INTRODUCTION OF DIPS3 (VERSION 2) FOR MAX/MSP

Takayuki Rai
Lancaster University
Lancaster Institute for the Contemporary Arts
t.rai@lancaster.ac.uk

Chikashi Miyama
Electronic Studio Basel
chikashimiyama@mac.com

Shu Matsuda
Digital Art Creation
Kunitachi College of Music

Takayuki Hamano
Takuto Fukuda
Kunitachi College of Music

Yota Morimoto
The Institute of Sonology,
Royal Conservatory, The Hague

ABSTRACT

DIPS (Digital Image Processing with Sound), was released publicly in 2000 in order to support the creation of interactive multimedia art. However, while it realizes sophisticated real-time image processing in the Max/MSP environment and quite effective interaction between sound and visual events the number of DIPS users have been limited because DIPS is based on OpenGL technology and users are required to know OpenGL programming technique. In this new version, 'DIPS3 version 2 for Max/MSP', while we introduce several new features we attempt to reduce the burden of DIPS programming task for ordinary creators.

1. INTRODUCTION

DIPS was developed in 1997 by Shu Matsuda for SGI computers, and in 2006 we released the third generation of DIPS 'DIPS3' for Max/MSP' environment running on a Macintosh computer. Since DIPS is based on OpenGL technology a certain knowledge of OpenGL is essential. But, nowadays we realize less and less composers and creators practice not only OpenGL programming but also common computer language programming while they intend to employ more and more complicated image processing technique as well as signal processing in real-time. Therefore, in this release, the second version of DIPS3 for Max/MSP, we decided to realize much more user friendly programming environment with two of DIPS sub-patch libraries as well as to introduce Apple's Core Image technology and OpenGL Shading Language programming environment. In this paper we will discuss the following new features of DIPS3 version 2;

- New DIPS Utility Objects
- DIPS Core Image Objects
- OpenGL Shading Language
- DIPS Sub-patch Library
- DIPS Visual Effect Library

2. NEW DIPS UTILITY OBJECTS

Several new DIPS common utility objects are introduced in this release. These include; 'DIPSQTPlayer' object that plays back a Quick Time movie file stored on a hard disk or via Internet, 'DIPSMovieRecord' object that enables to record DIPSWindow rendering event in real-time on a hard disk as a movie file, 'DIPSWindowMixer' object that realizes to mix several sub DIPSWindows, and 'DIPSQCRenderer' object that imports the Quartz Composer files.

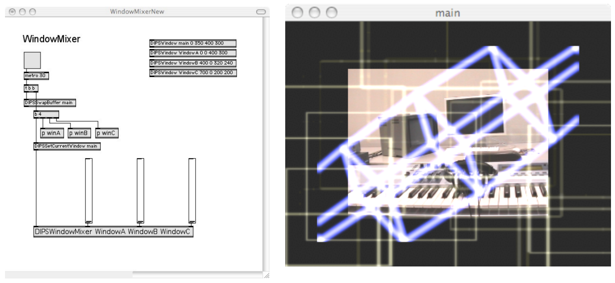


Figure 1. example of DIPSWindowMixer patch and its main window

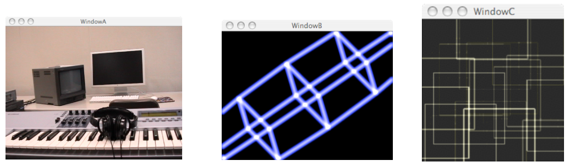


Figure 2. sub DIPSWindows mixed in the patch in Figure 1

We also implemented the capability of rendering text to this release as shown in Figure 3.

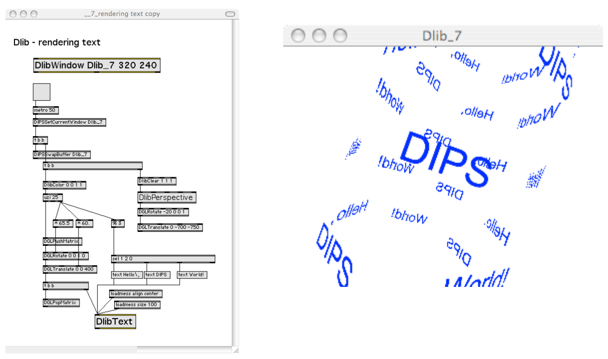


Figure 3. example of text rendering using DlibText object

Beside these new utility objects, the DIPS help file structure has been also revised. By just creating a DIPS object called ‘DIPS’ in a patch users can access to the list of all DIPS objects and to each help file. Now it opens without causing any conflicts of DIPSWindow names as it did before.

3. DIPS CORE IMAGE OBJECTS

Apple's Core Image Technology is implemented as DCI objects (DIPS Core Image objects), thus users can utilize about 100 Core Image Filters in DIPS patches. Since most of DCI objects have a GUI interface to control various parameters it is easy to activate these filters.

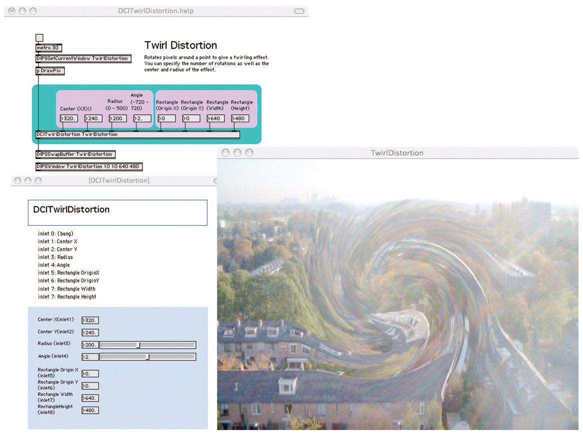


Figure 4. example of DCI object patch

Each DCI object is built with a single DCI object 'DCIFilter'. Because of this structure new and custom made Core Image filters can be implemented immediately without difficulty.

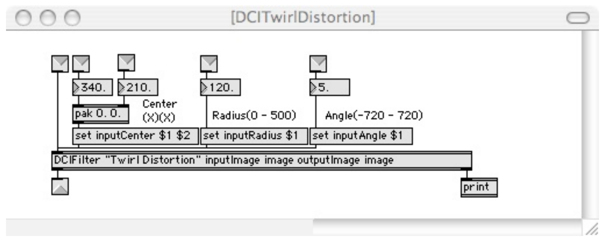


Figure 5. example of DCIFilter sub-patch

4. OPENGL SHADING LANGUAGE

OpenGL Shading Language(GLSL), also known as ‘GL slang’ is a high level shading language, which allows developers to control the graphics pipeline directly without having to use assembly language or hardware-specific languages. The DIPS3 environment offers the user objects to easily edit, compile, and employ GLSL. (see Figure 6.)

4.1. Four essential objects for GLSL

GLSL on DIPS consists of four objects;

‘DGLShaderEditor’, ‘DGLVertexAttrib’, ‘DGLUniform’, and ‘DIPSUseProgram’.

The names of these objects, except DGLShaderEditor, are based on names of OpenGL 2.1 functions.

· *DGLShaderEditor*

A Cocoa-based editor for coding both vertex and fragment shaders. This object is also responsible for creating shader and program object, compiling the codes, attaching shader objects to program object, linking these objects, and detecting errors in the source codes.

· *DGLVertexAttrib*

This object allows the user to modify correspondent attribute-qualified variables declared in the vertex source code.

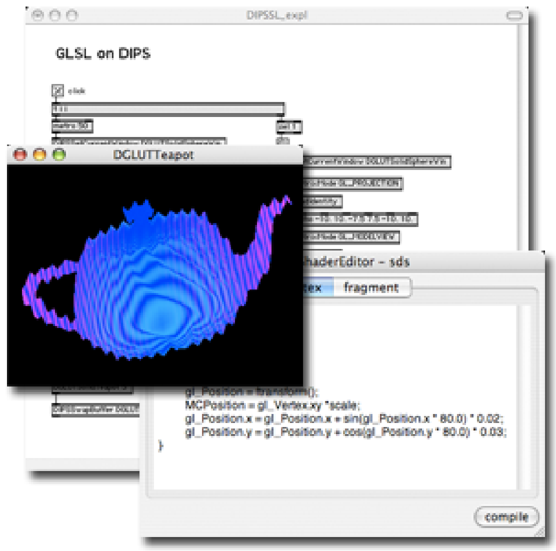
· *DGLUniform*

This object allows the user to modify correspondent uniform-qualified variables declared in the vertex source code.

· *DGLUseProgram*

The compiled GLSL program will be executed when this object receives a bang message.

The first argument of these four objects is the ‘name’ of the shader. If there are several shaders (DGLShaderEditors) in one DIPS patch, ‘DGLVertexAttrib’, ‘DGLUniform’, and ‘DGLUseProgram’ distinguish the correct ‘DGLShaderEditor’ object, by using the defined names.



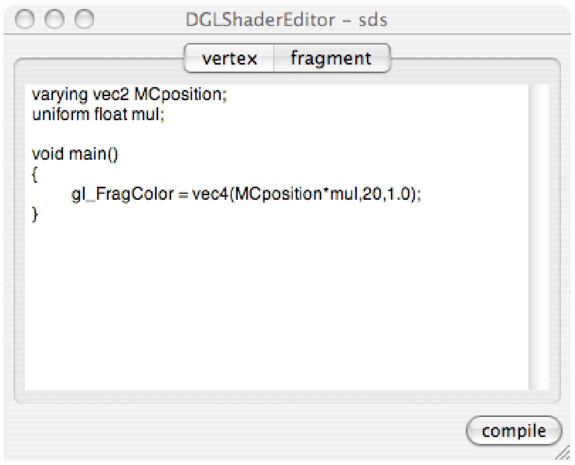
```
gl_Position = ftransform();
MCPosition = gl_Vertex.xy *scale;
gl_Position.x = gl_Position.x + sin(gl_Position.x * 80.0) * 0.02;
gl_Position.y = gl_Position.y + cos(gl_Position.y * 80.0) * 0.03;
}
```

Figure 6. GLSL objects integrated in a DIPS patch

4. 2. Interactive programming interface for GLSL DGLShaderEditor

The DGLShaderEditor allows users to work with the source code interactively. (See Figure 7.)

Usually, a GLSL program consists of two kinds of source-code: vertex shader and fragment shader. With the DGLShaderEditor it is possible to switch between the two source-codes using the tabs on the top of the edit-window. The “compile” button automatically makes two shader objects, compiles the source-codes, attaches them to the program object, and links them.



```
varying vec2 MCposition;
uniform float mul;

void main()
{
	gl_FragColor = vec4(MCposition*mul,20,1.0);
}
```

Figure 7. example of DGLShaderEditor Interface

Moreover, it is not necessary to stop the rendering process to compile GLSL source code. Users are able to check the results immediately after the compile.

5. DIPS SUB-PATCH LIBRARY

While the DIPS GLSL feature may delight experts in OpenGL programming the new DIPS sub-patch library meets a demand from ordinary creators. DIPS Library (Dlib), a new set of DIPS sub-patches, is added to this distribution. Most of Dlib objects have a GUI interface.

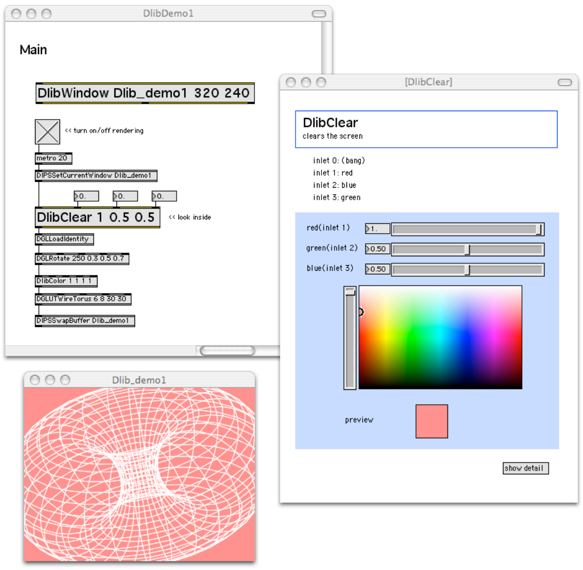


Figure 8. example of Dlib object
with the graphical interface

Therefore, users may be able to control some parameters of the DIPS objects graphically. This library reduces the burden of DIPS programming task significantly.

By using DIPS library objects fewer numbers of DIPS objects as well as less numbers of arguments in each DIPS object are required in the patch. Here is an example of how the Dlib object reduces the number of DIPS objects in the patch. Figure 9 shows the DIPS patch programmed with only ordinary DIPS objects. The rendering result is shown beside it as well.

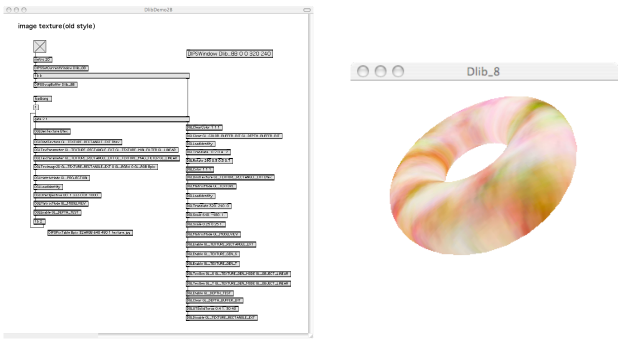


Figure 9. patch example with ordinary DIPS objects

On the other hand Figure 10 shows the patch programmed using Dlib objects to produce the same rendering result as Figure 9.

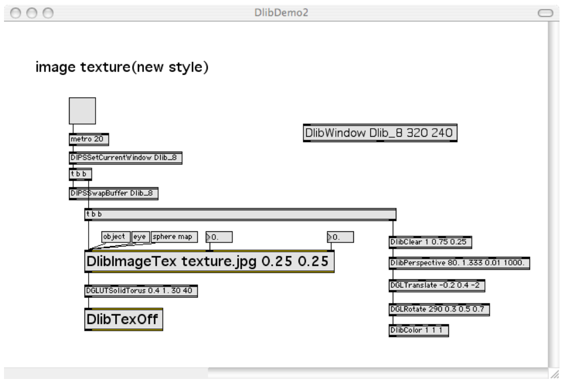

DlibDemo2
image texture(new style)
DlibWindow Dlib_8 320 240
metro 20
DIPSSetCurrentWindow Dlib_8
t b b
DIPSSwapBuffer Dlib_8
t b b
object
eye
sphere map
0.
0.
DlibImageTex texture.jpg 0.25 0.25
DGLUTSolidTorus 0.4 1. 30 40
DlibTexOff
DlibClear 1 0.75 0.25
DlibPerspective 80. 1.333 0.01 1000.
DGLTranslate -0.2 0.4 -2
DGLRotate 290 0.3 0.5 0.7
DlibColor 1 1 1

Figure 10. patch example using Dlib objects

As the two examples show the ordinary DIPS patch requires more than 40 objects while the patch programmed with Dlib objects requires about 15 objects. A burden of DIPS programming task can be reduced dramatically with this Dlib feature.

6. DIPS VISUAL EFFECT LIBRARY

We also distribute the DIPS visual effect library (Dfx) that contains several attractive visual effects such as Gaussian Blur, Radial Blur, Recursive Blur, etc.. Since those Dfx objects are programmed using the OpenGL method they are cheaper and work more smoothly than being realized with pixel calculations. Users can employ sophisticated visual effects with a single Dfx object without any difficulties. Most of them have GUI control possibility as well. And we include these objects as abstract (DIPS sub-patches) so that users can study complex DIPS programming technique by looking inside of Dfx objects as well as Dlib objects.

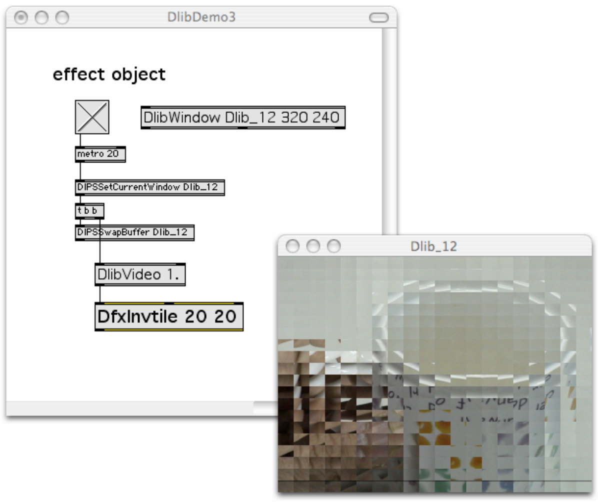


Figure 11. example of DfxInvtile object

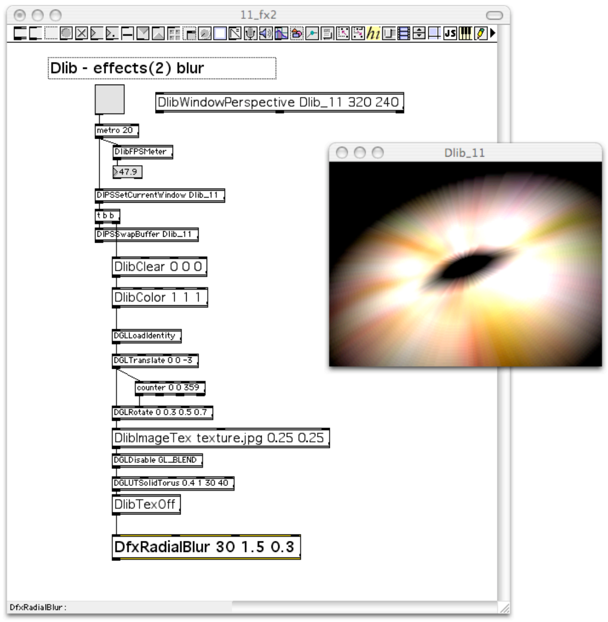


Figure 12. example of DfxRadialBlur object

7. CONCLUSION

This second version of DIPS3 is especially intended for composers and artists who don't know OpenGL so well but like to practice more matured interactive multimedia art with real-time image processing technique. We attempted to realize much more user friendly programming environment than its previous releases while utilizing the advantage of OpenGL technology still. We believe we could achieve the first step of turning around and hopefully this release will attract more creators. And we keep improving DIPS programming environment furthermore, especially by introducing more Dlib and Dfx objects, and more sophisticated examples such as DIPS GLSL realizes.

We hope DIPS supports various creators who are practicing and who wish to practice interactive multimedia art. The DIPS3 version 2 for Max/MSP can be obtained from 'http://dips.dacreation.com'.

8. REFERENCES

[1] Matsuda, S.,Rai, T., "DIPS : the real-time digital image processing objects for Max environment", in Proceedings of the International Computer Music Conference 2000.

[2] Matsuda, S., Miyama, C., Ando, D., Rai, T., "DIPS for Linux and Mac OS X", in Proceedings of the International Computer Music Conference 2002.

[3] Matsuda, S., Rai, T., "DIPS : the real-time digital image processing objects for Max environment", in Proceedings of the International Computer Music Conference 2000.

[4] OpenGL Architecture Review Board, Shreiner, D., Woo, M., Neider, J., Davis, T.,(2005), OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL, Version 2(5th ed.), Addison-Wesley.

[5] OpenGL Architecture Review Board, Editor : Shreiner, D.,(2004). OpenGL(R) Reference Manual: The Official Reference Document to OpenGL, Version 1.4(4th ed.). Addison-Wesley.

[6] Miyama, C., Rai, T., Matsuda, S., Ando, D., Introduction of DIPS Programming Technique, in Proceedings of the International Computer Music Conference 2003.

[7] J.Rost, R., M.Kessenich, J., Lichtenbelt, B., Malan, H., Weiblen, M. Bailey, M.,(2006), OpenGL(R) Shading Language(2nd ed.), Addison-Wesley.

[8] Apple Inc., Apple Developer Connection : Core Image Programming Guide.,(2005) http://developer.apple.com/documentation/GraphicsImaging/Conceptual/CoreImaging/index.html