Real-time Image and Video Processing in GEM

Mark Danks

University of California, San Diego
mdanks@cybermed.ucsd.edu,
http://cybermed.ucsd.edu/mdanks

Abstract

The Graphics Environment for Multimedia, or GEM, is a visual programming language for real-time computer graphics. Operating within the Pd environment, a digital audio and MIDI processing program, GEM creates and manipulates polygonal graphics. Real-time image and video processing have recently been added, increasing the artistic possibilities. Combined with Pd, users have access to an integrated visual and sonic programming environment.

1 Introduction

The Graphics Environment for Multimedia (GEM) is a real-time computer graphics language introduced in 1996 [1]. GEM was written with OpenGL, a platform independent graphics library, created by Silicon Graphics, Inc. (SGI). Originally designed for the Max/FTS environment, GEM is now an integral part of Pd [2], a real-time sound and graphics processing program that runs on Silicon Graphics and Windows NT computers. While it is compatible with Max/FTS 0.26 patches, the patch is not the focus of the program. Pd provides increased flexibility and control over data (Pd stands for pure data). For a more complete discussion, see Puckette [3].

Developed by the author for computer graphics, GEM supports polygonal manipulation, controlling the graphics as objects, instead of as a sequence of vertices. Recent developments in GEM have increased the capabilities beyond polygonal objects to include real-time image and video processing. Instead of using a traditional programming language and environment such as the Graphics Library Utility Toolkit (GLUT) [4], GEM provides a way for non-programmers to generate complex, and artistic, graphics.

OpenGL was designed to be a high-performance, window system independent interface to computer graphics. Originally created by SGI, OpenGL is now maintained by an Architectural Review Board consisting of various computer companies such as Intel, SGI, and Intergraph. Because OpenGL is a standard which exists on many different computer platforms, GEM can be ported to any operating system which supports OpenGL.

2 Background

In OpenGL, color values are represented by four floating point values, red, green, blue, and alpha. The alpha value is used when two colors are blended together. Alpha represents how translucent the color is, from 0.0 being totally transparent, to 1.0 being totally opaque. For instance, the alpha value can be used to simulate colored glass which might appear in a scene. Images and texture maps can have different alpha values for each pixel. For example, any black pixel can be made transparent so that one can see through parts of an image.

OpenGL uses double buffering as a technique to produce flicker free animation. All drawing occurs in a hidden graphics window, which is swapped with the viewable one when the drawing is done. This process continues for each new frame. With single buffered graphics, the user can see the graphics being drawn instead of being presented with a completed image. Single buffered graphics also do not clear the frame buffer until the user designates it as such, so that the graphics window acts like a canvas for drawing.

3 Usage

MIDI and digital audio systems operate alongside graphics and video systems, rather than in two competing environments. By integrating both sight and sound, GEM and Pd allow composers and artists' to create inter-media compositions which utilize both visual and musical ideas. One instance of this combination is the piece *headingsouth* by the author [5].

3.1 Objects

Five types of objects exist in GEM. *Geometrics*, or *geos*, are objects which create a concrete shape. The *non-geometrics*, or *non-geos*, consist of objects which affect the screen display, but do not have a defined shape. The pix objects, which are the focus of this paper, perform pixel operations and generate images. The *manipulators* control and transform the various graphics. The *controls* have global effects and manage the graphics window.

3.2 Changes from Previous Version

GEM has undergone some fundamental changes from the previous version. In the old version, the start of a GEM patch could be a square, which was then connected to `color` and `rotate`, and finally to render. This was accomplished through display lists and other convoluted methods because GEM did not mirror the OpenGL process. Now, a gemhead starts the rendering process, which can be connected to color and rotate, and finally to square. The gemhead object connects to the graphics window manager, and must be the first object in any GEM chain. The color object sets the state for subsequent vertex operations, while the square object makes the vertex OpenGL commands. See Figure l for an example of how to generate a red square.

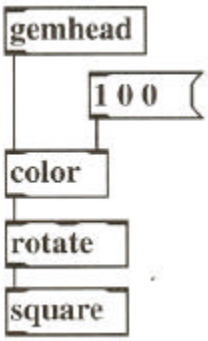

Figure 1: A simple GEM patch

This change makes it easier for GEM to integrate the various OpenGL operations. OpenGL is a state based system [6] with only one state, consisting of the various rotations, translations, and scalings that have been applied, active at any one time. Because the vertex commands are modified by the current state, each chain can have multiple *geo* objects. In Figure 1, the chain could be expanded to include a translate and sphere after the square. The sphere would also be red because the color had been set earlier in the chain. This same technique can be used with any of the manipulators, because they establish the state for all *of* the *geos.* Texture mapping is also a state, so images can be used by multiple objects, assuming that a pix_texture was used earlier in the chain. A much more detailed explanation of state based systems and transformation matrices appears in Computer Graphics: Principles and Practice [7].

3.3 Image Processing

GEM makes it easy to load in images and process them. The processing can range from gain controls and thresholds to convolution kernels. Once an image is loaded with a pix_image, any objects "downstream" from the pix_image will affect the pixels. See Figure 2 for a simple image patch which applies a gain to the pixels. The three numbers used in the example are the gain components for the red green blue (RGB) values of each pixel.

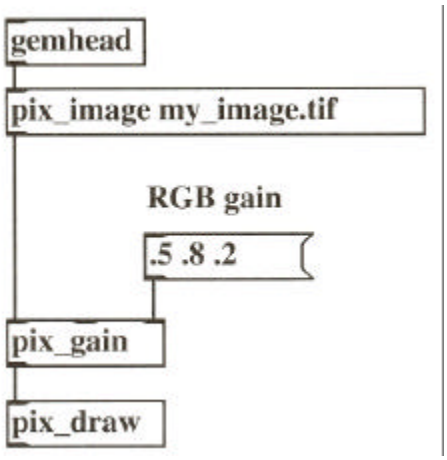

Figure 2: A simple image

A large number of objects have already been developed, including pix_gain, pix_convolution, pix_threshold, and pix_2grey (greyscale conversion). Many traditional image processing filters exist as well [8]. Because GEM is written in C++, a complete class hierarchy is provided, making it easy to create new objects. All a developer needs to do to create a new GEM object is to derive a new class.

GEM uses a demand driven, or pull system, to reduce unneeded computation. Once an image has been processed, the completed image is used until one *of* the pix objects changes. The pix_buffer is an object which helps with this demand based system. The object buffers images so that computation is only done when objects upstream from the pix_buffer change. For example, if an image is converted to grey scale with pix_2grey, the processed image can be stored in a pix_buffer so that it does not have to be recomputed every time something "downstream" in the chain changes.

3.4 Video and Movie Processing

Video and movie processing use the same pix objects as static image processing does, simplifying the system for end users. Figure 3 demonstrates the same processing as Figure 2, except that it uses a video camera as the source signal. Internally, video is just a stream *of* static images that are constantly being changed. An object like pix_buffer would only add extra computation because the image is never the same.

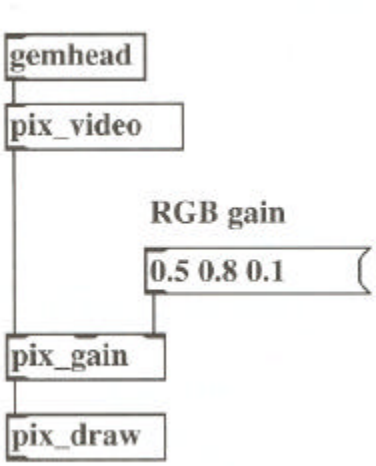

Figure 3: A simple video

3.5 Polygonal Graphics

Instead of being a dedicated image processing program, such as Adobe Photoshop, or a polygonal based program, as many CAD packages are, GEM integrates pixel processing with polygonal manipulation in a manner which is transparent to the end user.

As Pd combines MIDI with digital audio, GEM integrates polygonal graphics with pixel based graphics. Combinations range from displaying images as under or overlays for polygonal graphics, to texture mapping images or video onto objects. As shown in Figure 4, it is easy to texture map video onto a sphere. All of the *geometrics* provide appropriate texture mapping coordinates, depending on their shape. For instance, `sphere` wraps the image around itself. Of course, all of the usual pixel processing objects can be used at the same time.

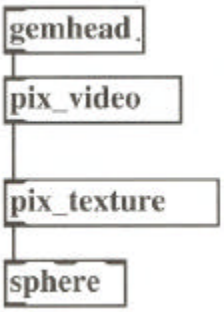

Figure 4: Texture mapped

Texture mapping is done by modulating the polygonal graphic with the RGBA pixel values of the image. Normally, the polygonal graphic has a white color so that the image appears undistorted. However, by changing the color of the polygon, the color of the image can be modulated. Effects such as lighting and alpha blending also contribute to the texture. These techniques increase the processing capabilities without having to modify the actual pixel values.

3.6 Pixel Manipulation

Two powerful techniques for image processing which use the alpha color value are compositing and alpha masking. Compositing is the merging of multiple images to generate a single new image. The blending is controlled by the alpha values of the images. The `pix_composite` object combines two images together. Alpha masking is the technique of "masking" out part of an image so that it becomes transparent. An example patch using `pix_mask` is shown in Figure 5.

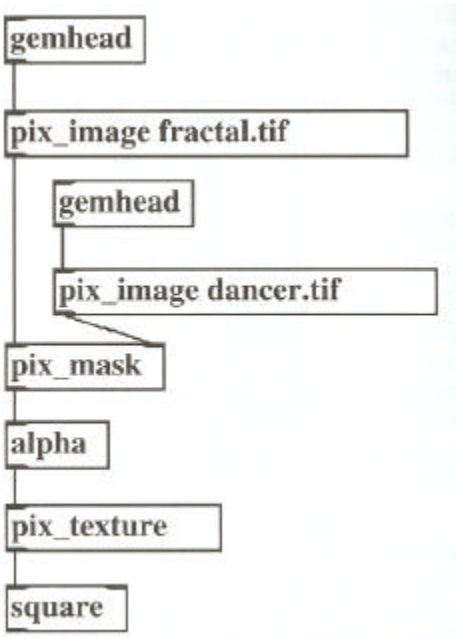

Figure 5: A patch with `pix_mask`

In Figure 5, the mask is generated by overlaying a second image on the primary image. Instead of blending the colors of the two images together as `pix_composite` does, `pix_mask` changes the alpha values of one image based on the RGB color values of another image. Figure 6 is an example of an image that was created with `pix_mask`. The original image was a fractal pattern, and the mask was a dancer.

Another image technique can be achieved using a single buffered window. Because the frame buffer is not cleared, translating and rotating a line is equivalent to painting with a brush. When the alpha value of the "brush" is modified, an effect similar to water color paint is produced [9]. Using `pix_snap`, users can make a "snapshot" of the frame buffer and use it as an image for processing and drawing.

Figure 6: Resultant

One area of recent development is motion and static image analysis. This processing ranges from simple motion detection through the comparison of sequential video frames, generating a single number, to edge detection, a process which determines the shapes of objects. The motion analysis objects are made up of both static analyzers for still images and time based objects for movies or live video.

4 Summary

The new developments in GEM give increased manipulation of pixel-based graphics. By integrating polygonal and pixel graphics, non-programmers can generate complex images which respond to real-time controls. Because GEM is an easy to use visual language, users do not need to learn a complex programming interface. Combined with Pd, a single unified environment has been created, bringing together visual and sonic media for creative expression.

References

[1] Danks, M. 1996. "The Graphics Environment for Max," *Proceedings of the]996 ICMC,* San Francisco, CA: International Computer Music Association, pp. 67-70.

[2] Puckette, M. 1996. "Pure Data: another integrated computer music environment," *The 2nd InterCollege Computer Music Concerts,* Kunitachi, Japan, pp. 37-41.

[3] Puckette, M. 1997. "Pure Data," *Proceedings of the 1997 ICMC,* San Francisco, CA: International Computer Music Association.

[4] Kilgard, M. 1996. *OpenGL Programming for the X Window System,* Reading, Mass.: AddisonWesley.

[5] Danks, M. 1996. *headingsouth,* San

[6] Neider, J., T. Davis, and M. Woo 1993. *OpenGL Programming Guide,* Reading, Mass.: AddisonWesley.

Foley, J., A. van Dam, S. Feiner, and J. Hughes 1990. *Computer Graphics: Principles and Practice, 2d ed.,* Reading, Mass.: AddisonWesley.

Jain, A. 1989. *Fundamentals of Digital Image Processing,* Englewood Cliffs, New Jersey: Prentice-Hall.

[9] Haeberli, P., and M. Segal 1993. "Texture Mapping as a Fundamental Drawing Primitive," *Proceedings of the Fourth Eurographics Workshop on Rendering,* Paris,