# DIPS : the real-time digital image processing objects for Max environment

**Shu Matsuda** syu@kcm-sd.ac.jp

**Takayuki Rai** rai@kcm-sd.ac.jp

Sonology Department, Kunitachi College of Music
5-5-1, Kashiwa-cho,Tachikawa-shi,
Tokyo,190-8520 JAPAN

## Abstract

In this paper, we would like to present the set of new Max objects that handle real-time visual image events in the jMax running environment. This set of objects, named "DIPS"(Digital Image Processing with Sound), enables the interaction between audio events and visual events in the Max patch, thus strongly supports the realization of real-time interactive multimedia art.

## Introduction

The development of the DIPS has been started two years ago by Shu Matsuda in the Max FTS environment running on SGI computers. First the object that controlled movie files was constructed, then video I/O objects were added, and later each OpenGL function became available as a Max object. In last winter the DIPS was re-designed in order to port to jMax environment. Now, the DIPS offers more than a hundred various objects.

## Hardware

Currently DIPS runs on the SGI O2 computer and Octane computer that contains either the personal video board or the digital video board. DIPS is designed to get the maximum execution ability from both hardware. When DIPS starts up, it detects the hardware configuration and prepares the hardware-specified DIPS object-set. Thus, a user can program DIPS objects in the same way for both hardware, but may notice it when DIPS runs in real-time.

## DIPS objects

DIPS objects, developed as external objects of jMax, were programmed in C language with OpenGL and the SGI development software tools including the digital media library. DIPS projects visual images in the X Window using GLX. The off-screen pixel buffers are also available since the window system is supported by GLX-pbuffer extensions.

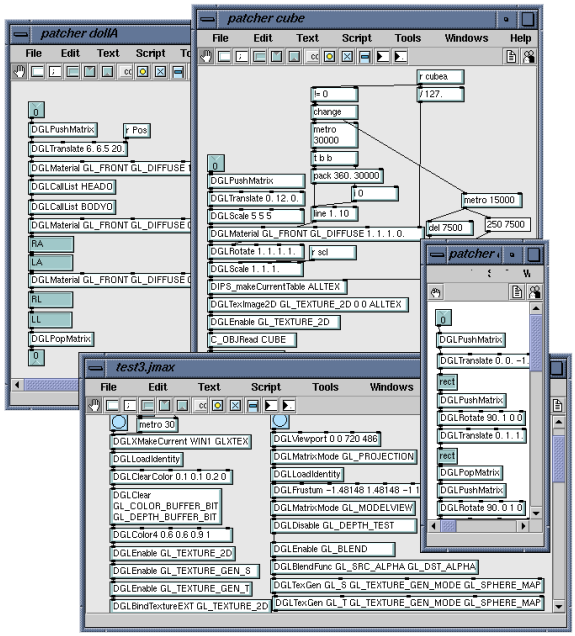


**Figure 1: DIPS objects in jMax.**

DIPS objects can be categorized into the following seven functions:

### 1) Video signal I/O

DIPS can handle video signal I/O in real-time. These objects work a bit differently according to the hardware circumstance. At the SGI Octane computer with the digital video board the direct data transfer from the video node to the texture memory is employed, in order to avoid the

image data passing the system bus. On the other hand, DIPS takes the advantage of the Unified Memory Architecture of SGI O2 computer to make the transfer of incoming video data as fast as possible.

**2) Transformation of incoming video image**

Various pixel calculation objects are included in DIPS for making video effects. It is also possible to combine several pixel calculations. For example, combining two of DIPS objects; 'DPXCopy' (DPX is a prefix of pixel handling objects) and 'DPXBlend', a simple motion blur effect can be realized.

**3) Video signal analysis**

DIPS includes several objects that analyze incoming video data. **[Figure 2]**

For example, edge, area and motion detections are employed. The results of these analyses are sent to the outlets of the object, and can be used as controlling parameters for MIDI events, audio signal events, and video image transformations.

**Figure 2: analyzing area of moving body.**

**4) Direct movie rendering**

The 'DMVPlay' object decompresses the movie file on the hard disk and projects it in the GLX contexts in real-time. This object can handle a long movie file. Nevertheless, since it accesses the hard disk, decompresses and projects at the same time it consumes significant amount of calculation time. The following DIPS table object solves this time consumption.

**5) DIPS table object**

DIPS can create its own frame buffer in the main memory space. The 'DIPS_table' object can contain a two-dimensional frame or even a sequence of frames, which can be considered as three-dimensional with the time domain. It fetches the movie file from the hard disk or incoming video images from video node.

**[Figure 3]** If the file is compressed, 'DIPS_table' decodes it first and keep uncompressed images in the memory. This stored images can be used for raster image rendering, for making delayed effects, moving texture, and so on. Due to this table object the speed of this process is maximized.

**6) OpenGL objects**

Most of OpenGL 1.2 functions are introduced in DIPS as objects.

These objects take completely same arguments as common OpenGL functions. Some of GLU and GLUT functions are also available.

Here are some examples:

```
glScalef(2.0,2.0,2.0);
```
⇩
```
DGLScale 2. 2. 2.
```

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```
⇩
```
DGLClear GL_COLOR_BUFFER_BIT GL_DEPTH_BUFFER_BIT
```

```
float vec[] = {0.5,0.7,1.0,1.0};
glLightfv(GL_LIGHT0,GL_DIFFUSE,vec);
```
⇩
```
DGLLight GL_LIGHT0 GL_DIFFUSE 0.5 0.7 1. 1.
```

However, OpenGL functions that require frame buffer, such as 'glTexImage2D', must be combined with DIPS table object.

```
glTexImage2D(GL_TEXTURE_2D,0,GL_RGBA,320,240,0,
    GL_RGBA,GL_UNSIGNED_BYTE,(const GLvoid *)data);
```
⇩
```
DIPS_table aPIXBUF 320 240 GL_RGBA /tmp/aa.mov 1
DGLTexImage2D GL_TEXTURE_2D 0 0 aPIXBUF
```

**Figure 3: video texture mapping on surfaces of cubes.**

**7) Video device control**

SGI O2 and Octane computers have some video hardware device control applications. All the parameters, including for video effects such as blend, wipe, keyer, and so on, which are indicated in the 'Video Control Panel (vcp)'

application, can be controlled from DIPS objects as well.

## DIPS programming in Max patch

DIPS keeps the principal idea of Max application, "providing simple objects so that users can make anything". Almost all the DIPS objects are implemented as bang driven 'control objects' except hash-table definition objects. Therefore, its timing-control is very flexible. For instance, it is possible to let the video operating objects run every 33.3 ms while the graphics rendering objects run twice faster than that speed. All the two dimensional images handled by DIPS objects can be transported among various frame buffers available in that hardware or created by 'DIPS_table' objects freely by connecting lines and giving 'bang' to the DIPS objects.

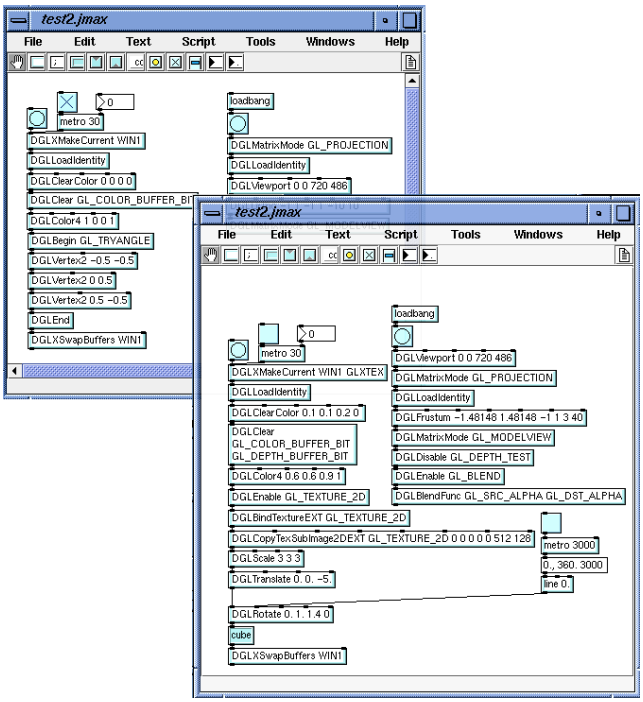


**Figure 4: DIPS programming in jMax patches.**

## Developing customized DIPS object

As same as programming jMax external object, DIPS gives programmers the possibility to make custom objects. Already some of students at our studio have been successfully developing their own DIPS objects.

For instance, Mitsuyo Hashida is developing video analyzing objects, Daichi Ando is writing video pixel calculation objects, and Chikashi Miyama realized objects for particle rendering and for handling imported three dimensional model files created in 3D animation software such as Maya of Alias/Wavefront. **[Figure 5]**

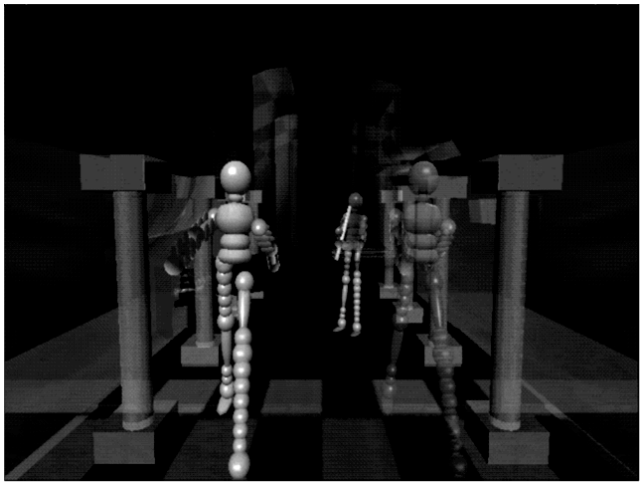

**Figure 5: DIPS handling 3D model files.**

## Creation of works using DIPS

Composers can use DIPS objects as same as common Max objects. Thus, DIPS will make it much easier for composers to realize interaction between visual and audio events; connecting one or several DIPS objects to any signal and control objects, or even other DIPS objects in Max patches. Already several composers including Shu Matsuda, Shintaro Imai and Takayuki Rai, have presented interactive multimedia works using DIPS in concerts. **[Figure 6,7]** Most of those works were written for instrument(s) and a live computer electronic system, which realizes both real-time audio signal processing and real-time visual image processing. The video signal from the camera, that is shooting the performing player on the stage, is sent to SGI computer, and DIPS objects in Max patches transform those images. The transformed output images from the computer are projected on the screen behind the stage simultaneously.

**Figure 6: performance of 'onyx fluctuation' by Shu Matsuda.**

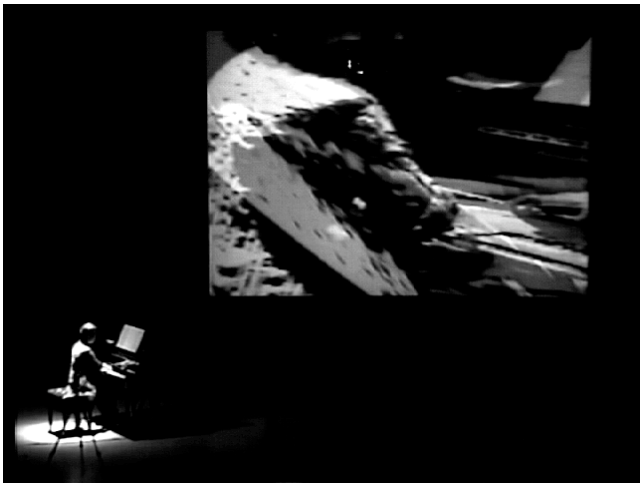

**Figure 7: performance of 'Deep Blue' by Shu Matsuda.**

The parameters for the image transformation (inputs of DIPS object) are often controlled by the incoming audio signal (outputs of Max signal objects) and also Max control objects. Vis-à-vis is also applied by analyzing incoming visual signal in various ways.
In addition to these visual/audio transformations DIPS also presents the opportunity to handle various OpenGL functions and to import the 3D model objects created by the computer animation software and control them in Max patches as mentioned the previous section.

## Further development

The number of DIPS objects is kept expanding. More OpenGL functions will be introduced. And also we are planning to provide the library, just like Jimmy's library for Max/FTS, in order to make DIPS programming more convenient. In the near future, the DIPS package will include the help windows, examples and tutorials as well as the library. Furthermore, we are planning to port the DIPS to the Max/MSP on Macintosh as well as to the jMax for Linux.

## Conclusion

The DIPS unifies the real-time signal processing technique and the real-time image processing technique. It also offers the possibility to control various 3D computer animation tools in Max patch in real-time. It expands the composers' creativity into the interactive visual art field as well as computer music. We are sure this kind of approach will flourish in the coming century and the DIPS will give creators one of starting opportunities for new art works.

## References


[1] Dechelle, F., De Cecco, M., Maggi, E., Schnell,N. "jMax: An Environment for Real Time Musical Applications", in Computer Music Journal, 1999, Beijing, China.

[2] Dechelle, F.,De Cecco, M.,Maggi, E., Schnell,N. "jMax recent developments", in Proceedings of the International Computer Music Conference 1999, Beijing, China.

[3] Dechelle, F., Dececco, M., Maggi, E., Schnell, N., Rovan, B., Borghesi, R. "jMax: A new JAVA-based editing and control system for real-time musical application", in Proceedings of the International Computer Music Conference 1998, Ann Arbor, Michigan, USA.

[4] Dechelle, F., Dececco, M., Maggi, E., Schnell, N., Rovan, B., Borghesi, R. "Latest evolutions of the jMax real-time engine", in Proceedings of the International Computer Music Conference 1998, Ann Arbor, Michigan, USA.

[5] Maggi, E., Dechelle, F., "The evolutions of the graphic editing environment for the IRCAM musical workstation", in Proceedings of the International Computer Music Conference 1996, HK.

[6] Lindemann, E., Dechelle, F., Starkier, M., Smith, B., "The Architecture of the IRCAM Musical Workstation.", in Computer Music Journal, 15(3):41-50,1991.

[7] "The Lurker's Guide to Video", SGI developers ToolBox,

[8] "Open GL on Silicon Graphics System", SGI developers ToolBox,

[9] Danks, M., "The Graphic Environment for Max", in Proceedings of the International Computer Music Conference 1996.

[10] Takashiro, O.,RAI, T.,"The Interactive Multi-media Computer System using SGI and NeXT/ISPW computers" , in Proceedings of the International Computer Music Conference 1996.

[11] Matsuda, S., RAI, T., "A visual-to-sound interactive computer performance system 'Edge'", in Proceedings of the International Computer Music Conference 1995.