LİVE CODİNG FOR INCORPORATİNG SELF-EXPRESSİON

SELÇUK ARTUT [*]

Abstract: This paper is exploring live-coding systems as self-expressive tools in utilizing creative activities that are focusing mainly on audio-visual embodiments. Live coding, as a performance tool is a particular activity that incorporates writing code on the fly with the shared experience of the activity being involved. Many creative platforms are equipped with exuberant tools available to the users to process materials. On the contrary live coding platforms are formed as blank canvases like a tabula rasa so that users may decide what to include for their requirements. As a consequence live coding motivates performers to develop authentic methods to convey their expressive ideas.

Key Word: *Live coding, Programming, Creative Coding, Human-Computer Interaction*

1. Introduction

Competency in programming languages are becoming necessities for creative artists to investigate and digest in order to explore alternative tools to articulate various creative expressions of abstractions. The explosion of Digital Revolution in creative studies enabled the users to be equipped with the liberty of undertaking limitless opportunities of making trials and errors. As a consequence of such continuous building processes, the final outcome becomes a unique reflection of the multitude contingencies. According to G.Wang and

[*] Sabanci University, Visual Arts and Visual Communication Design [sartut@sabanciuniv.edu]

INTERNATIONAL BALKAN UNIVERSITY
ICSSH2016
INTERNATIONAL CONFERENCE ON SOCIAL SCIENCES AND HUMANITIES

P.R.Cook (2004) most of the design programs are limited to off-line development and preparation, leaving only the finished program to "go live".

In this regard from an observer point of view, experience of confronting a final end result becomes divorced from the creative process and therefore an observer becomes a bystander for the particular production. In order to illustrate this proposition with an example when laptop performances are examined, audience becomes separated from witnessing the actuality of the performing process and an absolute attention is demanded from the participants. On the other hand in many laptop performances it is observed that presentation of visuals are additionally joined into the spectacle. However even in this setting participants are considered to be in the status of being passive observers of a performance, starring at a surface locked in one place. From an artist's point of view these aforementioned detachments are concealing the experience of seeing the artist's authentic talent in performing the act.

Although commonly used assistive tools and creative applications are becoming more and more affluent in providing diversified components and features, yet users are experiencing the difficulty in orienting themselves within this exponentially becoming sophisticated environments and complicated interfaces. On the other hand in order to handle this situation most of the applications are providing assistive tools to write scripts to enhance the creative interaction between the user and the system. This allows many users to diverge from the given settings and the frameworks provided on the inter(sur)face, and it contributes to the facilitation of improved customized expressions. However even in this setting users are yet constrained within the limits of the given tools and platforms (Collins et al 2003). On the contrary an alternative form of overcoming such limitations is to build personal tools from a scratch. Progressively a decent number of programmers have been concentrating on producing self-expressive tools that are based on elemental structures to enable the users to be independent of given schemes (Collins 2003).

As a consequence Live Coding performances have become an expressive form of creativity occurring in public spaces. The live coders are able to program things on the fly and apply changes dynamically while the code is still running (Brown et al 2007). In terms of the creativity involved, live coders have backgrounds in several areas including music, programming and performing arts. Therefore coding becomes an instrumental tool to create and express (Wang 2014). Widely observed, during live coding activities performers are creating audio-visual experiences real time and the tools are accessible to the audiences to observe (Brown et al 2009). Alternative to DJ performances, the live coders are using low-level programming environments such as Supercollider, Sonic Pi, Max/MSP, Impromptu etc. running on computer environments and presenting their activity of development live on stage with the use of projection surfaces. Despite the fact that there exists the potential risk of failing to program correctly and crushing the system to perform on stage, the live coders are courageous enough to overcome such difficulties with depending on their experiences and talents. As Collins (2003) states risk taking becomes an important part of the work.

2. DEvelopment PLATFORMS

Due to their flexible nature, software developers have engineered live coding development platforms with proliferating features. Although some of the available development platforms have ceased to exist, currently there are over 30 different platforms provided by independent developers, research groups and commercial companies etc. (Retrived from http://toplap.org/wiki/ToplapSystems.)

Live Coding platforms can be categorized under two main groups: Graphical Building and Text-Based Coding. In Graphical Building environments users are given a set of objects to configure visually by combining chords on a patch system. Some of the commonly used Graphical Live Coding platforms are MaxMSP/Jitter, Reaktor, vvvv and Pure Data.

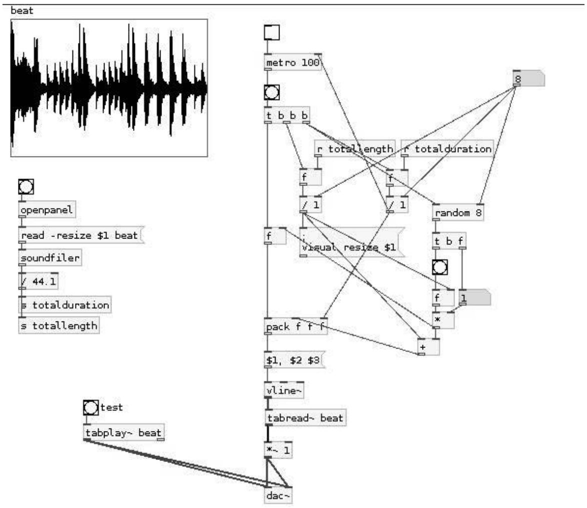

Figure 1. (a) A Pure Data Patch

Combination of chords connected to objects provided within the development platform enables users to avoid holding comprehensive background in advanced programming structures. Live coders are able to work with visual elements to organize their sound and image composing structures. As a consequence graphical building platforms have considerable advantages in terms of their usability and learnability. Since the users are preferably selecting the preexisting objects to combine, it is less likely to make mistakes in generating these objects. Certainly occurrence of incorrect patching combinations between the objects is still prone to ground failures in the system. Moreover graphical building platforms are capable of providing developers with smooth parametric controls triggered by the interactive objects. Subsequently while the developers are employing musical structures with the use of graphical objects, at the same time they are creating interactive interfaces for their performances.

On the contrary text-based coding environments are facilitating their users with more rigid contents empowered by strict wording formats. Typing mistakes in such development environments are not tolerated, so developers need to have competent experience about the coding standards of a particular platform. In addition to typical way of developing computer programs that are written with lines of codes, several musical concerns such as temporal relations, harmony etc. are additionally included in the production process.

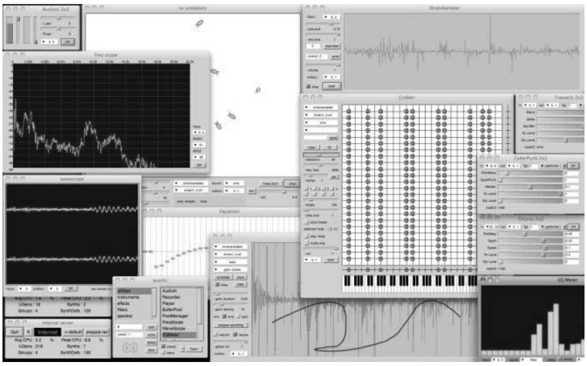

Figure 2. (b) Supercollider Environment

Pros and cons of the two approaches are distinctive in some respects. Comparatively Text-Based Coding environments are more straight-forward in understanding their structure, since the given code can be followed from top to down, function call to function, pointer to instance approaches more systematically. However graphical building structures are difficult to follow due to their multi-directional node architecture. Some patches may become incomprehensible because of their disorderly organization. On the other hand a Text-Based Coding system may provide its users with opportunities to consider far generative approaches when compared to a Graphical Building system because of its flexibility in using procedural functions.

3. Programming PARADIGMS

It is widely observed that several different programming approaches are maintained within different algorithmic live coding situations. Platforms such as Supercollider, SonicPi allow the programmers to integrate structural object oriented models with functional programming techniques. In practice the nature of live coding requests faster results while building up a musical idea due to expectancy of compositional structures. Hence hybrid techniques based on functional programming utilizations are providing its users enhanced skills in building things quicker with relatively small amount of typing. In contrast to structural programming functional programming avoids the use of statements, and uses expressions instead. In that respect in functional programming creating musical gestures becomes analogous to live stage situations for the performing artists.

4. ConclusIon

With the extensive use of technological tools provided to creative minds, the users are seeking alternative methods to deliver their ideas in distinct forms. In that respect live coding as a creative approach distinguishes itself as a dialectic system based on human intelligence and computer programming. From an experience point of view being exposed to a live coding session constitutes a welcoming aura around the deployment of artistic skills.

* Dr. Selçuk Artut is Professor of Visual Arts and Visual Communication Design at Sabanci University, Istanbul. His research is mainly focused on contemporary media practices on human technology interactions. Artut is the author of Datareality (Atropos, 2013).

References

[1] Wang, G., & Cook, P. R. (2004, June). On-the-fly programming: using code as an expressive musical instrument. In Proceedings of the 2004 conference on New interfaces for musical expression (pp. 138-143). National University of Singapore.

[2] Collins, N., McLean, A., Rohrhuber, J., & Ward, A. (2003). Live coding in laptop performance. Organised sound, 8(03), 321-330. ISO 690

[3] Collins, N. (2003). Generative music and laptop performance. Contemporary Music Review, 22(4), 67-79.

[4] Brown, A. R., & Sorensen, A. C. (2007). aa-cell in practice: An approach to musical live coding. In Proceedings of the International Computer Music Conference (pp. 292-299). International Computer Music Association.

[5] Brown, A. R., & Sorensen, A. (2009). Interacting with generative music through live coding. Contemporary Music Review, 28(1), 17-29.