MERGING AND TRANSFORMATION OF RASTER IMAGES FOR CARTOON ANIMATION

Bruce A. Wallace

Program of Computer Graphics
Cornell University

Abstract

The task of assembling drawings and backgrounds together for each frame of an animated sequence has always been a tedious undertaking using conventional animation camera stands and has contributed to the high cost of animation production. In addition, the physical limitations that these camera stands place on the manipulation of the individual artwork levels restricts the total image-making possibilities afforded by traditional cartoon animation. Documents containing all frame assembly information must also be maintained.

This paper presents several computer methods for assisting in the production of cartoon animation, both to reduce expense and to improve the overall quality.

Merging is the process of combining levels of artwork into a final composite frame using digital computer graphics. The term "level" refers to a single painted drawing (cel) or background. A method for the simulation of any hypothetical animation camera set-up is introduced. A technique is presented for reducing the total number of merges by retaining merged groups consisting of individual levels which do not change over successive frames. Lastly, a sequence-editing system which controls precise definition of an animated sequence, is described. Also discussed is the actual method for merging any two adjacent levels and several computational and storage optimizations to speed the process.

KEYWORDS: computer animation, computer graphics, merging, transformation
COMPUTER REVIEWS CLASSIFICATION: 3.41, 8.2

Introduction

The production of cartoon animation has always been a large-scale undertaking involving many man-hours of drawing, inking (or xeroxing), cel painting, background painting, and frame-by-frame film recording using an animation camera stand. The traditional cartoon animation process takes the following steps from the original storyboard to the finished product:

(1) Cartoon characters are drawn in pencil on separate sheets of white paper. A new and different drawing is made each time a change is to occur in the appearance of that character.

(2) Backgrounds which illustrate the settings into which a character is placed are painted on cardboard sheets.

(3) A tabular exposure sheet is written to direct the final assembly of cels and backgrounds and to specify how the camera stand should be configured for each frame.

(4) Each pencil drawing is either traced in ink or xeroxed onto the face of a clear sheet of acetate known as a cel. At this point, only a black line drawing appears on an otherwise transparent sheet of acetate.

(5) The various areas of each transparent cel which are to appear as color are painted with an opaque paint, similar to coloring inside the lines of a child's coloring book. The painting is done on the reverse side of the cel so that none of the black lines on the face are obliterated.

(6) Using the exposure sheet as a guide, the specified background and overlaying cel(s) are placed on the base of the camera stand. The lateral position of the base is

adjusted depending upon the desired position of the characters and the background. Generally, the cels and background can be moved independently. The camera is raised or lowered on its vertical support until the desired scaling of the combined image is achieved. It is not possible to enlarge levels of artwork independently without the use of an expensive multi-plane animation camera stand such as the one introduced by Walt Disney in 1936. The aperture on the camera lens can be varied depending upon the desired brightness of the frame. When these physical processes are complete, a single frame of animation is recorded onto film with the open and close of the camera shutter.

A review of Madsen's book "Animated Film:Concepts,Methods,Uses" [5] is beneficial in understanding in greater detail how the traditional animation process operates.

In recent years several computer-assisted cartoon animation systems have been developed, primarily for the purpose of quickening and improving various stages of the animation process. Two such systems, one at Cornell University and the other at New York Institute of Technology, along with several other installations, have shown cost savings and greater flexibility over traditional cartoon animation methods.

In each of these systems, drawings are maintained either in vector format as streams of connected vertices, or in raster format as areas of pixel data. Advantages and disadvantages of the two approaches are discussed by Levoy [3]. This paper will assume the raster format in all further discussion. Methods for computer assistance in steps 4 - 6 of the animation process have been developed for each system. Brief descriptions of computer-assisted cel creation and coloring processes (steps 4 & 5) are followed by computer methods for assisting in the final camera work (step 6).

A. Computer-Assistance in the Cel Coloring Process (Step 5)

The time required for cel coloring has been reduced tenfold through the use of area flooding algorithms such as those introduced by Smith [6] and Levoy [3]. Such algorithms process the pixels of a raster image directly. An enclosed area of the image is quickly flooded with color simply by indicating one "seed" pixel within that area. Each color has an opacity value associated with it which refers to the extent with which that color obscures whatever is behind it. This value is used in the process of overlaying images, discussed later in this paper.

Area flooding algorithms lay at the heart of all cel opaquing programs. Because of the great time savings such a program offers to the animation process, it has been the major inspiration for incorporating computers for production assistance.

B. Computer-Assistance in the Drawing-to-Cel Process (Step 4)

An opaquing program operates on raster data. This means the animators' drawings must be entered into the computer in pixel format to correctly interface with such a program.

Typically, in a raster-based, computer-assisted animation system, a drawing is input as pixel data by using a video scanning camera connected to a digital frame buffer. One such implementation is discussed by Stern [7]. The contents of the frame buffer can be manipulated by an opaquing program, such as the one mentioned above.

Much like the drawings, conventionally painted backgrounds can be input by scanning with the same type of video camera. Three separate passes are made using optical color filters (red, green, blue). Each pixel of the background image is stored as an RGB triplet. All pixels within a background are assumed to be fully opaque.

C. Computer-Assistance in the Frame Assembly Process (Step 6)

All of the computer-assisted processes discussed above deal with the alteration or creation of individual pieces of artwork. Traditionally, these levels have been assembled into a final composite frame on a standard animation camera stand. However, since cels and backgrounds are maintained in the computer in raster format, a method is necessary for performing the task of building the final frames of animation from the pixel data.

Merging is the process of combining pixel-based artwork into a final display frame using digital computer graphics. Building a frame of animation on a traditional animation camera stand can be simulated on a computer. The pixel-based artwork is assembled and transformed into a final raster image. Methods for this simulation process will be the discussed in detail later in this paper. Once computed, the final image can either be written to a frame buffer and output onto videotape or recorded directly onto film using a precision film recorder.

There are several advantages for using a computer to assist in this step of the animation process in this manner:

(1) The allowable number of cel levels is no longer limited by the physical density

of the acetate on which it is normally painted. The bottom level of artwork in a computer merged frame is as bright as all the levels above it.

(2) The physical limitations of the camera stand, which have restricted the independent manipulation of each artwork level, are no longer present. Levels may be transformed to simulate any hypothetical camera stand, including the multi-plane camera.

Merging Two Artwork Levels

A. Image Data (Artwork)

The majority of artwork processed by computer consists of "cels" colored by an opaquing program. Backgrounds, input by the three-pass scanning method, previously discussed, comprise the remainder of raster data used in producing an animated sequence. Each pixel of an artwork level is comprised of three color components (Red, Green, Blue) and an opacity value. The opacity refers to the percentage contribution of the pixel's own color versus whatever artwork is on the next lower level. While all pixels in a background are fully opaque, pixels in cels assume one of three states:

1) Fully Opaque: OPACITY(PIXEL)=1

This occurs in all colored areas.

2) Fully Transparent: OPACITY(PIXEL)=0

This occurs in all areas void of color.

3) Partially Opaque: 0 < OPACITY(PIXEL) < 1

This occurs at the edges between opaque and adjacent transparent areas, producing soft, anti-aliased edges.

B. The Merging Process

Merging multiple levels of pixel-based artwork for cartoon animation can be thought of in terms of a Z-buffer algorithm. Typically, a Z-buffer algorithm is used for hidden surface removal of polygonal data. It determines what portions of each polygon are ultimately visible in the final raster display. A merging algorithm determines what portions of each level of artwork are ultimately visible in the final frame. Because a Z-buffer algorithm only considers each level once, an increase in the number of levels produces a linear increase in the time required to build a final display image. However, a Z-buffer algorithm can not offer one necessary feature desirable in a merging algorithm: acceptable anti-aliasing along edges. There are two cases where adverse effects can occur:

(1) Catmull [1] has pointed out that Z-buffer algorithms do not produce correctly anti-aliased edges in a raster image. Only pixel percentage clipping, using precise edge information, generates the correct anti-aliasing of edges in all cases. Without the edge information, slightly incorrect results will occur when overlap occurs between two or more pixels through which edges pass. This occurs when combining levels of scanned data, because the scanning process does not yield precise edge information. Incorrect results are more noticeable in applications where a Z-buffer algorithm is typically used than in cartoon animation produced by a merging algorithm. In the case of polygonal data, many edges may overlap. Unfortunately, as the number of overlapping edges increases, so does the error. Since many of these images are animated slowly, such errors are easily detected. This is not true in cartoon animation. The number of edges that actually overlap are minimal since the number of artwork levels is much less than the number of overlapping polygons found in a typical image produced by a Z-buffer algorithm. More importantly, frame to frame motion in cartoon animation is typically quick and tends to hide slightly incorrect color at intersecting anti-aliased edges. For these reasons, the results, while not perfect, are more than adequate for cartoon animation.

(2) It has also been pointed out that a Z-buffer algorithm will produce inconsistent results at the edges if the order in which levels are pairwise combined is not bottom-up. This is due to the fact that the calculations involved in the Z-buffer algorithm are not associative. The option of merging levels of artwork, independent of the order in which the merges occur, is desirable in a computer-assisted animation system. Frequently, groups of adjacent levels of artwork will not change from one frame to another. It would be advantageous to be able to merge these levels together and keep the resultant for use in subsequent frames. This reduces the total number of merges which have to be performed for an animated sequence. Due to the nature of the animation, the order by which groupings of adjacent levels are merged and retained may not necessarily be bottom-up. Thus, the need to be able to assemble the final image in any order becomes important. An order-independent merging method has been developed which does not require the strict bottom-up merging order by which levels are paired and merged into intermediary images. It eliminates this restriction, producing no adverse effects in the anti-aliased edges caused by variations in the order by which merges between levels occur.

Before presenting the method for order-independent merging, the basic logic for a simple merging process must be presented. This closely parallels the logic used in a Z-buffer algorithm. Only two levels are merged at any one time,

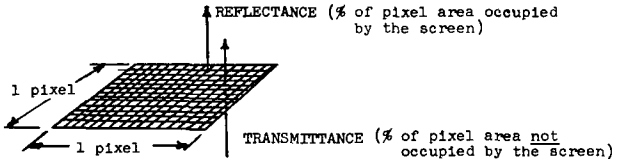
forming a resultant level with its own color and opacity components, which can in turn be merged with any other existing level. The calculations for resultant pixels along the edges have been included. The formulations in the simple process rely upon order dependence for correct results. Levels are merged in order from the background upwards. Since the background is known to be fully opaque, the opacity of the pixel resulting from each merge operation is equal to 1. Thus, a merge is always between a fully opaque bottom level and a top level assuming any of the three opacity states outlined previously. To produce the merged pixel, RESULT, from the two adjacent level pixels, TOP and BOTTOM, the following algorithm is performed:

```
Case 1 if OPACITY(TOP) = 1 then

       COLOR(RESULT) = COLOR(TOP)

            If top is fully opaque
            result assumes top color

       else
Case 2 if OPACITY(TOP) = 0 then

       COLOR(RESULT) = COLOR(BOTTOM)

            If top is transparent
            result assumes bottom color

Case 3 else

       COLOR(RESULT) =
       {OPACITY(TOP) * COLOR (TOP)} +
       {{1 - OPACITY(TOP)} * COLOR(BOTTOM)}

            If top is partially opaque
            result assumes interpolant
            between top and bottom colors
            regulated by the top opacity

       fi
```

The term "COLOR" refers to one of the three color components found at the pixel, either red, green, or blue. The algorithm is executed once for each component.

The bottom-up method always involves building an image which contains all artwork from the background up to and including some overlaying cel. In the case of order independence, the process of assembling a final raster image is mathematically associative. This is true, because the act of cummulatively combining levels of given color and opacity is an additive process. A physical model which has these same properties has been constructed to clarify the formulas necessary for calculating a resultant pixel's color and opacity values from the known color and opacity values of two cel pixels which are being merged together:

A pixel can be thought of in terms of the physical analogy of a homogeneous screen of a given density. A given percentage of the area contributes the color of the screen material to the overall appearance of the screen. This percentage will be referred to as the "reflectance". The remaining percentage of the area is contributed from levels below. This will be referred to as the "tranmittance".



REFLECTANCE = 1 - TRANSMITTANCE
for 0 <= REFLECTANCE <= 1

Figure 1 - Screen Model of Pixel

The use of the terms "reflectance" and "transmittance" should not be mistaken for those found in discussions of optics and ray tracing. The two terms assist in conceptualizing the derivation of the final formulas.

Overlaying any two pixels, each with its own opacity (or "reflectance") values, can be thought of as overlaying two screens of given density and color with a spatial integrator (diffuser) of 100% transmittance between them. A new interpolated color and resultant "reflectance" of this combination can be obtained from the original known densities and colors of the two screens.

Using the following symbols, Figure 2 aids in deriving formulas for calculating the resultant components:

CT = Color value for top pixel
CB = Color value for bottom pixel
CR = Color value for resultant pixel

RT = Reflectance of top pixel
RB = Reflectance of bottom pixel
RR = Reflectance of resultant pixel

TT = Transmittance of top pixel
TB = Transmittance of bottom pixel
TR = Transmittance of resultant pixel

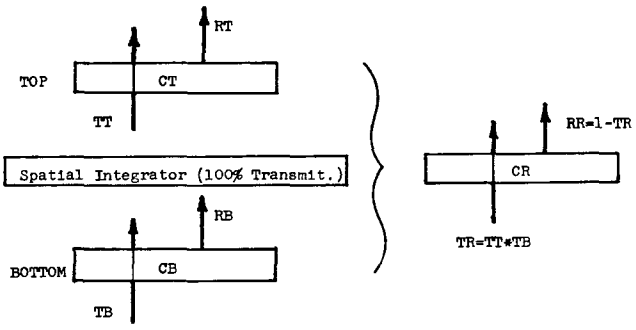


Figure 2 - Cross-Section of Two Level Merge and Resultant Level

The following calculations arrive at the solution for CR and RR in terms of the known values CT,RT,TT,CB,RB,TB. Note that

the term "BELOW" refers to whatever is below the bottom pixel.

```
Color Contribution of Resultant =
          (RR * CR) + (TR * BELOW),

which is =
          (RT * CT) +
          (TT * (RB * CB)) +
          (TT * TB * BELOW)

where
(RT * CT)
 represents Color Contribution of Top,
(TT * (RB * CB))
 represents Color Contribution of Bottom,
(TT * TB * BELOW)
 represents Color Contribution of Below,
(RR * CR)
 represents Color Contribution of Resultant,
(TR * BELOW)
 represents Color Contribution of Below.
```

Knowing that TR = TT * TB, the color contribution from below can be subtracted from both sides of the equation. Substitutions for RR and TR are performed since RR = 1 - TR and TR = TT * TB. Thus, solving for CR and RR produces:

$$CR = \frac{(CT * RT) + (TT * CB * RB)}{1 - (TT * TB)}$$

$$RR = 1 - (TT * TB)$$

The merging algorithm for the order-independent method is subject to the same three case conditionals as the bottom-up method, each governed by the opacity value of the top pixel. The first two cases remain the same. Case 3 (partial opacity) assumes the above formulas for CR and RR for the resultant color and opacity values.

Figure 8 offers a visualization of the opacity mask for merging the cels from Figures 4 and 5. Pure white areas represent fully transparent pixels, while areas of pure black represent fully opaque pixels. The penumbre surrounding the opaqued areas represents pixels of partial opacity, producing soft, anti-aliased edges. The final image created by using this opacity mask to merge these cels with the background is also pictured in Figure 8.

Transforming One Artwork Level

In addition to the task of merging levels of artwork into a final image, it is necessary to be able to manipulate each level relative to the "camera's eye". Zooming and panning are examples of commonly used operations. Variations in the intensity with which a level contributes to the final image are also necessary to simulate the effect of the camera's aperture. Thus, there are two general classes of transformations which affect each level of either original or merged artwork. The first class consists of geometric transformations and the second class consists of intensity transformations. Geometric transformation can be accomplished in several manners:

(1) A standard 4 X 4 transformation matrix is constructed from scaling, translation, rotation, and perspective information For each pixel in the transformed image, an inverse transformation is performed using the 4 X 4 matrix. This determines the location on the untransformed image from which pixel information can be used for sampling purposes. Methods of bilinear interpolation and filtering are used to calculate the final color of the transformed pixel. Refer to a detailed discussion by Levoy, Feibush, and Cook [4].

(2) Alternatively, the transformation can be expressed as an x-pass and a y-pass. Filtering is performed in only one direction at a time. Such a two-pass stream processor, introduced by Catmull and Smith [2], has the same effect as the 4 X 4 transform above, but reduces the total compute time required to transform a full raster image.

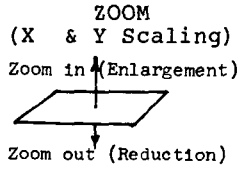
Intensity transformation can be accomplished by altering the opacity component of each pixel by a given factor. An intensity transform performed on one level of artwork will either increase or decrease the extent to which that level contributes to the final raster image.

Assembling Composite Images

The basic method for merging any two adjacent cels and transforming the resultant image have been presented. It is desirable to be able to specify to the computer many cels and accompanying transformations for automatic assembly of a final composite image. For this reason, a data structure which models any animation camera configuration is necessary. There are two components to consider when devising a structural model for a physical animation camera stand:

(1) Artwork - One level of artwork, whether it be cel, background, or merged resultant is the basic material unit upon which the camera stand operates.

(2) Operations - For the purposes of cartoon animation, a subset of all possible geometric transforms will be used. A linear intensity transform will also be included in the model. Refer to Figure 3 (below).



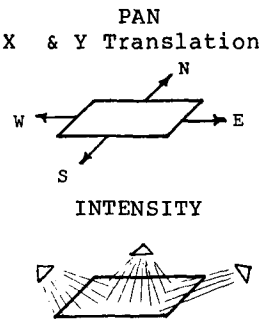


Figure 3 - Three types of operations affecting artwork

A modified tree structure, such as the one shown in Figure 10, is used to model all camera set-ups. The three types of nodes used in this structure are:

(1) Artwork (Original Scanned Image Data)
(2) Operation (Zoom, Pan, Intensity)
(3) Merge (Adjacent levels merged into resultant level)

All original, scanned artwork resides at the leaves of the tree. By traversing toward the root, performing either a merge or operation at each intermediate node, the final frame is constructed. The traversal can be seen as an expression evaluator which uses three classes of nodes: operands, unary operators, and binary operators. The three types of configuration nodes listed above fall into these same three classes. The artwork is the basic operand upon which all operations are performed. A transform operation is a unary operator, only performed on one level of original or merged artwork. A merge is a binary operator, requiring two levels as input.

An intermediary image is kept at any operation or merge node whose subtree does not change over a predefined number of frames. This means it is necessary to build the subtree only once even if it is used for subsequent frames. As it was earlier stated, order-independent merging allows such intermediary images to be built. The configuration tree structure presented here allows such frame-to-frame coherencies to be recognized.

In addition to this cost saving feature, new and elaborate camera configurations can be modeled by constructing the proper tree structure to simulate the desired camera stand. Configuration 1, as illustrated in Figure 10 produces a final merged image as pictured in Figure 11. By slightly varying the tree structure in Configuration 1 to incorporate independent panning of the cels of Example B (Figures 6 and 7), Configuration 2 is created as shown in Figure 12, and a final merged image is produced, as pictured in Figure 13.

Improved Merging Methods

In typical cartoon animation, and particularly in the case of limited animation, three types of coherency can be identified in the final frame assembly process: coherency between frames, between areas within each frame, and between pixels within each area. Several algorithmic approaches have been developed for taking advantage of each type of coherency, thus reducing the overall time, space, and logic required to produce an animated sequence.

A. Frame-to-frame Coherence

In typical cartoon animation, some or all artwork levels may change from one frame to another. Those portions of the configuration tree structure which do not change may be kept for subsequent frames. This greatly reduces the number of merges and transform operations which must be performed on each frame. In fact, it is possible for entire frames to be kept and used again for subsequent frames if no animation occurs at these frames. One example of this occurs when the animation is drawn on "two's". This means new drawings are only animated on every other frame of the final sequence. Drawing on two's is prevalent throughout traditional cartoon animation as a method of reducing the number of drawings by 50%.

All assembly information about the frame-to-frame layout of an animated sequence is contained within a tabular exposure sheet. Each row on a sheet contains information pertaining to the construction of one frame, while each column refers to one level of artwork. An entry within a column on one row can assume one of two states:

(1) Non-blank : Name of artwork.
(2) Blank : Assumes the default from the first non-blank entry above in the same column.

In order to achieve totally automatic production on the computer, a sequence database has been developed which makes this information available to the computer. A screen editor is used to enter and edit all information concerning configuration structures, names of artwork, and operation parameters. A video terminal, such as a DEC VT100, simplifies this job by displaying a facsimile of the physical exposure sheet. A sample display from such an editor is pictured in Figure 16.

Once all assembly information has been entered into the sequence database using the interactive exposure sheet editor, the process of deciding in which order levels should be merged and which merged levels should be retained begins. A frame spread spanning over a predefined number of frames

starting at the frame currently being built is examined for groupings of cels on adjacent levels which remain unchanged. These intermediary groupings are built while assembling the current frame. A pool of these intermediary images is maintained by the computer, and before starting a new frame, the work requirements are first determined and any intermediary frames available in the pool are used.

B. Area Coherence

In many cases, a character drawn for cartoon animation is found to reside within a small portion of the full screen area. Generally, cartoon characters are drawn in the center of the viewing area. For limited animation, small details, such as eyes or mouths, are drawn as a separate level of artwork. Consequently, the majority of the corresponding cel is transparent. Only a small central portion contains the artwork. Rather than work with a full screen image for every level of artwork, it is desirable to determine the smallest containment area which surrounds the artwork on each cel and to consider only that area when merging or transforming that cel. This further reduces the time required to build a final frame.

A min-max boundary box which surrounds the non-transparent artwork of a cel can be determined from the original scanned artwork. This can be achieved by manually locating the four min-max boundaries on the original full-screen scanned image. This boundary search can also be accomplished automatically in software by examining each pixel of the original scanned data. A third and faster method uses a hardware image analyzer capable of sampling intensity levels of pixels in a designated rectangular area of the display. Figures 4, 5, 6, and 7 are examples of cels which each have surrounding boundary boxes shown against a 12 field animation grid for reference.

The min-max boundaries of two cels to be merged are used to construct a table which describes how the two cels overlap and specifies how they are to contribute to the resulting merged image. This table will be referred to as the "result table" of the two cels being merged together. The min-max boundary box of the merged image is assumed to be the minimum rectangular area containing the union of the two cel boundary boxes. By extending the boundaries of both cels to the edges of the boundary box of the merged image, nine regions are defined. Each of these regions corresponds to an entry in the result table. Two graphical representations of result tables are pictured in Figure 14. Example A in this figure represents the result table of the cel from Figure 5 prior to merging on top of the cel from Figure 4. Similarly, Example B represents the result table of the cel from Figure 7 before merging on top of cel from Figure 6. Each entry in a result table refers to a particular area in the merged image. An entry contains the following information:

(1) Type of contribution toward the merged image

   a. BOTH cels
   b. TOP cel only
   c. BOTTOM cel only
   d. NEITHER of the two cels

(2) X dimension of the area (Number of pixels wide)

(3) Y dimension of the area (Number of scanlines high)

The actual result table dimensioning and entry values for Example B are pictured in Figure 15.

The result table is used as a guide in assembling the resultant merged image. The merging algorithm outlined on page 10 need only be executed in an area of the merged image in which a contribution exists from both cels. No other areas require actual calculations.

C. Pixel Coherence

Since cels are comprised primarily of large homogeneous areas, it is desirable to maintain all cels in a run-length encoded (RLE) format. On the average, a cel consisting of anti-aliased lines, contains approximately eight pixels to every encoded run along a scanline. In order to take advantage of this coherency, an algorithm has been developed which allows cels to be merged and transformed in RLE format, reducing the time required to assemble a final frame by as much as a factor of eight.

The merging algorithm on page 10 can be changed slightly to accommodate an encoded run rather than a single pixel. Figure 9 shows an enlarged area of the cel from Figure 4, outlined by a white box on the final merged image in Figure 8. The top portion of the close-up figure is shown as alternating black and white lines of various lengths. This visual format is used to "expose" the underlying RLE scheme. A switch between white and black on any one of these scanlines indicates the start of a new encoded run. A run is stored as a pixel counter, three color components (RGB), and an opacity value. The algorithm has been expressly designed such that the merged image produced by merging two RLE artwork levels together is also in RLE format. All original, opaqued cels are stored in RLE format and all resultant levels are kept in this format during merging and during transform operations .

The lateral position of a level of artwork is described by the X and Y

coordinates of the boundary box surrounding the artwork. In order to perform translation on a level these coordinates are simply reset to the desired location. Thus, no calculations on the image data are required.

Typically, enlargement and reduction, which are scaling operations, have been performed in raster format by one of the two geometric transformation methods discussed earlier. Both zoom operations can be performed directly on the RLE image data by utilizing a two-pass scheme. First, the image is scaled in the X direction. The resulting stretched or compressed image is then scaled in the Y direction to produce the final scaled image. Scaling may just as easily be performed by fist scaling in Y and then in X. This two-pass algorithm involves logic which incorporates linear interpolation between encoded runs along a scanline when scaling in X, and between scanlines when scaling in Y. Figures 11 and 13 were produced using RLE merging and zooming, and, as expected, took a small fraction of the time required to perform the same operations on all pixels.

Lastly, in order to perform an intensity operation on an RLE image, the opacity component of each encoded run is transformed by a given factor, as outlined earlier.

Conclusion

The methods presented in this paper have been implimented on a DEC VAX 11/780 computer and have produced images acceptable for television broadcast. Within the next decade many of these methods will be designed into the hardware of "smart" frame buffers. Not only do these methods greatly reduce traditional production time requirements, but they also increase the image-making possibilities available to cartoon animation.

References

[1] Catmull, Edwin, "A Hidden-Surface Algorithm with Anti-Aliasing", Computer Graphics, volume 12, number 3, August 1978.

[2] Catmull, Edwin, and Smith, Alvy Ray, "3-D Transformation of Images in Scanline Order", Computer Graphics, volume 14, number 3, August 1980.

[3] Levoy, Marc, Computer-Assisted Cartoon Animation, M.S. Thesis Cornell University, August, 1978.

[4] Levoy, Marc, and Feibush, Eliot, and Cook, Robert, "Synthetic Texturing Using Digital Filters", Computer Graphics, volume 14, number 3, August 1980.

[5] Madsen, Roy, Animated Film: Concepts, Methods, Uses, New York, Interland Publishing Inc., 1969.

[6] Smith, Alvy Ray, "Tint Fill", Computer Graphics, volume 13, number 2, August 1979.

[7] Stern, Garland, "SoftCel - An Aplication of Raster Scan Graphics to Conventional Cel Animation", Computer Graphics, volume 13, number 2, August 1979.

Acknowledgements

The author wishes to thank Dr. Donald Greenberg and Marc Levoy for their help and discussion on numerous design approaches, with special thanks to William Hanna whose continual support has made all this work a reality at the Hanna-Barbera Productions studios in Hollywood, California.

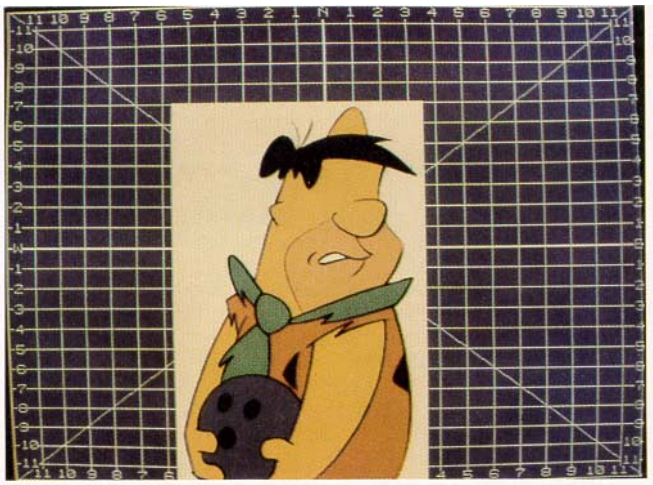


Figure 4 - Example A , Cel 1

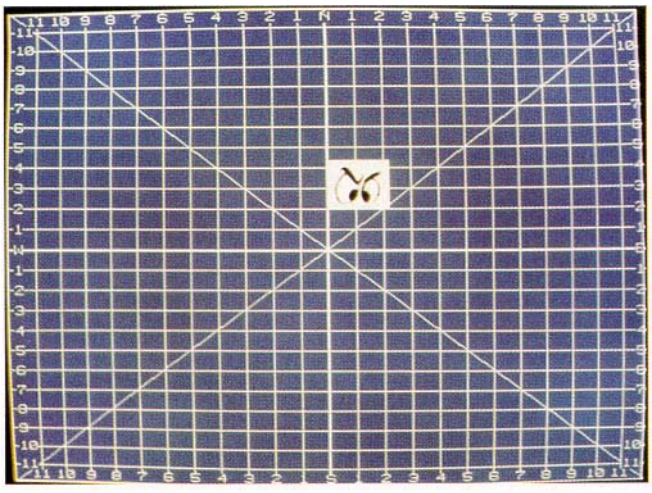


Figure 5 - Example A , Cel 2

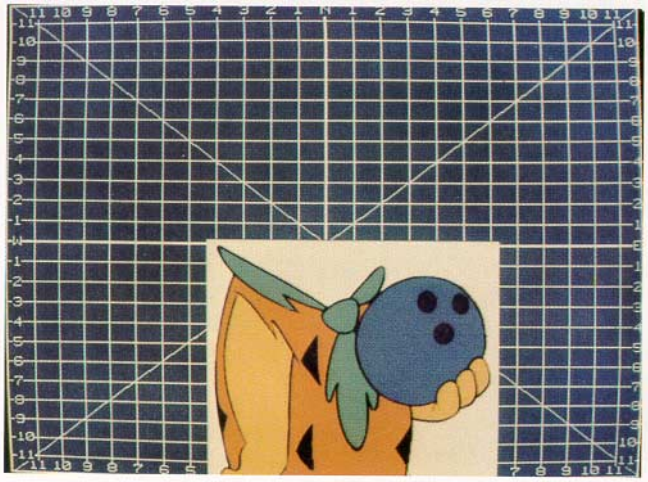


Figure 6 - Example B , Cel 1

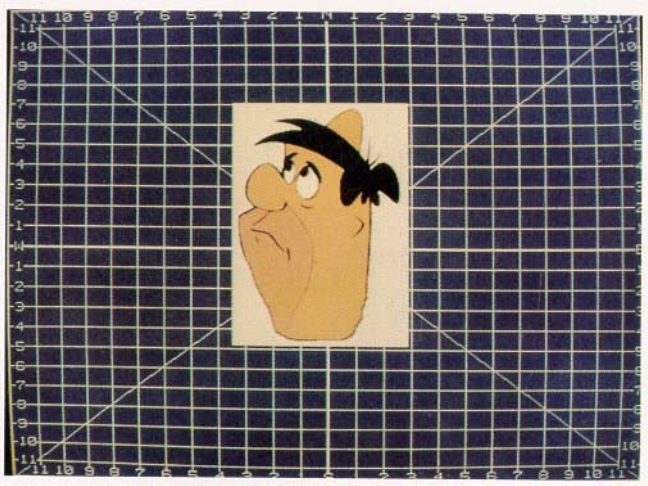


Figure 7 - Example B , Cel 2

Figure 8 - Opacity Mask and Final Merged Frame

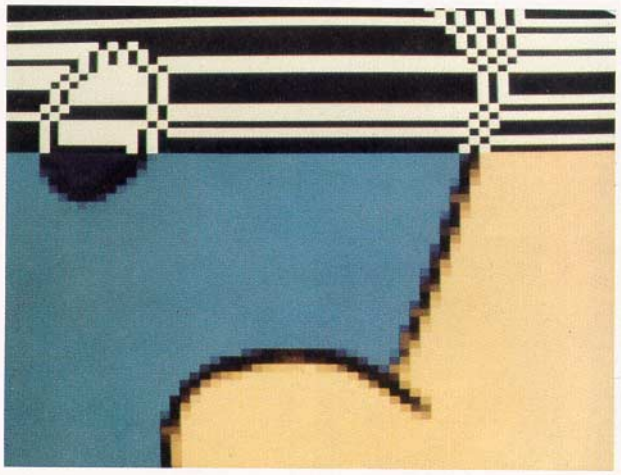


Figure 9 - Close-up of RLE cel showing encoding scheme

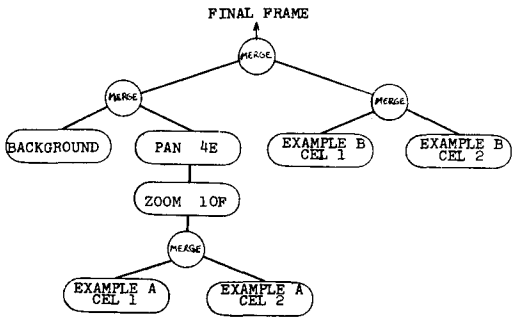


Figure 10 - Tree structure for Configuration 1

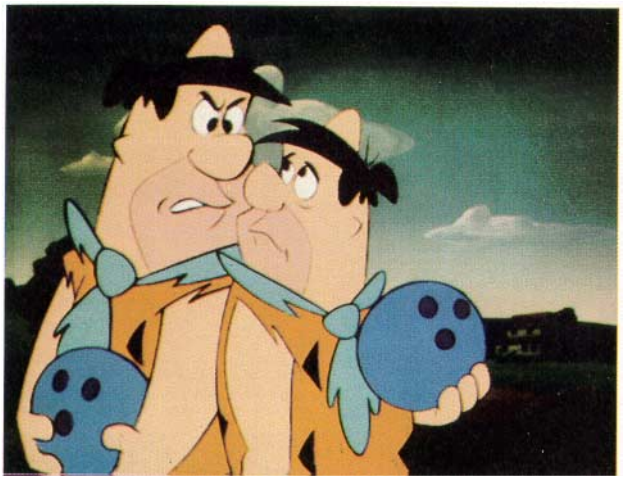


Figure 11 - Final merged image for Configuration 1

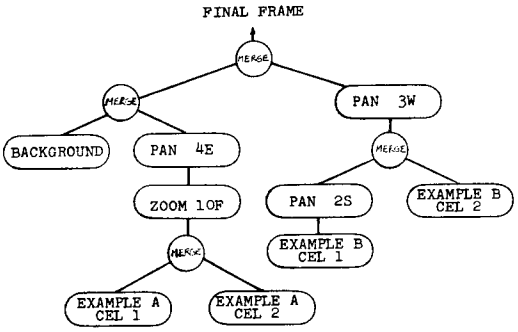


Figure 12 - Tree structure for Configuration 2

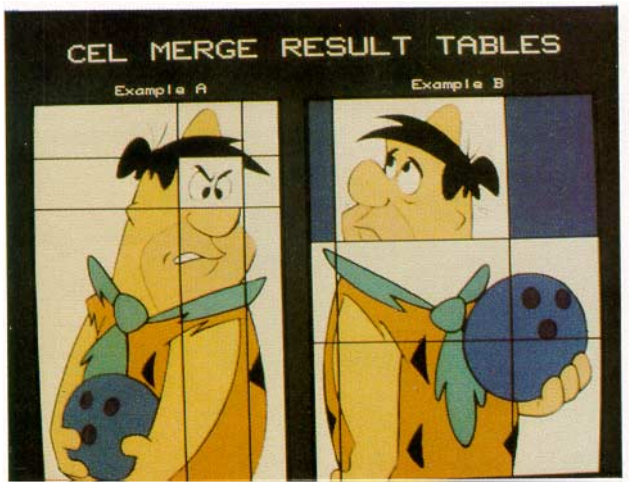
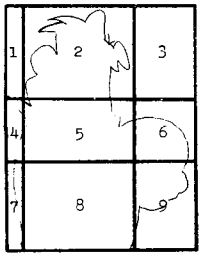


Figure 14 - Graphical layout of Result tables for Examples A & B



Area	Type	X	Y
1	NEITHER	18	138
2	BOTTOM	134	138
3	NEITHER	74	138
4	TOP	18	100
5	BOTH	134	100
6	TOP	74	100
7	TOP	18	144
8	TOP	134	144
9	TOP	74	144

Figure 15 - Result table for Example B

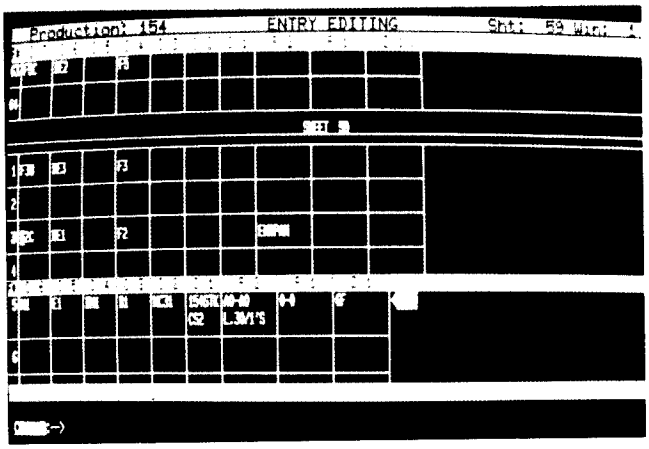


Figure 16 - Sample display from interactive exposure sheet editor