Live Coding Tools for Choreography: Creating Terpsicode

Dr. Kate Sicchio
Virginia Commonwealth University
ksicchio@vcu.edu

Zeshan Wang
Virginia Commonwealth University
wangz24@mymail.vcu.edu

Marissa Forbes
Virginia Commonwealth University
forbesmc@mymail.vcu.edu

 ICLC 2020, February 5-7, 2020, University of Limerick ,Limerick, Ireland

Abstract

Terpsicode is a developing mini programming language for live coding dance performance scores. This paper explores the process of creating a choreographic patterning language using images, and discusses the creation, capturing, and naming of movement. It also reflects on the premiere performance using this system with a live coder and improvising dancer and how score-making with code may be translated by a performer. The final result of this venture seeks to provide a computer language for choreography that utilises dance terminology alongside visual performance scores that may be used within various improvising settings.

Introduction

This paper discusses an ongoing process in the development of a programming language for choreography. It explores the idea of sampling movement through still images as a way of forming discrete units for creating patterns over time. These patterns in turn become composed through algorithmic processes. The main goal of the language is to be utilized within live coding practices, as seen in previous work around choreography, and in real time score making[1]. The language itself is meant for a choreographer to utilise when creating work, building off of choreographic tools by artists such as Myriam Gourfink (Gourfink, 2013), Wayne McGregor or WIlliam Forsythe (de Lahunta, 2014).

Terpsicode will allow for computers and algorithmic processes to become more integrated into choreographic practices. The unexpected outcomes of image concatenation will inspire artists to break out of habitual movement patterns, which creates new movement possibilities for choreographers and dancers to consider unexplored avenues for creative work and improvisation (Collins, 2011).

Project Background

Choreography lends itself well to the realm of live coding due to the nature of dance, and how just within a single performance can alter itself from the so-called score. Unlike music or language, up to the present there is no widely accepted "language" for transcribing dance. It cannot be replicated without being experienced, to see or feel a body moving through time (Birringer, 2013). This inherent "liveness" and mistranslation of the original is magnified with the use of live code simply as a medium to invoke reactions from dancers without thought, and the programmer's own lack of expectation of the final outcome of their code.

Within dance, one way of communicating movement is the use of scores. Scores may be compared to musical scores in that they may notate a work but more so in dance they are used as a starting point for movement instructions that are to be performed by a dancer or dancers. Choreographer Jonathan Burrows (2010) discusses two approaches to dance scores, one being akin to a classical music score where there are clear instructions for a piece, but the other being an inspiration for a performer. "...what is written or thought is a tool for information, image and inspiration, which acts as a source for what you will see, but whose shape may be very different from the final realization." Score are also common in live arts practices and have been part of various art movements such as Fluxus in the 1960s. Scores may be given to performers via speech, text, visuals or other means and may be created before, during or after a performance. Many dance improvisation artists work from scores, or sets of guidelines or instructions meant to be interpreted in order to create movement in a given time and space.

Because the movement and sequences in dance can be algorithmic in nature, communication between a programmer and dancer is not so difficult to accomplish. The body is instructed to move between basic positions, one after another, forwards or backwards or repeated in loops, or maybe splitting their interpretations like an if/else statement. These analogies provide a natural transition to transcribe dance from paper and oral scripts to codes and bits.

Moving Patterns

Developing Terpsicode has taken several steps, during which language, imagery, and choreography have been developed. This process was also inspired by the previous performance work Moving Patterns.

Moving Patterns was a live coded dance performance, which utilized DanceDirt[1], a custom library written by Tom Murphy for TidalCycles[2]. In DanceDirt, the programmer was able to use the patterning capabilities of TidalCycles to create visual patterns of images, which were viewed in VLC player. Within the performance, the images acted as a score for dance improvisation. The piece created a feedback loop between the performer, who improvised movement based on the projected images, and the live coder, who selected the images and composed them in real-time sequences in reaction to the performer's movements. The collaboration of these two elements resulted in a codependent performance in which the coder and the performer sought direction from each others' decisions.

However, Moving Patterns made it apparent that TidalCycles did not have the capabilities to live code choreography in a way that felt complete for a dance composition. Much of its grammar and functions were based upon principles found in sound, or that result in specific sonic outputs such as reverb or reverse. The desire to develop a language more adapted to the vocabulary used when working in a dance studio arose during this process. Thus, the incentive to design a choreographic programming language was realized.

Sampling Movement with Photos

One of the questions that live coding visual choreographic scores raises is what medium is best used to convey instructions or inspiration to the performer. Still images were decided to be an appropriate medium due to the atomic nature of photo in capturing movement and their ability to be sequenced arbitrarily, like shuffling frames in a video.

Such flexibility allows for the spontaneity that defines live coding. Although images have become a starting point for Terpsicode, other media such as video or motion capture data may be more appropriate at a later point in this process of language creation.

The images used to create the visual score are from a large library of photographs taken through a time lapse camera. The camera took two photos per second of a dancer improvising movement. This resulted in a range of imagery that could be used for sampling movement.

*

Similar to how a sound sample is a select moment from a longer audio track, the still image is a single moment from a larger set of movements. While this could be seen as problematic in the documentation of dance, as it is missing key elements of movement such as spatial paths and dynamics, it provides interesting source material for dance improvisation and choreography. Choreographers will have access to frames of movement that they can piece together in unconventional ways, and dancers must instantaneously interpret the connections between those images. These photo libraries reference the initial Moving Patterns performance, where the use of images as samples was first implemented as a framework for improvisation.

Tagging Movement

After the images were captured, the clustering algorithm t-SNE was used to assemble like images together into one document, resulting in a spread of groupings based on visual similarities. The images were then labeled according to the dancer's inferred actions. t-SNE lacked the ability to understand the body as a three dimensional object, leading to discrepancies in how the images were to be labeled. For example, if the dancer was doing the same movement but facing different directions, the algorithm recognized the directional shift as separate from the original. This resulted in similar images existing in different groupings throughout the document. These unnecessary separations made labeling the images more difficult. Additionally, the process of tagging still images of motion unearthed the apparent gap in kinesthetic understanding of movement versus the ability to verbalize the action, or rather, one frame in an assumed action.

Figure 2: Movement terms for development of the grammar and syntax of the programming language.

Manual labeling of frames was done after the initial clusters were formed, adding a human choreographic approach to the tagging of images. This process initially started as a simple way of finding groups

Figure 1: Example of images used within the development of Terpsicode as sorted by the t-SNE algorithm. Dancer: Marissa Forbes.

of images that would have the same name, but it also came to demonstrate how the machine learning algorithm is of service to an artistic process, rather than the art being produced strictly by a computer.

Developing the language

Terpsicode is under development using PEG.js[3] to create a parser and javascript to create a mini language. In order to populate the lexical grammar, appropriate terminology had to be selected from the discipline of dance and choreography.

To begin this process, choreographic vocabulary describing movement, timing, and phrasing was collected and collated into categories. Some of the positional terms were used in tagging the images, while choreographic terms around compositional phrasing and timing structures became the key words for function names in the programming language.

Pseudo code was utilised as a means to determine if the language had the ability to convey choreographic information, from the programmer to the computer, in a way that still worked with the domain specific ways that choreographers communicate with performers. For example, terms such as “retrograde” or “coin flip” expresses language that choreographers understand , but must be broken down in order for computers to process. Writing the pseudo code was useful in determining how pattern structures, timing and rhythms, and movement combinations could be created, as well as what text fragments the computer must be taught.

Writing a parser to pattern the images in a browser window was the next step in the development of the mini language. Taking the pseudo code and applying this to displaying the tagged images was done in javascript. The tagged images can also be patterned in different orders and the length of time they are displayed are also determined in javascript. Figure 3 gives an example of how the code appears in the console log and the image appears in the browser.

Further development of the language includes expanding the image library with more figures, expanding those images outside of the human form, and building Terpsicode into a working plugin for existing compilers.

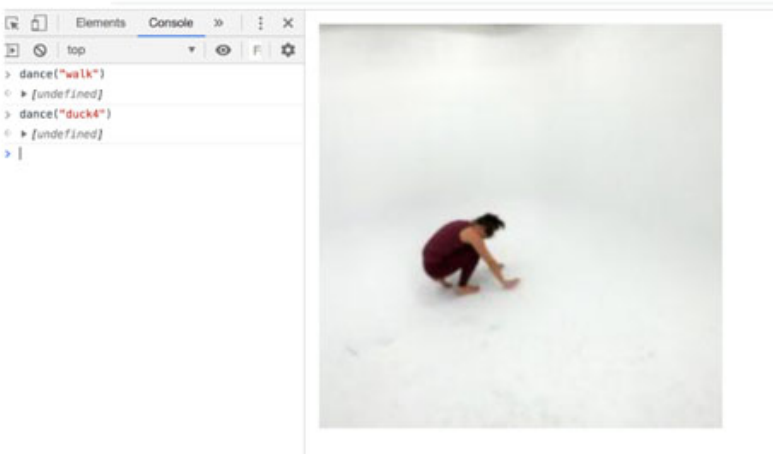

Figure 3: Terpsicode in the browser calling the tagged images.

Terpsicode in Performance

Though the language is still under development, the first performance with Terpsicode was May 17, 2019 at the Festival of Algorithmic and Mechanical Movement in Sheffield, UK, danced by Tara Baker and live coded by Kate Sicchio. Here, a twenty-minute dance was improvised using a limited lexicon from the mini language, which included the terms “fall”, “duck”, “walk”, “stomachspiral” and “flick”. Each word displayed the first tagged image in the directory for that term for a default time of three seconds. As the words were added via the live coding language, images were added into a cycle of pictures. If a number followed the term, that image would appear for the number specified amount of time. The order, pattern and timing of the images were determined by the live coding.

Even with a limited amount of images, language and no real com-

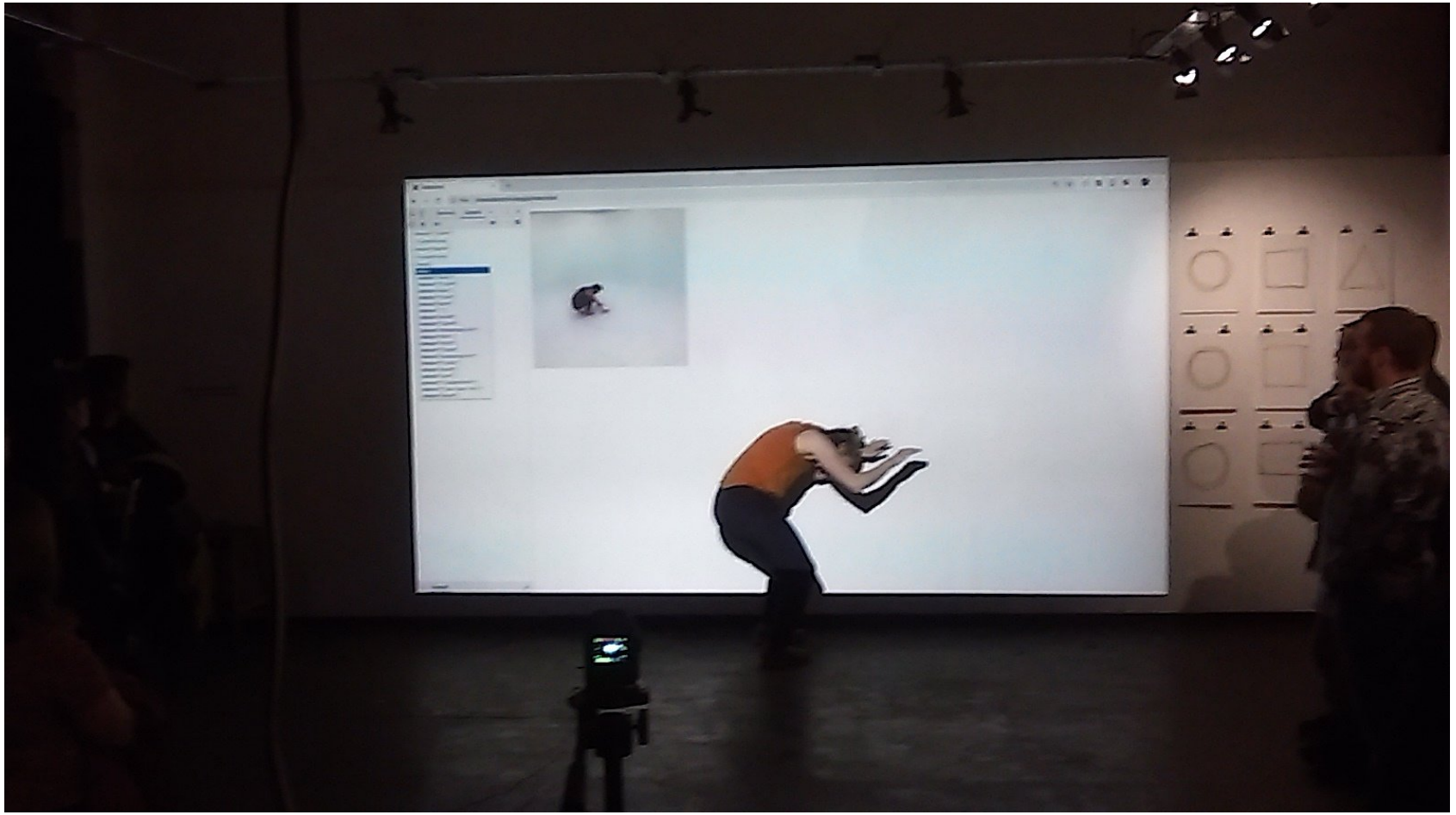

Figure 4: Terpsicode in performance at AlgoMech Festival, May 2019. Dancer: Tara Baker

positional terms available for the performance, a piece was beginning to unfold in this improvisation. The dancer was given the instructions to begin the piece by copying the images and transitioning from shape to shape. The order and pattern of the images should be demonstrated clearly for the first minutes of the work. However as the piece progressed, the dancer was told to respond to the images, giving her more freedom to interpret the shapes or use the images as a starting point for further exploration in her movement.

This approach of strictly following the score versus opening it up for interpretation also affects the live coding of the images. The opening must consider the speed of changes and the complexity of patterns. If the live coder speeds up quickly they must be aware that it may be challenging for the dancer to follow. This may mean certain shapes are not created or an impossible score is created. This in itself is not an issue as this can lead to create problem solving on behalf of the dancer, but it should be made aware as a choice of the live coder. After gaining familiarity with the process, the score becomes more of an inspiration for the dancer, the live coder is less in control of the output and therefore playing more with possibilities for the dancer to explore. These ways of interpreting and working with the visual score resonate with the definitions above from Burrows (2010). Live coding becomes an exploration of the possible but not necessarily what is actually performed because the output is meant to be danced by a human, who has agency within this work to make the final decision on what movement is created.

One issue arising from this work was the presence of the code onscreen next to the images. While this idea was to highlight the nature of the score being produced live and provide an example of the TOPLAP Draft Manifesto, it was questioned as part of this work. The dancer was unclear which to follow at times, the words or the images. Also having the images next to the words may start to imply certain movement must be performed when the shape itself is just an artefact of that movement. This starts to undo some of the reasoning in using still images as described above. The dancer may have less autonomy in the pathways of the shapes when text is also presented as part of the score. More experiments with the text will be explored in future iterations of this piece.

Summary

The development of a mini language for patterning images to create a choreographic score is an ongoing exploration in live coding, sampling movement, and the creation of dance improvisation performance with technology. There is no predominant choreographic dictionary so key words were only derived from professional practice. The process of designing Terpsicode brings to question the naming and vocabulary used for dance in the context of reformatting to programming syntax and how choreographers may further interrogate the utilization of live coding within their creative work. Samples of dance movements were recorded and processed with machine learning algorithms in order to break down some of that vocabulary. Implementation of the resulting language designed for coding was built on javascript. Further development will be made into making Terpsicode more usable. A first iteration was used in performance to create an image based score for dance improvisation, demonstrating how this language can be developed further for this purpose. Future performances will start to explore more compositional elements in the language as well as more ways of presenting the final score within the performance.

References

Birringer, J (2013) What score? Pre-choreography and post-choreography in International Journal of Performance Arts and Digital Media 9(1):7-13.

Burrows, J (2010) A Choreographer's Handbook. Routledge: London.

Collins, N (2011) Live Coding of Consequence in Leonardo 44(3):207–211.

deLahunta, S (2010) The Choreographic Resource: Technologies for Understanding Dance in Contact Quarterly 35(2):18–27.

Gourfink, M (2003) L'Innomee. Available from: www.myriam-gourfink.com/lInnommee.html [accessed Sept 10, 2019].

Sicchio, K (2014) Hacking choreography: Dance and live coding in Computer Music Journal 38 (1), 31-39.

Sicchio, K (2019) Programming paradigms for the human interpreter, International Conference on Live Coding. Available from: http://iclc.livecodenetwork.org/2019/ingles.html [accessed Sept 10, 2019].