a user to manipulate documents (windows) on a screen just as he was used to manipulate paper documents on his own desk: file them in a cabinet, throw them in a waste paper basket, open and close documents etc. This familiar "look and feel" reduces the learning time which is an important aspect in the office environment.

Similarly, this direct manipulation concept can be extended to three-dimensions to result in a very natural and intuitive user interaction mechanism within three-dimensional and even dynamic environments. It is the key to create artificial realities (also known as virtual environments or cyberspace), in which the user is free to manipulate simulated three-dimensional objects in a natural way, can walk through and interact with imaginary worlds, in short: can be a part of the virtual model.

Based on the direct manipulation concept we can for instance create remote control systems with which a user is able to control remotely in space (hazardous or far away environments), remotely in time (future) or allow control in a different scale (macro- or microworld) [Shneiderman, 1989].

In order to give the user the feeling that he is part of a virtual model and interacts with it, the most important aspect of direct manipulation appears to be direct visualization. If the result of the manipulation is not visualized immediately, the user will lose contact with the environment. As Scott Fisher stated [Fisher, 1989] "the quality of the graphics appears not to be extremely important, as long as the response is good".

Summary

In the area of animate response we could distinguish two classes, implicit animation and direct manipulation. In the class of implicit animation, the system is in control of the transition phase. As a result, intermediate states are known beforehand and in principle the temporal behavior is predictable. In the class of direct manipulation however, the transition is controlled by the user. As a result the intermediate states are unknown and the temporal behavior is unpredictable. However, the impact of temporal requirements can be generalized to animate response as a whole.

What can we learn from Computer Animation?

Why is a film, a long band of *static* images, often called a "movie"? Answering that question also gives an answer to why computer animation is at all possible, using video displays which can only display static images. Apparently it is possible to give a viewer an impression of continuous movement by successively presenting displaced static images (frames). Given this, at what rate should these frames be generated? Intuitively it is clear that the frame rate has to be related to the speed at which the objects move. Since video displays have a refresh rate of about 100 Hz at maximum, we find ourselves restricted in frame rate as well[1]. This limitation has effect on visualization of fast moving objects. Due to their fast movement, the displacement of objects between two individual frames becomes quite large. When these object are displayed just as if they were static, the perception of the spatial relation between images is lost: the impression of watching objects in continuous motion is no longer there. A movie projector however, shows separate frames at a rate as low as 24 frames per second. Even this rate is sufficient to generate an impression of continuous motion. Why this is so can been found by looking at individual images of a fast moving object on a camera film. There we see that the finite exposure time of the film results in blurred images of the fast moving object. Due to this motion blur the impression of continuous movement is maintained, even at a rate of 24 frames per second. From this it is concluded that the human vision system can trade off temporal resolution against spatial resolution.

A number of papers present methods to generate motion blur (also known as temporal anti-aliasing) [Korein, 1983; Potmesil, 1983; Max, 1985; Grant, 1985], but arguments why it is needed do not go any further than "animation which simulates motion blur feels more natural" or "it smooths out jerkiness". A more theoretical foundation of why and to what extend motion blur is needed can be found in E.H. Blake's PhD thesis on computing adaptive detail [Blake, 1989]. The next subsection is a short summary of relevant issues from this thesis.

A Viewer Centered Metric for Computing Adaptive Detail

Blake's study aims at reducing the computational complexity of computer animation by computing just what is needed to produce convincing pictures. The thesis is centered around two metrics, a spatial (static) priority: "objects further away from the viewpoint are visually less important to the picture than those closer by" and a temporal (dynamic) priority: "objects moving quickly with respect to the observer need to be redrawn more often than those at rest". These initial intuitive formulations of the two metrics are further developed and extended with the notion that human vision introduces a trade-off between temporal and spatial resolution. These are the basics for arriving at measures to determine to which extent detail can be left out without being noticeable by a viewer.

Changing images are a function of two space variables and of time. A Fourier transformation of this function results in a spectrum, a function of temporal and spatial frequencies. This function is non-zero in a limited domain only. The effect of motion on this spectrum is that it is sheared in the temporal frequency dimension. The amount of shear is proportional to the velocity of the object components in the two dimensional image.

---

[1]Note that *frame rate* is the rate at which successive and thus different frames are displayed. This is not to be confused with the *refresh rate*, which is dictated by the persistence time of the display medium. The frame rate can never exceed the refresh rate, but one frame may last for several refresh cycles.

The human vision system is limited in spatial and temporal frequencies by the various transmission systems between our mind and the outside world, resulting in a so-called window of visibility [Watson, 1986].

Since the spectrum of moving objects is sheared, some spatial frequencies which would fall inside the window of visibility for static objects, may fall outside this window when these objects are in motion. Removing the spatial frequencies that fall outside the window of visibility is equivalent with removing invisible detail of the object. However, this is exactly the detail that in combination with the frame rate of the display system appears in aliased form, causing the unnatural or jerky motion mentioned above.

Blake remarked that such Fourier analysis is useful for analyzing a problem and obtaining insight into what is happening. However, the computations and algorithms are less likely to use Fourier techniques directly.

Blake argues that optic flow analysis as introduced by Gibson [Gibson, 1979] can be used to obtain a measure of the velocity and/or distortion of object components in the image. This measure can be used to determine the amount of detail needed, in other words: which spatial frequencies have to be left out, i.e. to what extend the image should be motion blurred.

For planar objects, the recognition of four orders of optic flow effects results in four orders of frame to frame coherence:

- no relative movement
- translation in the image plane
- linear transformation in the image plane (shear, rotation, scaling)
- non-uniform motion, too high distortions.

Computer Animation versus Animate Response

The major concern of computer animation is to convince the viewer, to make him see what he is meant to see. In general, images are rendered at the highest quality possible. For real-time animation systems such as flight simulators this requires a major exertion which can be maintained by specialized systems and for pre-defined situations only. Off-line computer animation involves three main tasks: modeling, animation (i.e. the design of the choreography) and rendering. These tasks, are iterative design processes. As far as these design processes are supported by an interactive editor, the interactivity involves just a minor facet of the animation. This type of animation production is a computational intensive process which in general takes orders of magnitude more time than the real-time display of the animation itself.

This is quite different from what is needed in an interactive environment of which the main device is: "the quality of the graphics is not extremely important, as long as the response is good". In our situation the main objective is real-time display of continuous motion, even if we have to deal with complex structures. There is not just one, but two temporal limits that complicate the real-time display of continuous motion. These limits are the maximum frame rate of the display system as well as the capacity of computing resources.

Conclusion

What we learned from computer animation is how to act upon temporal limitations as enforced by the display system and the human vision system. We also learned that in computer animation production, the limitation of computing resources is traded against production time.

In order to account for limited computing resources, all we can do for animate response is trade against image quality, i.e. simplify the image generation. Unfortunately, we have to simplify the image generation more than just by leaving out detail which is not noticeable by a viewer, yet we like to present the best possible image. From the example mentioned in the introduction --manipulation of a window boundary instead of manipulation of the window itself -- we learned that even to such extent, a trade off between image quality and system response time can often be quite adequate.

For animate response, we would have to have a metric for the level of detail related to the computing resources available. The implicit dependency as presented in Blake's work should be made explicit.

How the limitation of computing resources effects visualization of continuous motion will be discussed in the next section.

Visualization of Motion

As became clear in the previous section, for animate response there are two temporal limits that complicate the visualization of continuous motion. One is the limit on the frame rate as dictated by the display system, the other is the limited capacity of computing resources. The impact of these two limitations is the same: the frame rate needed to display continuous motion is not sufficient, so that visible temporal aliasing may occur[1].

Temporal Anti-aliasing

Can, whenever the frame rate introduces temporal aliasing, one of the temporal anti-aliasing methods used for computer animation be of help? If we analyze the methods presented in [Korein, 1983; Potmesil, 1983; Grant, 1985] we see that these methods are basically supersampling techniques, due to which the computational cost of one anti-aliased frame is much more than the cost of an aliased frame. Also the more efficient method presented in [Max, 1985] adds computation cost to each frame.

In case the computing resources can be considered to be unlimited with respect to what is needed by the application, we are in fact dealing with a real-time animation system. In this situation, we only have to face the limit as set by the maximum frame rate of the display system and the supersampling methods used for computer animation might be useful.

In an interactive environment, however, it is much more likely that complexity of scenes and at the same time real-time requirements turns out to be a combination that makes us face the limitation on computing resources. In this situation we cannot allow any extra computational effort because it would reduce the frame rate even further. It is remarkable that most common temporal anti-aliasing methods used for computer animation production, produce images containing less detail (since the high spatial frequencies are filtered out) at a cost higher than the images containing full detail. From this it is clear that these methods are not useful. If we are limited by computing resources, we should have a method that saves rather than adds computational costs.

Reducing the computational costs can be done by reducing the image quality (i.e. complexity), yet we would like to generate the best possible image. In order to do this, we need an adaptable image generator.

Adaptive Image Generation

In computer graphics, the image generation process always has to be tuned to the specific needs of the application. Even when the image generation is a batch mode process performed by the most powerful supercomputer, certain trade-offs have to be made. If we were at all to know how to generate a physically perfect image, it would by far exceed the processing power of any state of the art supercomputer. Because of this, a whole scale of rendering models emerged, each with specific features and each of which tries to approximate a physical correct image to a different level. Based on the requirements of a specific application, one of these rendering models can be selected. The fact that worst case situations also have to be considered, quite often leads to a choice which in practice is far from optimal.

A more sophisticated approach that avoids this problem, is adaptive image generation. By adaptive image generation, the application is able to adjust the image generating process to the specific needs of a particular moment. Ideally, this results in the best possible image at any time. Globally speaking, there are two ways in which the image generation process can be adapted: based on the rendering process or based on the object representation. We will elaborate on this, in order to find out to what extent this can be used for animate response.

---

[1] In fact no matter what discrete frame rate is used, always temporal aliasing will occur. However, it becomes visible only when the alias frequency falls within the window of visibility [Watson, 1986].

A method based on an adaptable rendering process was suggested by Forrest [1985]. Rendering of primitives should be supported at different quality levels. As an example, Forrest recognizes five quality levels to draw a line, starting with an aliased Bresenham line drawing, up to a perfect anti-aliased line of which even the line ends can be specified (e.g. rounded or square). He suggested that such a hierarchy in quality could be exploited in the context of personal workstations. Images should first be rendered at the lowest quality level to get the fastest response and upgraded to higher quality levels if the user does not take immediate action.

Another example of a method based on adapting the rendering process, is image rendering by adaptive refinement [Bergman, 1986]. Similar to what Forrest suggested, this method improves the quality of a static image as long as there is time to do so. The image successively goes through the following phases: display vertices of polygons, display edges of polygons, display flat shaded polygons, add shadowing, display Gouraud shaded polygons, display Phong shaded polygons and finally anti-alias the image where needed. The performance of the method is enhanced by each phase making use of results of the previous phase and trimming of the data, i.e. selecting which of the polygons should be handled by the next phase.

The interest of these methods is the combination of interactive response and the generation of the best possible image by exploiting different levels of rendering. Both methods however, become effective only in a static situation, making use of otherwise idle cycles in a personal computer. Since the rendering always starts at the lowest level, it is likely that when displaying continuous motion these methods will produce images of the lowest level only. Also, rendering of a certain level will at least partly undo the results of previous levels. For instance, each phase will have to overwrite pixels resulting from previous phases. At a time-scale of a few seconds, this may not be a point of concern, but at frame rates needed to display continuous motion, the actual writing of pixels becomes a dominant factor. Therefore these methods could be made more efficient if we would have a way to determine which level of rendering can be supported from the start.

Methods based on object representation adapt the rendering costs by adapting the complexity of the scene description. Hierarchical data structures [Clark, 1976], in which sub-hierarchies contain objects modelled in greater and greater detail, are used to render objects at different levels of detail. Such hierarchical structures allow for a simple incremental approach: as long as there is time, objects can progressively be refined. This can be improved by combining it with the metric described by Blake [1989]. This metric, which takes into account the viewing distance and speed of the objects, can be used to indicate which objects are best candidates for further refinement. The effectiveness of this type of adaptive image generation is quite dependent on the hierarchical data structure, since properties of the structure determine to what extend increments can make use of previously obtained results.

Adapting the complexity of the scene description seems to be best suited for an incremental image generation method. In the next section we will focus on hierarchical data structures that can be used for this purpose. We note however, that ultimately a combination of adaptive scene complexity and adaptive rendering should be considered.

Hierarchical Data Structures

Since we have to be able to deal with limited computing resources, we have to be able to manage the image generation costs. This managing can be provided by exploiting hierarchical data structures. Hierarchical data structures allow definition of a so-called graphical working set [Clark, 1976] which is that fraction of the structure that at a certain time [Hegron, 1987] is potentially of interest. This notion has been put into practise in flight simulator applications for a long time. There, the most appropriate representation of an object is selected from a hierarchical data structure at display time. Such data structures are carefully optimized for the application.

Numerous hierarchical data structures in which objects are modelled in greater and greater detail, have been proposed. To select candidate data structures the following issues should be considered: automatic generation of the hierarchy, motion, relation between level of detail and the cost of rendering, temporal anti-aliasing and texturing.

Automatic Generation. Not all object representations have an inherent hierarchical structure. In such a situation we need a hierarchy that can automatically be generated. Recursive spatial subdivision seems an obvious way to automatically generate an oct-tree like hierarchical structure. Although the result is a simple uniform representation, rendering of such a hierarchy is not very efficient and the representation is not compact. Furthermore, transformation of objects will require restructuring of the hierarchy. As a result such hierarchies are not particularly suitable for animate response.

Rubin & Whitted stated: "creation of a hierarchical database is a non-trivial operation" [Rubin, 1980]. They presented a homogeneous representation from which nodes of the hierarchy are procedurally generated. Their representation is a graph structured hierarchy of nothing but bounding volumes. Each bounding volume contains subspaces, which are the bounding volumes of the next level in the hierarchy. Their representation allows sharing of subspaces. Increment of one level in a hierarchy of bounding volumes implies that a volume, being the difference between the bounding volume and its subspaces, has to be subtracted. This subtraction would reveal previously obscured parts of the scene, which would imply a redo of (a part of) the visibility calculation. It would be much more efficient to have a hierarchy that incrementally adds volumes, leaving the rest as it was. The proposed sharing of subspaces greatly reduces the amount of data storage but introduces combinatorial problems when creating the hierarchy. Since an optimal result requires careful consideration, Rubin & Whitted concluded that it would benefit to off-load the structuring of the hierarchy to the model creation stage. Rubin [1982] expanded the representation to allow for geometrical transformations between nodes. Such an expansion is extremely welcome in an animate environment, because it facilitates the hierarchy to represent moving objects.

Detail versus Cost. In order to optimally balance image quality and temporal aspects of the animation, the level of detail should not only be based on the visual aspects, but also on the actual costs of the image generation. For implicit animation in particular, it would be desirable to have an absolute quantification of the rendering cost as function of the level of detail. With this, we could determine the maximum level of detail for which the animation can be sustained. To obtain an absolute quantification of the costs however, would require an extensive benchmark of the actual rendering hardware used. Another complication that has to be dealt with is that the complexity of the scene is likely to be view dependent. How to obtain such an absolute hardware dependent quantification and how to embody this knowledge in the hierarchy is a question that remains open.

A requisite of a data structure in case absolute quantification cannot be supplied is that the cost to render objects should be more or less proportional to the level in the hierarchy. Adding one level of detail should not undo all results obtained so far, but instead make optimal use of previous calculations. Ideally, in this way, rendering by incremental addition of subsequent levels should at most be as expensive as directly rendering at that particular level of detail.

Temporal Anti-aliasing. How well can data structures which contain object descriptions in levels of detail support temporal anti-aliasing? As we saw in the above, temporal anti-aliasing is equivalent with removing high spatial frequencies above a certain cut-off frequency. This cut-off frequency is inversely proportional to the speed of an object. An object can be described by a sum of band limited terms of increasing frequencies, a representation which can for instance be obtained by Fourier analyses. The effect of temporal anti-aliasing on such representation would be that the faster this object moves, the less terms would remain. This behavior may seem ideal since then the rendering costs would be inversely proportional to the speed of the object. Unfortunately however, an object description by a sum of band limited terms is in general not the most efficient. It is not a natural representation of objects, so that Fourier transformation is needed. Also the rendering of such a representation is complicated. In a static or near to static situation, a description of even the simplest object would at least need terms up to a frequency equivalent with the display resolution, otherwise visible detail would be lost. As a result, whether a description by band limited terms pays off is dependent on the extent to which Fourier transformation simplifies temporal anti-aliasing and complicates rendering. It is likely to pay off only in situations where any other representation would be of similar complexity, such as may be the case for textures.

Textures. For textures expressed as a sum of band limited terms, Norton et.al. [1982] described a method of limiting texture detail by "clamping" those terms that exceed a certain frequency. Since the authors just considered spatial anti-aliasing, their clamping frequency is fixed as it is determined by the display resolution only. This method can be extended for temporal anti-aliasing if the clamping frequency is related to the speed of the object. Other spatial anti-aliasing methods for textures exploit hierarchies of various resolution texture maps [Crow, 1984; Glassner, 1986]. Temporal anti-aliasing of such textures can be done by selecting the appropriate resolution map, again dependent on the speed of the object. However, for animate response, incremental image generation is of more concern than temporal anti-aliasing. It may be clear that in this sense textures expressed as a sum of band limited terms are favorable, since they are inherently incremental. This as opposed to textures represented by various resolution maps, since the costs of texture mapping is independent of the texture density, so that reducing texture resolution will not reduce the cost of the image generation.

We conclude that at this moment there is no unified structure which serves all needs. We noted that for rendering optimal structuring is a requisite so that automatic generation of hierarchies is not likely to be a real-time process. Procedural hierarchies appear to be very attractive, especially when they feature sharing of sub-hierarchies and transformation between nodes. Since obtaining an absolute measure of the cost of rendering is a problem, requisite of a data structure is that increment of one level in the hierarchy should result in a simple incremental calculation. We did not find an efficient structure that does support temporal anti-aliasing in a general way. A special situation is representation of textures by band limited terms. This representation can easily be temporal anti-aliased and also has the advantage that rendering costs relates to detail.

Research Topics

From the above, it seems that from a technical point of view, basic knowledge and techniques to implement a human computer interface which supports animate response is already there. However, from a human perceptual point of view, adequate knowledge of how to exploit cognitive skills is still lacking. This knowledge is especially of interest for implicit animation, which also incorporates more abstract types of response. Especially the more abstract type of response leads to interesting issues such as the concepts of illusion, the mechanism by which users are able to associate cues with their own experience and the issue of appropriate abstractions.

For this we would have to develop a human computer interaction environment which forms the basis for further research on implicit animation. At first we ought to restrict ourselves to an interactive system for dynamic exploration and manipulation (which includes editing) of inherently static models. The implementation would be targeted at future low cost workstations.

The development of the system should emphasize on data structuring. As became clear in the previous section, the internal data representation has to fulfil requirements based on visual and temporal aspects of animation. The data structure should preferably be a procedural hierarchical scene description of which all elements contain information how they should be rendered [Cook, 1984].

A straight forward method for temporal anti-aliasing is by simplifying the object representation to such extent that ample time remains for a conventional temporal anti-aliasing method. A data structure has to be found which offers a more general and efficient solution.

It should be noted that the internal data representation should match the hardware characteristics and vice versa. For instance, it makes quite a difference whether or not it is a parallel system and if so whether the system exploits image space or object space parallelism. Vice versa, if incremental calculations and temporal anti-aliasing has to be supported, it is unavoidable that pixels will have to be addressed multiple times and results will have to be accumulated. Multiple access per pixel in a real time environment can only be solved by massive parallelism.

For the external world (i.e. the user) the data representation should be naturally presentable and manipulable. This is a requirement which potentially conflicts with requirements of the internal data structure. It is not likely that one unified data structure can be found. Object oriented

methods however, provide the necessary data abstraction to be able to hide the internal details of a hierarchy implementation.

More on technical aspects. The first prototype system should exploit a double buffered frame store. Frames can be swapped at a constant frame rate. The invisible frame should be incrementally improved as long as there is time.

This basic system can be optimized in several ways. A first optimalization would be to make use of the metric for computing adaptive detail as proposed by Blake [1989]. This metric is based on two visual aspects, i.e. speed of the object and (weighted) viewing distance. In our case, this metric would not be applied to determine the absolute level of detail at which objects should be rendered, but would be used to give priority to the elements of the graphical working set. This priority will be used to selectively increment the level of detail of those elements that will likely improve the image quality most. A complication that has to be dealt with is that the priority is likely to change during the animation.

Making optimal use of frame coherence is an essential optimalization factor in interactive environments. In its simplest form frame coherence can be exploited by partially updating the image. The image has to be redrawn only in the area where the image is known to be changed. A more elaborate way to exploit this time based coherence is by manipulating the frame rate. As we saw in the above, the frame rate needed to display continuous motion should be related to the speed of the object components in the image. Since a movie projector is operated completely time independent of the camera, it is obvious that the simplest way to synchronize both mechanical devices --by operating at a constant frame rate-- is chosen. In our situation however, we do not have to deal with mechanical devices and there is a tight coupling between the image generation system and the image display system, so that synchronization is relatively easy. This gives us the opportunity to vary the frame rate, related to the speed of the object components in the image. Even more so, each object might have its individual frame rate.

As we saw in [Blake, 1989], the use of frame coherence does not have to be restricted to frames or parts of frames that do not change in time. Certain types of uniform 3-D motion can be approximated in a mere translation or a linear transformation (shear, rotation & scaling) of the object components in the image plane. Exploiting this more complex form of frame coherence might be worth considering. It should be noted however that in general 3-D motion has effect on shading. This will restrict the extend to which such translation or transformation of images can be used.

An incremental system as described is self adapting, given the real-time requirements it automatically generates images of the maximum obtainable quality level. A less desirable side effect of this solution is that varying complexity of the scene during the animation, may cause disturbing switches of level of detail. Since for implicit animation time restrictions are somewhat liberal, we can trade time for detail. It needs to be investigated to what extend it is acceptable to fiddle with the temporal behavior of the animation (change speed) to avoid disturbing changes of level of detail.

Conclusion

Ideally a graphical user interface should be a real-time animation system. However, we saw that limitations of computing resources enforce an approach by which image quality is adaptively reduced to be able to fulfil real-time requirements. To make this possible in a sufficiently flexible way, the need for hierarchical data structures was recognized. Such an hierarchical data structure should on one hand shield off and on the other hand exploit the hardware characteristics. Although the basic concepts of such structures are known, a unified structure which satisfies both internal as well as external requirements could not be found.

Acknowledgement

The author gratefully acknowledges the useful comments and suggestions given by E.H. Blake.

References

[Bergman, 1986] Bergman, L., Fuchs, and H., Grant, E., *"Image Rendering by Adaptive Refinement"*, Computer Graphics (**20**, 4) July 1989 pp 29-37

[Blake, 1989] Blake, E.H., *"Complexity in Natural Scenes: A Viewer Centered Metric for Computing Adaptive Detail"*, PhD Thesis. Queen Mary College, London 1989

[Clark, 1976] Clark, J.H., *"Hierarchical Geometry Models for Visible Surface Algorithms"*, Comm. of the ACM (**19**, 10) October 1976 pp 547-554

[Cook, 1984] Cook, R.L., *"Shade Trees"*, Computer Graphics (**18**, 3) July 1984 pp 223-231

[Crow, 1984] Crow, F.C., *"Summed-Area Tables for Texture Mapping"*, Computer Graphics (**18**, 3) July 1984 pp 207-212

[Fisher, 1989] Fisher, S., Panel *"Virtual Environments and Interactivity: Windows to the Future"* SIGGRAPH'89 Boston, August, 1989

[Forrest, 1985] Forrest, A.R., *"Antialiasing in Practice"*, in NATO ASI Series, editor Earnshaw, R.A. (**F17**) 1985 pp 113-134

[Gibson, 1979] Gibson, J.J. *"The Ecological Approach to Visual Perception"* Houghton Mifflin co, Boston, 1979

[Glassner, 1986] Glassner, A.S. *"Adaptive precision in Texture Mapping"*, Computer Graphics (**20**, 4) July 1986 pp 297-306

[Grant, 1985] Grant, C.W., *"Integrated Analytic Spatial and Temporal Anti-Aliasing for Polyhedra in 4-Space"*, Computer Graphics (**19**, 3) July 1985 pp 79-84

[Hegron, 1987] Hegron, G., *"Dynamic Management of 3D Scenes"*, EUROGRAPHICS'87, Editor Maréchal, G. August 1987 pp 529-542

[Korein, 1983] Korein, J., and Badler, N., "Temporal Anti-Aliasing in Computer Generated Animation", Computer Graphics (**17**, 3) July 1983 pp 377-388

[Max, 1985] Max, N.L., and Lerner, D.M., *"A Two-and-a-Half-D Motion-Blur Algorithm"*, Computer Graphics (**19**, 3) July 1985 pp 85-93

[Norton, 1982] Norton, A., Rockwood, A.P., and Skomolski, P.S., *"Clamping: A Method of Anti-Aliasing Textured Surfaces by Bandwidth Limiting in Object Space"*, Computer Graphics (**16**, 3) July 1982 pp 1-8

[Potmesil, 1983] Potmesil, M., *"Modeling Motion Blur in Computer-Generated Images"*, Computer Graphics (**17**, 3) July 1983 pp 389-399

[Rubin, 1980] Rubin, S.M., and Whitted, T., *"A 3-Dimensional Representation for Fast Rendering of Complex Scenes"*, Computer Graphics (**14**, 3) July 1980 pp 110-116

[Rubin, 1982] Rubin, S.M., *"The Representation and Display of Scenes with a Wide Range of Detail"*, Computer Graphics and Image Processing (**19**) 1982 pp 291-298

[Shneiderman, 1989] Shneiderman, B., *"Future Directions for Human-Computer Interaction"*, Summer School on User Interfaces, Tampere, Finland, June 1989

[Watson, 1986] Watson, A.B., Ahumada, A.J. (jr), and Farrell, J.E., *"Window of Visibility: A Psychophysical Theory of Fidelity in Time-Sampled Visual Motion Displays"*, J.Opt.Soc.Am.A (**3**, 3) 1986 pp 300-307