REAL TIME ANIMATION TECHNIQUES WITH MICROCOMPUTERS

Frank Dietrich

Pixel Creations
731 West 18th Street
Chicago, Illinois 60616

Animation movies are back in business. For a long time this genre was almost identical with Walt Disneys' cutely drawn animals and fairy tale characters. Now it is the Disney Studios again making the breakthrough into the animated space age of computers with a new creation TRON. The computer generated special effects of many other current films make headlines in the movie critiques. The type of computer animation possible with low-cost machines with less resolution, less colors, less speed, ... the list of deficiencies could easily be extended, are a far cry from these expensive and very sophisticated visual innovations. Nevertheless, as will be demonstrated throughout this article, the small graphics computers have strong features for animation used by industrial and educational video productions as sell as in the exploding world of cable TV.

TRADITIONAL & COMPUTER ANIMATION

Animation is the art of changing images in time. This is faster said than done. Traditional film animation requires many tedious and time consuming steps to produce the numerous drawings necessary for sophisticated motion effects. Computer animation can save labor by automating some of these tasks like cel inking or inbetweening. Computer Assisted Imagery (CAI) can even produce complete movies. But most computers do not produce real-time animation. Instead they generate one still image at a time. The illusion of motion is invoked later, as in traditional animation, when complete series of stills have been filmed and are projected at 24 frames per second.

Real-time animation has been possible in the past only with vector refresh displays, mainly because the computer could calculate the new endpoints of relatively few vectors forming the image during the refresh cycle of the display device. Much less picture information has to be processed in such vector systems than in a raster system, where the entire screen memory containing up to millions of pixels (picture elements) has to be rescanned for display. Changing parts of this amount of such large memory has been untill now too slow to allow for real-time animation. Only modern high speed computers with dedicated image processors can accomplish this task in the short time of a 30th of a second.

In this respect it is a surprise that microprocessor-based video game machines are highly interactive and capable of real-time animation. Their main limitation, low resolution, becomes a feature: less pictorial information has to be moved around. Even if this motion appears to be somewhat crude, it is nevertheless real-time animation and serves the interactive purpose.

The animation techniques discussed here were executed on a low-cost micro-graphics system, the Datamax UV-1, initially developed from a video arcade game computer. Its RAM was expanded to 32K, and 256K of screen memory were added, yielding a 320*200 pixel resolution with 16 2-bit 'thin' framebuffers. ZGRASS is the hi-level graphics programming language of the system, ready to go in 32 K of ROM. An internally generated standard NTSC signal puts the imagery from the computer right into the heart of the video world. Even though all this accounts for a specially configurated micrographics system, many of the animation techniques are commonly used and thus serve as general examples for getting a maximum of motion out of small computers.

ADDITIVE CHANGES

Two different ways exist to change an image, one, by drawing new graphics (or erasing existing ones for that matter), the other by changing previously created and stored images. Of course the later which can be either a change of color or a

PROCEIDINJ FOR SMALL COMPUTERS IN THE ARTS 71
82CH1831-7/82/0000/0071$00.75 © 1982 IEEE
PHIL. CONFERENCE - OCT 82

switch to another image plane happens much faster and therefore is better suited for real-time animation. In terms of how much information is changed by drawing, the least demanding animation technique simply keeps writing one graphic element over the previous ones, thus building up the image. This amounts to a gradual change of the image, where the order of introduction of the picture objects significantly contributes to the meaning produced. In some respect this is like story telling with pictures as an ongoing process. And, as in stories, one new element can drastically turn the already known facts around.

There are two flaws with additive changes. This technique can only produce gradual changes of an image and the viewer has to watch the drawing of the new graphic, which can divert the attention to a very insignificant action like filling in an outlined shape. Computer systems featuring at least two image planes offer help: With a technique called "Double Buffering" the immediate action of drawing can be hidden by always showing the image in the other buffer, where no drawing takes place. Once the drawing is finished the program switches to the plane displaying the new image and continues to draw onto the plane which is now not shown. In this manner the animation effect is concentrated on the images changing in time without having to show how the images are being drawn.

```
SHOW plane1; DRAW to plane2
SHOW plane2; DRAW to plane1
```

COLORMAP ANIMATION

The fastest method of changing a raster image is known as colormap animation. Since no drawing is done, the bitmap, storing the digital information of the screen, is not touched at all. Instead, only the representation of color for each given combination of numbers is altered. Since this is a minimal computation it can be quickly accomplished. Just consider the difference between changing only four color values to calculating up to 320*200 pixels. A simple but effective trick like the blinking of a particular object is done by switching back and forth between its color and the background color. Continuous motion (e.g. bloodstream, waterfall) can rather easily be visualized, if a series of colors is switched.

```
TEMPCOLOR = COLOR 1; COLOR1=COLOR2;
COLOR2=COLORn; COLORn=TEMPCOLOR
```

A more elaborate version of blinking is the creation of multiple frames in one image plane. Essentially it is a trade-off between color resolution against multiple images which are initially hidden in the background and then consecutively switched on to create the illusion of motion.

```
COLOR1...COLORn=BACKGROUND
COLOR1=FOREGROUND;COLOR1=BACKGROUND
          ...          ;     ...
COLORn=FOREGROUND;COLORn=BACKGROUND
```

SNAP ANIMATIONN

Numerous animation techniques are possible given the ability to store parts of the screen in a special image array. The DATAMAX UV-1 features a custom designed chip to handle the activity of storing and displaying images up to a quarter of the total size of the screen. Such a memorized image is called a SNAPSHOT or SNAP for short. The same effect can be done with other systems by storing images into another frame buffer for instantaneous retrieval.

The simplest method of animation is to make a SNAP and then continuously display this image along a computed path.

```
        SNAP name,xcenter,ycenter,
             width,height
1MOVE   DISPLAY name, xcenter, ycenter,
             displayoption
        xcenter=xcenter+offset;
        ycenter=ycenter+offset;
        IF Xcenter AND ycenter<
            screenborder,GOTO 1MOVE
```

The display option defines how the SNAP information is logically combined with the existing values in that particular part of the screenmemory. A total of 150 different display options is available: PLOP, OR, XOR, AND, etc. are the choices for the BOOLEAN operation to be performed. Additional color filters can be applied to determine which colors would be effective. For instance, one useful application of appropriate display options would be to let a red car drive behind a green house, disappear and become visible again.

A special trick called the "Difference Snap" relies exclusively on these display options.The difference snap deals with the problem of having to execute two drawing commands to move an object once: first the image has to be erased at the old location and then it must be displayed at the new location. Not only does this take more time, it also causes a disturbing flashing visually interrupting the movement.

The "Difference Snap" combines erasing and displaying into one single action. Its name describes what it is: the visual difference between a snap at location X1 and X2, created by XORing the snap onto itself while it was offset by exactly the amount of pixels it should move. The visual result becomes the difference snap and is stored. Now the movement can be achieved by displaying the original image only once and then continuing to XOR the difference snap with the pre-defined offset. Memory considerations restrict this technique, since two snaps are needed and the direction of movement is not variable.

SEQUENCING

By now we know how to move static images across the screen. In order to change the image itself the snaps have to be sequenced. This technique consists of pre-storing and playing back a series of snaps. First a number of snaps are created, each slightly different than the previous one. These images are labeled in series PIC1, PIC2,..., PICn. Again, memory constraints allow only a limited number of snaps of a limited size to be stored. The animation takes place as a simple sequencing through the snaps much like flipping through a flipbook. In this case the PLOP display option is used, in order to completely cover the old image information with the next one.

```
DISPLAY PIC1, x,y,PLOP
DISPLAY PIC2, x,y,PLOP
          ...
DISPLAY PICn, x,y,PLOP
```

The particular effect achieved depends on the chosen approach. One approach is a sequence in the tradition of Disney animation where two keyframes (or extremes) for each cartoon character are drawn first. Then a number of inbetweens are drawn to create a smooth transistion from one extreme to the next.

A pseudo rotation around either the X or Y axis can easily be created by scaling down either the width or the height of the snap until only a thin line remains. Along this scale subsequent new snaps are made and later played back in sequence. The net result of this technique is a rotation without any time consuming computation of the sine and cosine functions which are usually employed.

MULTI-PLANE ANIMATION

Similarly, the 16 planes of the UV-1 can be utilized for full screen animation by cycling though a series of previously designed images. This is an important extension of displaying graphic objects in snaps both in terms of the size of the image changed and the decrease in time this takes. The switching from one plane to the next (memory bank switching) is so fast, that the eye can not detect the switch. In addition to multi-plane animation, the UV-1 hardware has two other important features: video-digitizing as a source of input and programmable arrangements of 16 screens either into a panorama of X by Y planes or into a stack of 16 planes, one behind the other.

INTERPOLATION

Since the very beginning of computer animation various mathematical forms of interpolation were established mainly to produce keyframe animation of cartoon characters. Basically the interpolation algorithms are set up to divide a range formed by two given numerical poles into a specified number of intervals, thus producing intermediate values in between. The distribution of the intervals and the numbers of interpolated values depends on the function (e.g. linear, sinusoidal, etc.) and the step factor used. Both determine the number of intervals and whether they are constant (linear) or not (sinusoidal). Interpolation algorithms as such are so general that they can be applied to almost any element of the animation sequence: distance , size, angle, time, etc.

Interpolation techniques played a prominent role in a recent computer installation entitled DO-IT YOURSELF MACHINE ART by Joanne Culver, Zsuzsa Molnar and myself. The DATAMAX UV-1 was programmed in such a way that two participants from the audience could create instant animation. Each of them drew one keyframe into plane 1 and plane 16 respectively. Then the computer took over, interpolating the in betweens necessary to transform the picture drawn by the first player into the picture by the second one. After finishing the drawing job the computer cycled through the planes creating a short animation sequence.

FIELD MOVES

Most animation techniques discussed so far share one commonality which can easily turn into a major problem. They require memory space set aside for the storage of images. Fortunately another technique exists that does not need one single bit more image memory to move fields as large as the entire screen.

ZGRASS provides two graphic commands facilitating this welcome option, SCROLL AND WRAP. Both enable the move of an entire section of the bitmap to a different location (memory block move).

Both commands require the specification of X and Y location, width and height of the field, X and Y direction of the move and finally the display option. The main difference between the two is that SCROLL simply moves the designated field to a new location, automatically clipping it along the screen boundaries if necessary. WRAP performs a wrap-around function to the image information contained within the area, but displays it again at the same location.

Both techniques lend themselves very well to preparing animation sequences for later playback as well as copying large areas. But the drawing of these techniques is such a visible interruption , that it should be hidden by double buffering (as described earlier) to successfully utilize these field moves. Sometimes an artifact of WRAP can be turned into a visually interesting result. Since WRAP simultaneously executes both READ and WRITE functions, the image that appears during the WRAP as seen as though through venetian blinds in motion.

TIMING AND CONTROL

Finally the question of timing and rhythm must be addressed, since a high degree of control over time is absolutely necessary for animation. Unfortunately the option of greater speed is limited. Only two possibilities presently exist: to compile a program or to streamline it. The latter can be done, for instance by calculating lookup tables of moves defined by complex functions thus stripping down the computation during the animation to the bare bones of retrieving those values.

Slowing down the animation is much easier and offers more options. A WAIT command counts in seconds helpfully slowing down the action. System timers which need to be set only once and then count down to zero become helpful for triggering a new event after completion of a previous one.

For choreographing a number of simultaneous events parallel processing comes in handy. ZGRASS offers three different execution modes: normal, foreground and background. The normal mode executes a program only once and then returns. Background and foreground modes execute the program until explicitely stopped. The difference between them is that programs running in background mode are executed one line of code at a time whereas macros in foreground mode are executing a line of code each 60th of a second.

Features like this make use of the extendability of the graphics language to allow the development of libraries of special purpose routines. They also support the task of combining a handful of different events into one complex animation. A controller program for such a sequence could look like this:

```
READJOYSTICK.FOREGROUND
MOVESHIP.FOREGROUND
CHANGECOLORS.BACKGROUND
MAKESOUND.BACKGROUND
DISPLAYSCORE. BACKGROUND
```

SUMMARY

The variety of animation techniques available for small computers and the rich palette of examples already produced, will convince even skeptics that powerful tools are ready to enhance video and cable TV productions. Even the micro graphics computers of today's generation are far superior to the titlers and character generators widely used in audio-visual production.

By now the excuse that computer animation is so difficult that it can only be handled by programming staff can no longer be made. The software is so advanced that it allows the animator to concentrate on the graphic design without bothering about the internals of the assisting computer. What it does require however is the courage to develop new techniques for new machines.