From Virtual Reality to Digital Arts with Mosaicode

Flávio Luiz Schiavoni
Computer Science Department
Federal University of São João Del Rei, UFSJ
São João Del Rei – MG, Brazil
Email: fls@ufsj.edu.br

Luan Luiz Gonçalves
Computer Science Department
Federal University of São João Del Rei, UFSJ
São João Del Rei – MG, Brazil
Email: luanlg.cco@gmail.com

***Abstract*—Visual programming languages (VPL) simplifies the process of writing a program by letting users create programs through manipulation of graphical elements. Many VPLs focuses on a single domain, simplifying some complicated concepts of a General Purpose Language (GPL), being called Domain Specific Language (DSL). In Digital Art, PureData and EyesWeb are examples of VPLs that allow artists to do advanced projects with basic programming skills. In this paper, we introduce Mosaicode, a Visual Programming Environment to Digital Art domain, presenting some features of digital art and possibilities to implement these features in our VPL.**

***Keywords*-virtual reality; digital arts; mosaicode; VPL; DSL.**

I. Introduction

The cultural convergence of Art, Science and Technology provided to artists a new challenge to create art in a digital universe [1]. This challenge impacted and radically transformed traditional art activities like music, painting, dance and sculpture. Beyond it, entirely new forms of art had arose and are now recognized as artistic practices such as net art, media art, digital installation and Virtual Reality [2].

The intersection between Digital Arts and Virtual Reality remains in the sensitive domain where artists intends to create sensations to the audience. These sensations in arts are commonly created by the use of images and sound, basic ingredients of Virtual Reality, that can now be allied to the possibility of immersion. A next step on art is interactivity and the public interaction, also possible and tangible by Virtual Reality. In fact the essence of what VR is and will be are these three ideas taken together: immersion, interactivity and involvement [3], concepts fully explored in Digital Arts.

Art immersion is different from Games and similar to the immersion of a reader reading a book [4]. According to the same idea, art interactivity has its precursors and echoes in pre-electronic literary and artistic traditions.

This association of Virtual Reality and Digital Arts is shared by other authors. Christiane Paul in Digital Art[5, p. 125] presents Virtual Reality as “a reality that fully immersed its users in a three-dimensional world generated by a computer and allowed them an interaction with the virtual objects that comprise the world”.

The convergence of Digital Art and Virtual Reality fields can be percept in the academic world. Conference topics of interest on both fields have several intersection, like it is possible to see comparing Artech[1] and SVR[2] topics of interest. Both Conferences are interested in topics like Virtual and Augmented Reality; Input devices and interfaces; Audio and Electronic Music; Immersive Environment; Perception and Cognition and others.

On Digital Arts, several applications were developed to attend artistic desires, most part of these applications developed to a specific artistic purpose. On the other side, several APIs and frameworks were developed to attend programming requirement and features of Virtual Reality, most part of these APIs developed to a specific context but in a General Purpose Language.

In this paper we present the development of a Visual Programming Environment called Mosaicode, developed by a Research Group on the Computer Science Department of the Federal University of São João Del Rei. This Environment intends to offer to artists a Visual Programming Language (VPL) to the Virtual Reality specific application domain.

This environment was initially developed to Computer Vision domain and it is been expanded to aggregate more functionalities to Digital Arts, becoming a Specific Domain (Programming) Language (DSL) on this domain.

To implement a DSL a programmer can choose among several implementation approaches. There are guidelines by Mernik [6] with implementation patterns for DSL. Also there are ways to capture the variable parts of application domain using FDL (Feature Description Language), providing a textual description for feature diagrams expressing what given system is composed of [7]. We have taken this guidelines to develop our Visual Programming Environment.

To ensure a better coverage of Digital Arts domain, in this paper we will present some artistic application systems on Digital Arts related to Virtual Reality, bringing a technological point of view about how these application were or could have been developed. Afterwards, we will categorize the systems main features creating a taxonomy of Digital Arts tangential to Virtual Reality. Our next step

[1]Artech is the International Conference on Digital Arts. Conference Topics of Interest are available on http://2017.artech-international.org/call-for-papers/.

[2]SVR is the Symposium on Virtual and Augmented Reality (SVR), the premier conference on Virtual and Augmented Reality in Brazil. Symposium topics of interest are available on http://usuarios.upf.br/~rieder/svr2017/submissions.html.

978-1-5386-3588-9/17 $31.00 © 2017 IEEE
DOI 10.1109/SVR.2017.33

is to present some APIs and Libraries that could be used to develop these features.

II. Artistic Applications Related to Virtual Reality

In this section we will present some artistic application systems on Digital Art related to Virtual Reality. We do not intend to be exhaustive in this listing and more arts projects related to Virtual / Augmented Reality can be found in [5, p. 133], [1], [8], [9] and [2].

We chose some projects related to Digital Art that have different approaches and that are related to Augmented / Virtual Reality. We will present the Art projects and also a technical briefing about how the projects could be implemented giving the project presentation not an artistic evaluation but a technological analysis.

A. *Reactable*

The Reactable is an electro-acoustic musical instrument, controlled by several performers simultaneously, that works moving physical artifacts on the table surface producing sound by audio synthesis [10] [11]. The musician playing the Reactable also has a visual feedback about how the pieces are connected and how they will sound. A Picture of musicians playing the instrument is presented on Figure 1.

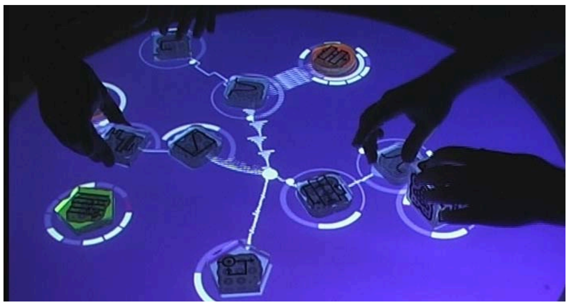


Figure 1. Four hands at the ReacTable[11]

This kind of Tangible User Interface is developed merging computer vision and sound processing as System's input and image and sound synthesis as output. Also, to allow communication between Instruments, it is necessary to have a computer network communication layer. The main schematics of a project like Reactable is presented on Figure 2.

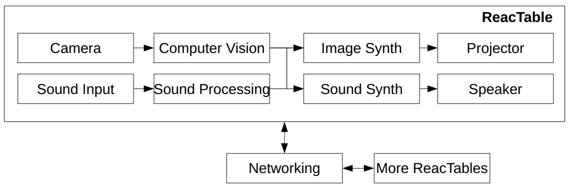


Figure 2. Reactable schematics

B. *Pixel*

Pixel [3] is a contemporary dance project created by the French dance company Käfig. The dance show presents eleven dancers in a virtual and living visual environment [12]. The dance company is directed by Mourad Merzouki and the dancing includes street dance, body contact, circus, energy, poetry, fiction and illusion[13]. In Pixel, the stage is a morphic space of interaction that is transformed in several immersive environments bringing a new level of interactivity to dance. The environment reads the dancers position and generates the virtual environment on the fly allowing imprecision and improvisation.

Figure 3. Pixel. Image from the video [12]

This kind of living visual environment may be created using two cameras and two projectors and a silk screen on the stage. One camera, positioned in front of the stage, will be responsible to track dancers position from left to right while the other camera, positioned on the top of the stage, will be responsible to track dancers position from back to front. Both camera should exchange data to a Image Synthesizer that creates two projections, one for each projector. Projectors should be also positioned in front of and on the top of the stage, creating a 3D immersive environment to dancers. This kind of environment / interface can be developed using the schematics presented on Figure 4.

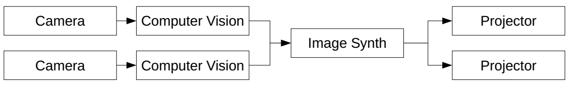


Figure 4. Pixel Schematics

C. *Glasbead*

Glasbead [4] is an "online art work that enables up to 20 simultaneous participants to make music collaboratively via a colorful three-dimensional interface" [8, p.54]. This instrument, gaming, toy and multi-user persistent collaborative musical interface is a Virtual Reality environment that allows players to play shared sound files and create soundscapes by the means of a Graphical User Interface. The GUI, presented on Figure 5 consists of a "rotating,

[3]Project website: https://ccncreteil.com/spectacles/pixel.
[4]Project website: http://www.cityarts.com/glasbeadweb/.

circular structure with stems that resemble hammers and bells"[14]. Users can import and exchange sound sample files into the bells and create rhythmic musical sequences by flinging the hammers into the bells.

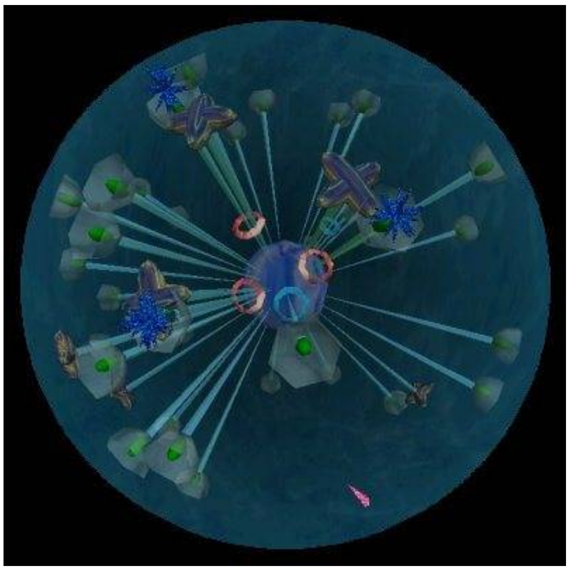


Figure 5. Glasbead. Image from the project website

This project is based on Network interactivity to create a virtual stage where remote participants can make music jam. The local interactivity in the Glasbead environment is made by a GUI. The GUI uses a 3D image synthesizer to control music files and the sequencer. A Sound processing API is used to loop samples and a network connection to send and receive newtork data and create remote interaction. The schematic of this system is presented on Figure 6.

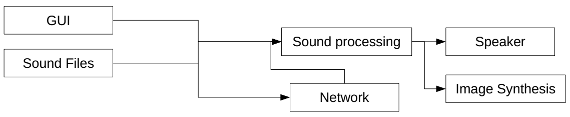


Figure 6. Glasbead Schematics

D. Beyond Manzanar

Beyond Manzanar[5] is an interactive 3D virtual reality artwork. According to the author, "It is shown as a room installation with the image projected life-sized on a large screen. One user at a time can navigate through the 3D environment in first-person viewpoint using a simple joystick; other can watch and share the experience" [15].

This Virtual Reality Environment is a realistic reconstruction of Manzanar Internment Camp in California, USA, used to host Japanese-Americans during the World War II post Pear Harbor and to threat Iranian-American in the wake of the hostage crisis of 1979-1980. This realistic environment is mixed with imagined landscapes of Japanese and Iranian gardens in a kind of interactive scenario like in Doom or Quake games [16]. This Scenario is depicted on Figure 7.

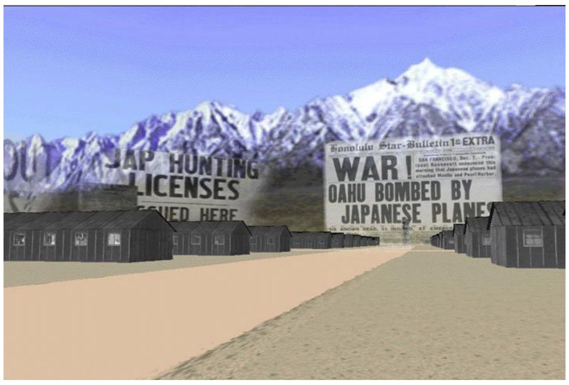


Figure 7. Manzanar. Image from the project website

This installation uses no paraphernalia like glasses, gloves or visors but a huge life size screen to create the immersion feeling. The system also uses music and ambient sounds to be more realistic and to help guiding the narrative. A schematic presenting this kind of art installation is depicted on Figure 8.

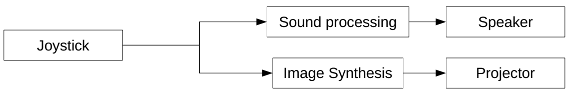


Figure 8. Manzanar Schematics

E. A Taxonomy of Digital Art Projects

The analysis of projects presented thus far may indicate some common implementation and a technological intersection on these projects. Our first observation is about a common data type. All projects use two data basic data type: **Image** and **Sound**, presented by Camera / Projector and Sound Input / Sound File / Speakers.

The possibility of improvisation and free interactivity instead of a preprogrammed interaction, leads to a real time sound and image synthesis. The only Art Project without this kind of interactivity is "Beyond Manzanar", that used VRML to pre-create the Virtual environment. These interactivity level can be seen by the Computer Vision feature present in some projects.

Another feature common to all projects is how to **Control** the interactivity. Common approaches, like "Beyond Manzanar" uses a traditional interface, like a joystick. Other projects uses sound and image processing to control the interactivity. Another common control is the GUI, sometimes necessary to set up the system.

To create interactivity beyond the local place, the **Network communication** is another feature present in some projects presented.

This analysis leads us to the following list of common features on Digital Art projects:

1) Image
    a) Processing

[5]Project website:http://www.mission-base.com/manzanar/.

        b) Synthesis
        c) Analysis
    2) Sound
        a) Processing
        b) Synthesis
        c) Analysis
    3) User Control
        a) GUI
        b) Common Interfaces (MIDI, Joystick, . . .)
        c) Sensors
    4) Networking

III. Related Work

There are some available visual programming language (VPL) environment to the specific domain of Digital Arts and Virtual Reality. These environments satisfy the list of common features presented before and bring the possibility of develop Art projects similar to those presented before. The two related tool presented here are open source and free to download.

Other related tools, like Max/MSP[6] and ISADORA[7] are also related but not open source or free to download.

Another interesting Programming environment on the same domain is the Processing Language[8]. Despite it is a easy-to-use programming environment, it uses textual programming instead of visual programming like the other related tools.

A. Pure Data

Pure Data [9] is a Visual Programming Environment for Sound and Music that plays host to GEM environment[17] to 3D graphic processing [18] [19]. This environment is extensible by user plugins, called externals, and several libraries extend Pure Data allowing integration with network communication, Arduino Sensors, wiimote control, Kinect, OSC messages, Joystics and others. Pure Data also has native interface with MIDI devices. A screen shot of Pure Data Visual programming is presented on Figure 9.

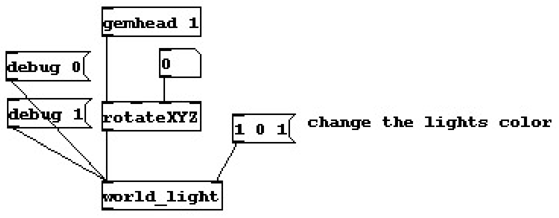


Figure 9. Pure Data patch. Image from the project website.

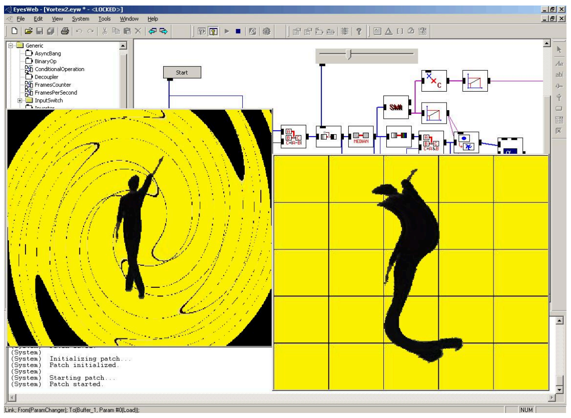
Pure Data is an open source project and it is a tool widely used to Digital Art installation and projects.

Pure Data is also wrapped into a C library called libpd[20], to be embedded as a sound engine into other systems.

[6]Project Website: https://cycling74.com/products/max/
[7]Project Website: https://troikatronix.com/
[8]Project Website: https://processing.org
[9]Project Website: http://puredata.info.

B. EyesWeb

Eyesweb[10] is a project focused on real time analysis of body movement and gesture[21]. According to the authors, such information can be used to create and control sounds, music, visual media and to control actuators. Eyesweb has native MIDI interface and network communication. A screenshot of EyesWeb Visual programming is presented on Figure 10.



```
(System)
(System)  Initializing patch...
(System)  Patch initialized.
(System)
(System)  Starting patch...
(System)  Patch started.
```

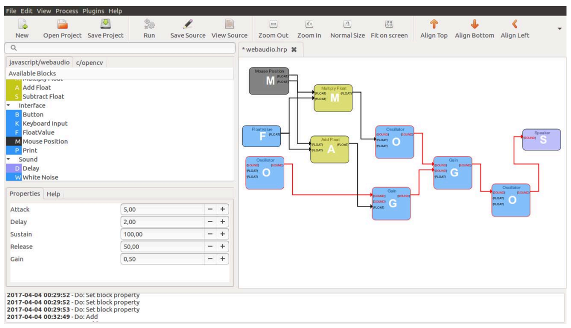
Link; From(ParamChanger); To(Buffer_1, Param #0(Load));

Figure 10. Eyesweb mapping. Image from the project website.

Eyesweb is also an open source application and an open platform [22] and it is widely used to video mapping installations.

IV. The Mosaicode environment

The Mosaicode project is an open source graphical environment to implement Specific Domain (Programming) Languages by Visual Programming Languages (VPL), presented on Figure 11. Mosaicode is a fork of the Harpia project, developed by S2i - Industrial Intelligent Systems, a research group on Machine Vision from Systems and Automation Department (DAS) and from Statistics and Informatics Department (INE) at Federal University of Santa Catarina (UFSC).



Properties	Help
Attack	5,00
Delay	2,00
Sustain	100,00
Release	50,00
Gain	0,50

```
2017-04-04 00:29:52 - Do: Set block property
2017-04-04 00:29:52 - Do: Set block property
2017-04-04 00:29:53 - Do: Set block property
2017-04-04 00:32:49 - Do: Add
```

Figure 11. Mosaicode Programming Environment.

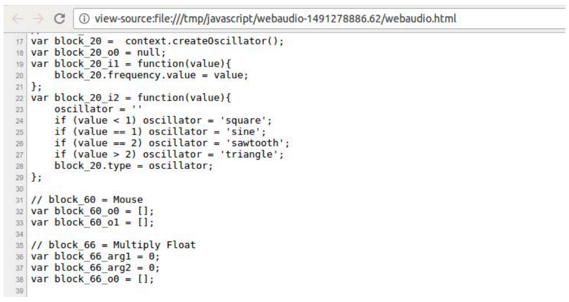
Different from related tools, that are Environments which execute the visual code, Mosaicode is a code generation tool. Thus, the application generated on the

[10]Project website: http://www.infomus.org/eyesweb_ita.php.

environment can be used as a standalone application with code optimization to grant performance issues. The Mosaicode tool can also execute the generated source code running the command line to perform this task. For this reason, probably several users do not understand the application as a code generation tool and can feel it as a Visual programming environment, like Pure Data or Eyesweb.

Initially, this environment was focused on computer vision specific domain based on the OpenCV library. Functions of computer vision were mapped onto Blocks, which could be combined by non-programmer users in order to simplify the creation of computer vision applications. Despite the good usage of the computer vision Blocks to Digital Art or Virtual Reality, modern art application often requires integration with other libraries like network communication, audio, MIDI or sensors management. For this reason, the environment had been expanded to work with other functionalities based on other libraries.

Our first implementation beyond C code generation to openCV library was to implement a set of plugins to develop javascript / webaudio applications. An generated Source Code to Javascript / webaudio is depicted on Figure 12

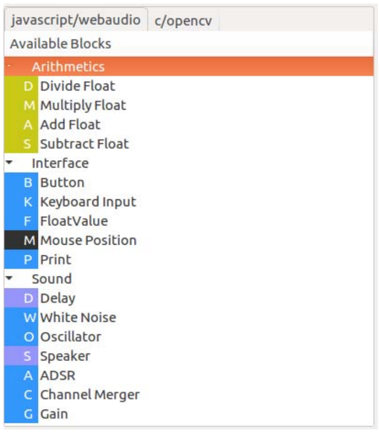

view-source:file:///tmp/javascript/webaudio-1491278886.62/webaudio.html

```
var block_20 =  context.createOscillator();
var block_20_o0 = null;
var block_20_i1 = function(value){
    block_20.frequency.value = value;
};
var block_20_i2 = function(value){
    oscillator = ''
    if (value < 1) oscillator = 'square';
    if (value == 1) oscillator = 'sine';
    if (value == 2) oscillator = 'sawtooth';
    if (value > 2) oscillator = 'triangle';
    block_20.type = oscillator;
};

// block_60 = Mouse
var block_60_o0 = [];
var block_60_o1 = [];

// block_66 = Multiply Float
var block_66_arg1 = 0;
var block_66_arg2 = 0;
var block_66_o0 = [];

```

Figure 12. Mosaicode Generated Source Code

A. *The programming Metaphore*

Our programming environment are based on Blocks. A **Block** is the minimal code piece in the workflow that represents a code abstraction of a functionality. Blocks are grouped into categories that represent groups based on the same processing principle, like Arithmetic and Logical Operations, Filters and Color Conversion, Features Detection and so on. Figure 13 presents some categories to Javascript / Webaudio programming.

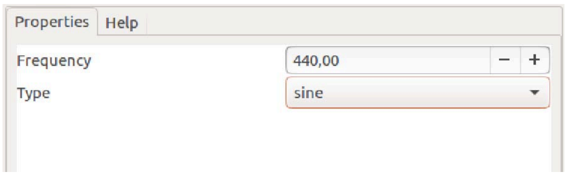

javascript/webaudio | c/opencv

Available Blocks

- Arithmetics
  - D Divide Float
  - M Multiply Float
  - A Add Float
  - S Subtract Float
- Interface
  - B Button
  - K Keyboard Input
  - F FloatValue
  - M Mouse Position
  - P Print
- Sound
  - D Delay
  - W White Noise
  - O Oscillator
  - S Speaker
  - A ADSR
  - C Channel Merger
  - G Gain

Figure 13. Mosaicode Javascript/webaudio categories

A Block can have static properties or dynamic properties. Static properties are configured by a setup window in the environment where its basic features like size, width, height, color, angle, matrix and other feature can be changed. Figure 14 presents the static properties of an audio oscillator.


Properties | Help

Frequency 440,00

Type sine

Figure 14. Mosaicode Block static properties

The dynamic properties of a Block is represented by an input port. The output processing of a block is represented by an output port. Each Block can have inputs and outputs ports that can be connected to combine different blocks into **Diagrams**. Inputs and outputs defines the programming data flow and can be connected by **Connections**.

The collection of interconnected Blocks defines a **Diagram**, like depicted on Figure 15. Each Diagram generates an individual Source Code and can be saved in a special application file format.

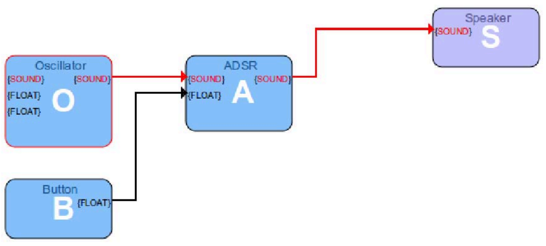


Figure 15. Mosaicode Diagram

B. *Extending Mosaicode with plugins to Digital Arts*

The idea to provide plugin support is to allow users to develop their own modules for Mosaicode, adapting the application to their own needs. Thus, we developed a Plugin manager interface to create new plugins including the possibility to create new plugin ports and Code templates that allows the user to define how Blocks and Connections can be combined to create a new source code.

Once that we listed some common features to Digital Art projects, is it possible to enumerate a set of APIs related with these features that could be implemented on Mosaicode and grant to this tool a better coverage on Digital Art domain.

Mosaicode already implements several functionality on **Image processing and analysis** using openCV. The open source computer vision library, OpenCV, is aimed at providing the tools needed to solve computer-vision problems. This library gives support to work with image analysis and processing, allowing users to benefit from GPU acceleration through module implemented using CUDA [23]. OpenCV also has some basic features to image synthesis wrapped as a Mosaicode plugin and some Artificial intelligence features still not wrapped as a Mosaicode plugin.

A good posibility to implement **Image Synthesis** is to wrap functionalities from OpenGL. OpenGL (Open Graphics Library) is an open source library to Image Synthesis widely used in art projects and Virtual Reality environment. This API allows a programmer to specify the objects and operations through the manipulation of several procedures and functions to produce high-quality graphical images, specifically color images of three-dimensional objects [24].

Our first implementation of **Sound** and audio API was developed in javascript language using the webaudio API. Despite of been really interesting to create web art, this implementation can not be integrated to C language projects. A better solution to integrate Audio to the former plugins is to use PortAudio[25] library.

To allow **User Control** we can choose Gtk API to GUI. Several Musical Interfaces uses MIDI protocol as a common interface. The PortMidi library[26] is a cross platform API to implement MIDI interface with all sorts of MIDI devices.

Lastly, the **Networking** capability can be developed using OSC. The Open Sound Control (OSC) is a protocol useful for a wide variety of networking applications over wide-area networking ranges [27]. This open source library is present in:

- Computer programming language: Director, Flash;
- Web graphics/animation systems: Bidule, Chuck, Common Music CPS, Intakt, Max/MSP, Open Sound World, Pd, SuperCollider, Reaktor, Traktor;
- Sensor/gesture: EtherSense, Gluion, IpSonLab Kroonde, Lemur, Smart Controller, Teabox, Toaster;
- Idiosyncratic control-message-generating software: EyesWeb, Picker, SonART, SpinOSC.

V. Comparison between related tools and Mosaicode

Mosaicode, Pure Data and Eyesweb have several similarity. They are open source tools, free to download and extensible by plugins allowing the user to customize the tool.

A deeper comparison of Pure Data, Eyesweb and Mosaicode is not an easy task. Firstly because they have different approaches as a Visual Programming Environment. While Pure Data and Eyesweb are DSL/VPL, Mosaicode is a code generator that intends to give user an easy and visual interface to code. The final product of Mosaicode is the source code of a standalone application that can be adapted, optimized and ported to other tools. Pure Data and Eyesweb final product can be only executed on these programming environment engine, including libpd applications on the case of Pure Data. This first difference can be noticed when using the Javascript plugins set. The generated code can be embedded in any web page since it is not necessary to have Mosaicode to execute it. This is not possible with Pure Data or Eyesweb.

Another huge difference is about maturity. Pure Data and Eyesweb are mature tools and several art projects were developed using these tools. They have decades of experience and had bean optimized since then. On the other hand, Mosaicode is a new tool that tries to learn from these tools, specially from the Art Domain, and bring expertise and experience from recognized libraries and frameworks to Art Domain, like openGL or openCV.

VI. Conclusion

Virtual Reality and Digital Arts are related fields with several common features. In this paper we introduced Mosaicode, a Visual Programming Environment. Initially developed to Computer Vision, this tool has been expanded to a more general tool on Digital Arts.

To give the next step and give a better coverage on digital arts field, we analyzed a set of art projects with a technical point of view trying to extract from these projects the common features of this domain.

Once we extract the projects features, we proposed an initial taxonomy of digital art programming systems, categorizing the common functionalities in groups.

Based on these groups, we proposed open source libraries / APIs that might implement these functionalities and could be wrapped into plugins to integrate the Mosaicode environment.

Our Future works include a analysis of the listed features going deeper in each feature and listing what kind of functionality can be useful for each specific feature.

References

[1] O. Grau, *Virtual Art: from illusion to immersion.* MIT press, 2003.

[2] B. Wands, *Art of the Digital Age.* Thames & Hudson, 2007.

[3] J. F. Morie, "Inspiring the future: Merging mass communication, art, entertainment and virtual environments," *SIGGRAPH Comput. Graph.*, vol. 28, no. 2, pp. 135–138, May 1994. [Online]. Available: http://doi.acm.org/10.1145/178951.178973

[4] M.-L. Ryan, *Narrative As Virtual Reality: Immersion and Interactivity in Literature and Electronic Media*. Baltimore, MD, USA: Johns Hopkins University Press, 2001.

[5] C. Paul and C. Werner, *Digital art*. Thames & Hudson London, 2003.

[6] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM computing surveys (CSUR)*, vol. 37, no. 4, pp. 316–344, 2005.

[7] T. Kosar, P. E. Martı, P. A. Barrientos, M. Mernik *et al.*, "A preliminary study on various implementation approaches of domain-specific language," *Information and software technology*, vol. 50, no. 5, pp. 390–405, 2008.

[8] M. Tribe, R. Jana, and U. Grosenick, *New media art*. Taschen London and Cologne, 2006.

[9] "Exhibiting artists," *Leonardo*, vol. 35, no. 5, pp. 581–582, oct 2002. [Online]. Available: https://doi.org/10.1162%2F002409402320774411

[10] M. Kaltenbrunner, S. Jorda, G. Geiger, and M. Alonso, "The reactable*: A collaborative musical instrument," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE'06. 15th IEEE International Workshops on*. IEEE, 2006, pp. 406–411.

[11] S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner, "The reactable: Exploring the synergy between live music performance and tabletop tangible interfaces," in *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, ser. TEI '07. New York, NY, USA: ACM, 2007, pp. 139–146. [Online]. Available: http://doi.acm.org/10.1145/1226969.1226998

[12] A. Mondot and C. Bardainne, "Pixel," https://vimeo.com/114767889, 2014.

[13] A. Davidson, "Ontological shifts: Multi-sensoriality and embodiment in a third wave of digital interfaces," *Journal of Dance & Somatic Practices*, vol. 8, no. 1, pp. 21–42, 2016.

[14] C. Paul, "Renderings of digital art," *Leonardo*, vol. 35, no. 5, pp. 471–484, oct 2002. [Online]. Available: https://doi.org/10.1162%2F002409402320774303

[15] T. Thiel, "Beyond manzanar: Constructing meaning in interactive virtual reality," *R GN E P CIES*, p. 73, 2001.

[16] M. W. Smith, *The total work of art: from bayreuth to cyberspace*. Routledge, 2007.

[17] M. Danks, "Real-time image and video processing in gem." in *ICMC*, 1997.

[18] M. Puckette *et al.*, "Pure data: another integrated computer music environment," *Proceedings of the second intercollege computer music concerts*, pp. 37–41, 1996.

[19] M. S. Puckette *et al.*, "Pure data." in *ICMC*, 1997.

[20] P. Brinkmann, P. Kirn, R. Lawler, C. McCormick, M. Roth, and H.-C. Steiner, "Embedding pure data with libpd," in *Proceedings of the Pure Data Convention*, vol. 291. Citeseer, 2011.

[21] A. Camurri, S. Hashimoto, M. Ricchetti, A. Ricci, K. Suzuki, R. Trocca, and G. Volpe, "Eyesweb: Toward gesture and affect recognition in interactive dance and music systems," *Computer Music Journal*, vol. 24, no. 1, pp. 57–69, 2000.

[22] A. Camurri, P. Coletta, A. Massari, B. Mazzarino, M. Peri, M. Ricchetti, A. Ricci, and G. Volpe, "Toward real-time multimodal processing: Eyesweb 4.0," in *Proc. Artificial Intelligence and the Simulation of Behaviour (AISB) 2004 Convention: Motion, Emotion and Cognition,*. Citeseer, 2004, pp. 22–26.

[23] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Real-time computer vision with opencv," *Communications of the ACM*, vol. 55, no. 6, pp. 61–69, 2012.

[24] M. Segal and K. Akeley, "The opengl graphics system: A specification (version 1.1)," 1999.

[25] R. Bencina and P. Burk, "Portaudio-an open source cross platform audio api." in *ICMC*, 2001.

[26] R. Bencina, P. Burk, and R. Dannenberg, "portmidi-platform independent library for midi," 2007.

[27] M. Wright, "Open sound control: an enabling technology for musical networking," *Organised Sound*, vol. 10, no. 03, pp. 193–200, 2005.