# Sketching by Programming in the Choreographic Language Agent

Luke Church
*Computer Laboratory*
*University of Cambridge*
*luke@church.name*

Nick Rothwell
*Cassiel*
nick@cassiel.com

Marc Downie
*OpenEndedGroup*
marc@openendedgroup.com

Scott DeLahunta
*Random Dance R-Research*
*scott@randomdance.org*

Alan F. Blackwell
*Computer Laboratory*
*University of Cambridge*
*Alan.Blackwell@cl.cam.ac.uk*

Keywords: POP-I.C. artistic process, POP-III.C. visual languages

## Abstract

We describe the Choreographic Language Agent, a programming environment designed for use by dancers and choreographers in the context of improvisatory composition methods. CLA provides a means for dancers to interact with a simple but powerful set of 3D geometric transforms, creating a wide variety of kinematic and dynamic configurations expressed in the form of phrases. These phrases can be composed in a dynamic visual arrangement, offering sophisticated facilities for provisionality and version control. Direct interaction with a live 3D rendering is a key feature of the system, although this rendering is only an intermediate product, designed to offer ample space for alternative interpretations when mapped onto dance movement. The result is a programming language that emphasises transience, ambiguity and creative flow rather than the conventional requirements of professional software engineering contexts.

## 1. Introduction

The Choreographic Language Agent (CLA) is a project directed by Scott DeLahunta within the R-Research arm of the London-based contemporary dance company Wayne McGregor|Random Dance. Working in collaboration with choreographer Wayne McGregor, CLA builds on more than a decade of research investigating the cognitive processes of making dance (e.g. McCarthy et al 2006). The early period of this research programme explored the sketch notations used by Wayne McGregor and members of the company as a component of the creative process. As in many other design disciplines, the formal notations associated with documenting dance (e.g. Labanotation, Benesh) are of limited value during exploratory design activity. A Cognitive Dimensions analysis (DeLahunta et al 2004) identified some of the notational strategies adopted by McGregor as improvised or habitual responses to the limitations of those more familiar conventions. During the period of this research, McGregor has been recognised in numerous ways as one of the world's leading choreographers (for example, he was appointed principal choreographer of the Royal Ballet, in addition to contributions to popular culture such as choreographing Harry Potter films and delivering a TED talk on the science of choreography), so the context of this research can reasonably be considered as representing respected leading approaches to contemporary dance.

The objective of the CLA project was to develop a computer support system that could contribute to this creative process. The term "language" refers to the need for novel notation, as well as the fact that each of McGregor's works is conceived as developing or extending new languages of dance, created through processes that include both studio improvisation and analytic perspectives involving multi-disciplinary collaborations.

## 2. Requirements

Extensive preparatory work with a wide range of academic collaborators, including a two day workshop with 12 scientists and 40 participants at a public session, made it clear that the scope of this project corresponded neither to any existing programming "language", or any existing choreographic notation. For those coming to the project from outside the world of professional dance (this includes some of the present authors), it is worth documenting some of the features that we find to be contrary to expectations among technical colleagues, although not necessarily surprising to those who have already worked in contemporary dance.

First, it was not considered necessary (or even desirable) for the display of the CLA to have any resemblance to the human body. On the contrary, dancers and choreographers find it both natural and convenient to represent configurations of human bodies using their own bodies. The CLA representation was intended to be open for interpretation either as movements of an individual dancer, or of a group, or a more abstract architectural form, not a simple maquette to be copied by a dancer.

Second, it was not required that the CLA should represent movements that fit within the motion constraints of the human body. On the contrary, Wayne McGregor's choreography is renowned for the way in which it extends the range of conventional body motion. To be a useful tool in McGregor's choreographic process (and that of many other contemporary choreographers), the CLA should present the dancers with a problem that must be solved through searching for new movements, rather than a solution in the form of a movement to be incorporated into the creative work.

Third, the CLA was not intended to create visual content for digital projection in a stage production. Although several of the authors have significant experience of programming visual arts commissions, and Downie has provided live computer graphics during Random Dance performances, the CLA is purely a creative tool, for use in the studio as one component of an improvisational process.

## 3. Strategy

Wayne McGregor often creates new work through a research process that requires dancers to conduct various conceptual or intellectual preparations, either in parallel with improvisatory exploration, or in advance of studio improvisation. During the period that CLA was being developed, several other scientific projects were also being conducted in association with the creative process – these included instrumentation of the dance studio with motion recording cameras, and introducing dancers to new conceptualisations of their dance process through the use of Barnard's Interacting Cognitive Subsystems framework (Barnard et al 2000).

CLA played a role that bridged the intellectual and embodied improvisation aspects of this process, by providing dancers with new abstract conceptions and representations of their work. Some of these drew rather literally on existing programming concepts – for example, the CLA display would include a verbal "language" in the form of geometric terms that can be aggregated into "phrases". These aspects of the programming language are analogous to familiar aspects of dance – individual movements that can be aggregated into dance phrases (although explorations of the linguistic structure within such phrases in contemporary dance does not find any clear token boundaries or grammar – McCarthy et al 2006). However some aspects were less typical of conventional programming tools – the program is a starting point from which to explore behaviour, rather than a precise specification of a required result. It is in this respect that it should be considered a sketching tool, rather than a software engineering tool.

Nevertheless, there is one respect in which CLA does draw on aspects of software tool design, which is to introduce some practices of software engineering into the studio context. We particularly emphasised models of version control, retaining intermediate work products, reviewing version history, and creating new branches in order to explore alternatives within an iterative and exploratory process.

## 4. Implementation platform

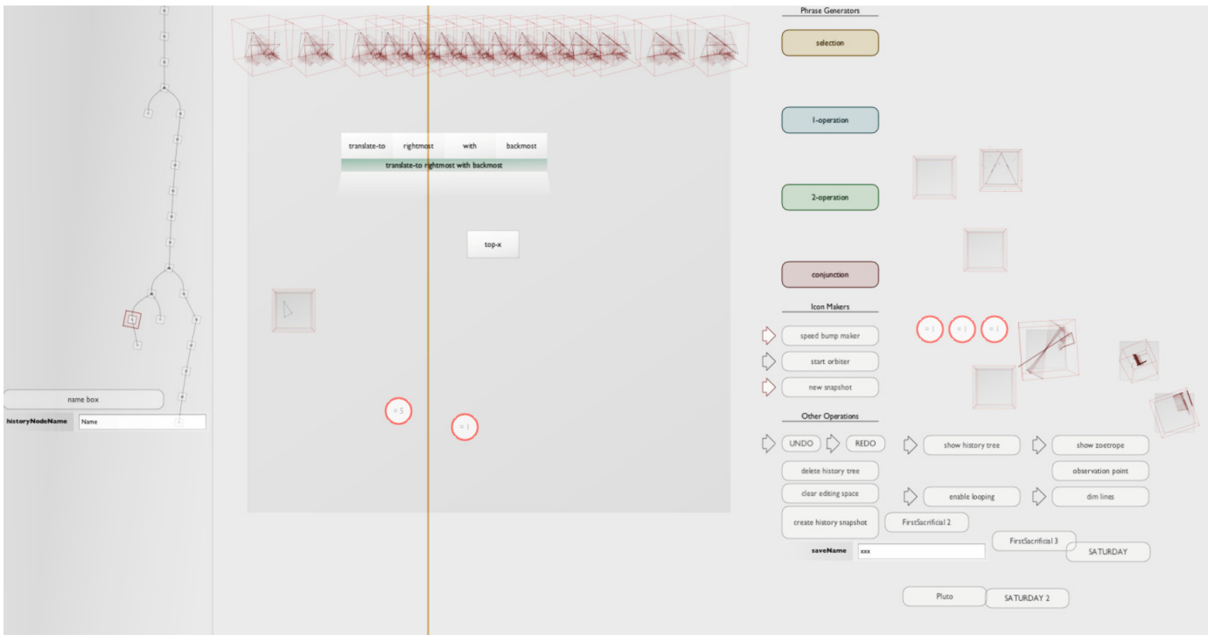

*Figure 1 – the right-hand screen of CLA, as implemented using the Field drawing surface (individual features are discussed later). The execution cursor is represented by a vertical red line.*

CLA was completely implemented within the Field environment from OpenEndedGroup (Downie 2008). Field is primarily a development environment for making digital art, but is sufficiently powerful that we were able to use it as the implementation platform for an interactive programming environment. In addition to relatively conventional text editing facilities, Field also includes a drawing surface that can be used either as a canvas, to arrange pieces of textual code, or to express control flow between other executable elements. In the case of CLA, control flow is represented as a cursor that sweeps from left to right across this drawing surface, triggering language phrases as it passes across them (Fig 1).

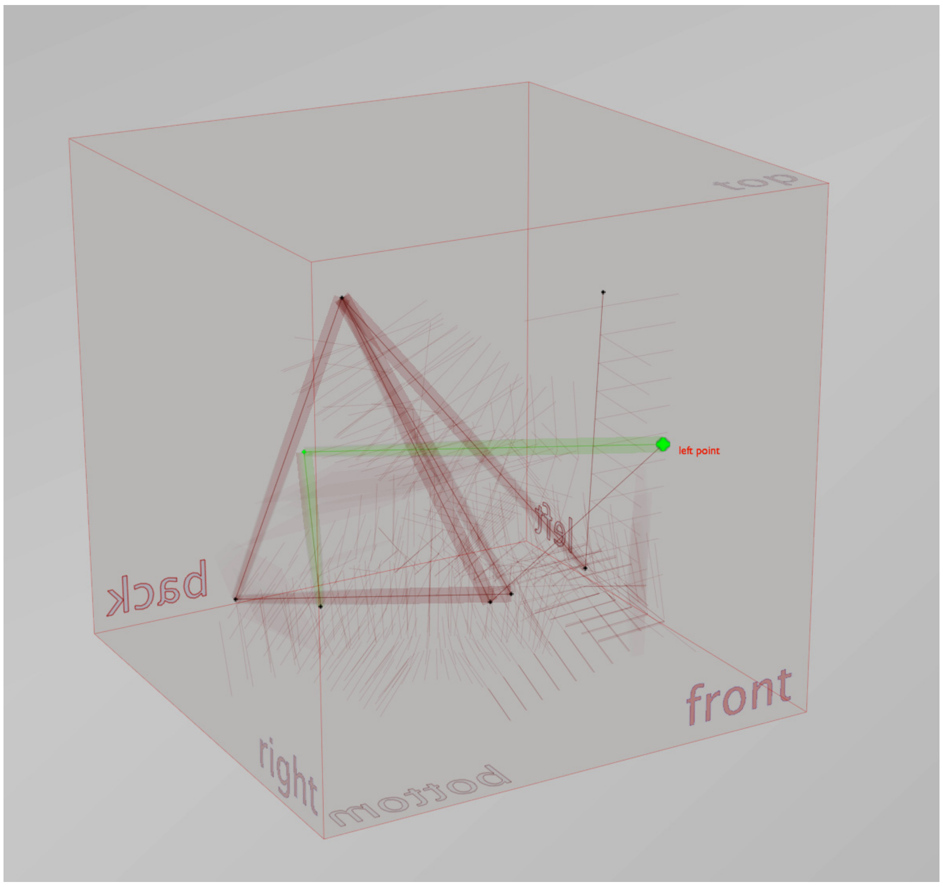

*Figure 2- the left-hand screen of the CLA workstation, showing an abstract 3D figure animated within a virtual stage. The stage is represented by the surrounding box, with labels indicating which sides are the front/back, left/right of the stage. The thick lines have been created by the user to represent the skeleton of an articulated figure. The fine lines are added by the system as the figure moves, leaving a trace of that movement in the 3D space.*

In the CLA deployments, Field is augmented with a second screen, which permanently displays a rendering of a 3D space – a "stage" within which abstract figures can be rendered and controlled by the language elements (Fig 2). The rendering of walls and floor around this space was an essential cue, both to navigation, and to understanding of the geometric figure as physically situated rather than simply an abstract "form in space". The view of this rendering is controlled by a 3D mouse (3D Connexion Space Navigator), allowing users to explore the figures by interactively rotating the viewspace during program creation or execution. The rendering of the geometric configuration in this space is intentionally minimal – points are simple spheres, connected by armatures. Both the points and connecting links are blurred, to emphasise their intention as material for interpretation, rather than precise specification of dance movement.

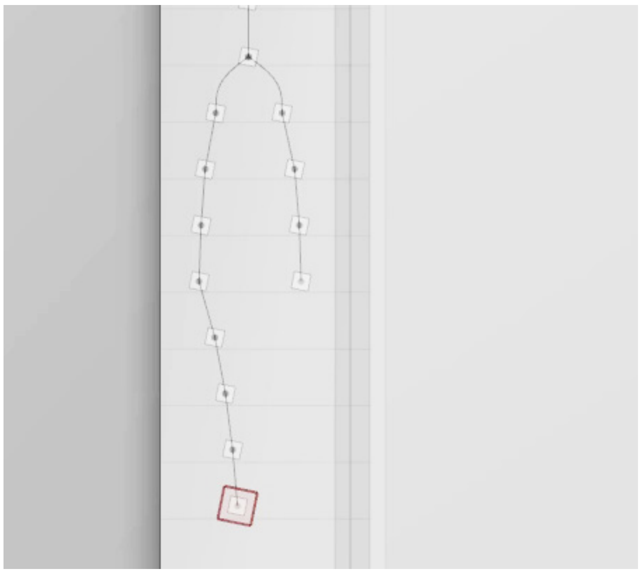

*Figure 3 – detail of the CLA window showing the history tree*

Finally, CLA relies on the Mercurial version control system to manage the capture and branching of exploratory changes. Every action within the Field drawing surface results in a change being logged to Mercurial, and users can return to any previous edit state by clicking on a node within a graphical tree at the side of the drawing surface (fig 3). Further changes then result in an automatic branch, starting from that node, and growing alongside the previous history. This provides considerably more power than conventional stack-based undo and redo (even when the whole stack is available, as in the Photoshop History palette).

## 5. Features of the CLA editor

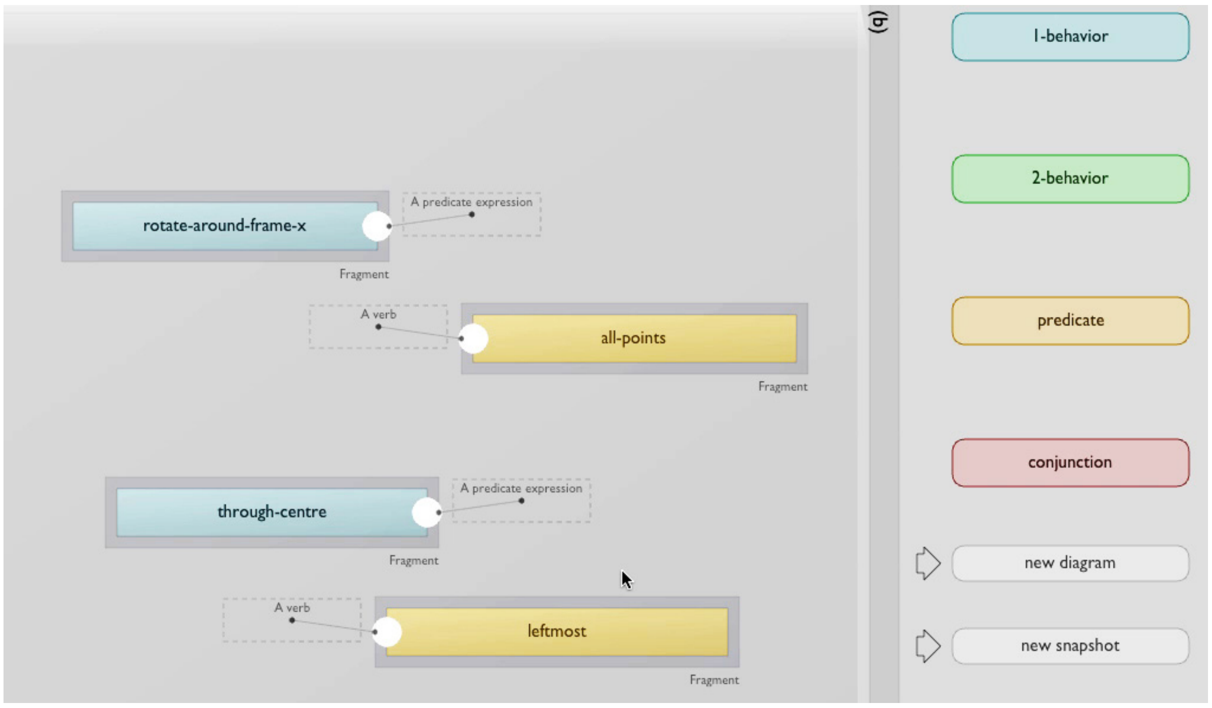

*Figure 4 – the structure of the phrase language in CLA*

The fundamental elements of the phrase language have three types, distinguished by colour-coding in the Field drawing surface (Fig 4). The first of these specifies a transformation to be applied to these points (e.g. "rotate-about-centre"). The second specifies one or more points, usually by reference to the 3D space (e.g. "leftmost"). The third specifies conjunctions of other phrases – either to be executed concurrently, or with one immediately following the other. Rather than the jigsaw-like visual cues that have become routine in systems such as Scratch, unbound connection sites are indicated more organically, as whiskers extending to the side of incomplete phrase elements offering hints about what might be placed there.

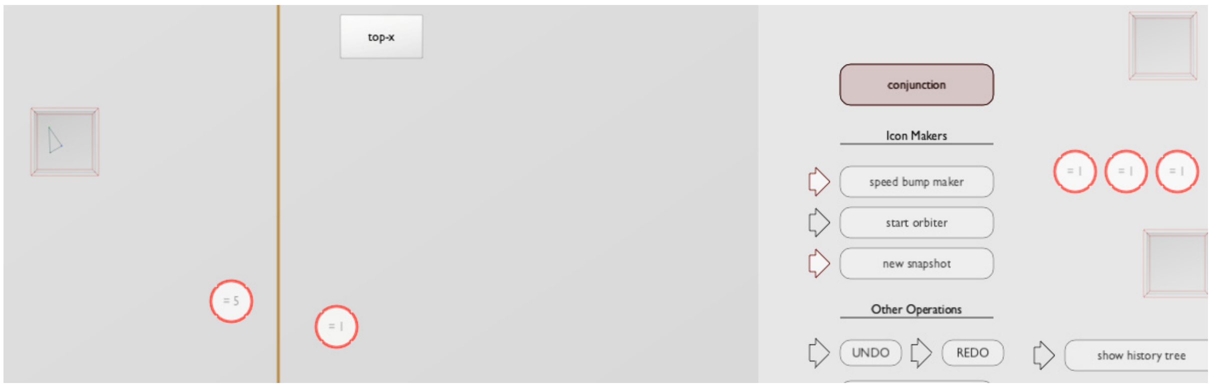

*Figure 5 – "speed bumps" to change the relative animation speed*

Any number of these phrases can be distributed around the drawing surface. On pressing play, the cursor moves across the drawing surface from left to right, animating the 3D transformations as it crosses them. The relative duration of a completed phrase corresponds to its width on the screen – it is possible to stretch or squash it by dragging the sides. The resulting combination of sequential and concurrent phrases of different durations can produce highly complex geometric effects. In response to initial trials, a feature was also added to control the global time-base – the speed with which the Field cursor moves. These "speed bumps" can be used to modify the dynamics of a composed piece independently of the relation between its elements (fig 5).

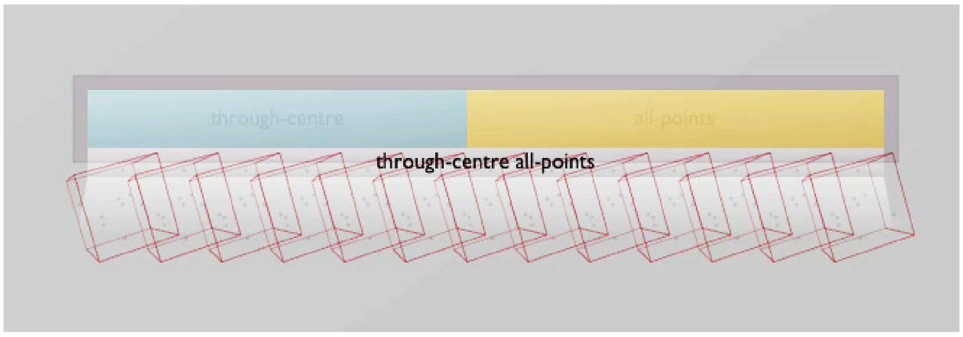

*Figure 6 – Zoetrope, visualising the effect of a single phrase*

As each phrase is executed, a sequence of snapshots is compiled, visualising at a glance the overall shape of the point trajectories. The whole series of snapshots is displayed above the phrases that generated them, in a form that we describe as a "zoetrope" (fig 6). Individual snapshots of the 3D configuration can also be dragged to the side of the screen, saved for reference, and used to initialise or combine system states. The zoetrope visualisation, although useful as a visual reference to a dynamic process, also has great visual appeal, leading us to provide zoetrope-like structures within the stage itself (fig 2).

As will be apparent from the figures in this paper, the visual style of CLA, as with the visual style of Field, is unlike that of most programming languages. Rather than bright colours, high contrast, and crisp edges, these systems intentionally avoid association with technical or educational diagrams. The intention of a diagram, like most computer programs, is to be unambiguous. But the CLA output must be open to interpretation. Although a given version of a CLA program has deterministic output (important, because it must always be possible for users to return to the state that produced a particular effect of interest), the visual and verbal rendering of the phrase should not constrain the potential reading of its output - and the two are necessarily linked, given that the creator and reader, in this end-user programming context, are often the same person.

## 6. Observations of use

The CLA has now undergone several iterations, in response to trials by Wayne McGregor and the members of Random Dance. Initial demonstrations made it clear that users would need extended tutorial support – this was initially addressed in the form of a live video produced by Nick Rothwell, walking through the system capabilities. Given the increasing prevalence of video walkthroughs as a tutorial medium for explaining system features of user-customisable platforms (examples range from YouTube videos of iPhone settings to live feeds from hack days), this deserves further study as a key element of end-user programming. In our first meeting after Wayne McGregor's initial period of experimentation with CLA, Wayne commented that he had watched the tutorial video so many times, it felt like Nick's voice had become lodged inside his own head!

As with many deployments of programming tools in highly novel contexts, some findings are relatively mundane from a psychology of programming perspective. Just as with a previous observation that live coding languages must still be usable when the programmer is inebriated (Blackwell & Collins 2005), we were relatively surprised to learn that a busy international choreographer does much of his "office work" in airport lounges or using WiFi in a Starbucks. The practical constraints of these contexts made the twin-screen setup and special 3D controller rather an obstacle in the choreographic process, by comparison to a compact laptop. It is almost certainly the case that a great deal of end-user programming research implicitly assumes a user who is sitting at a desk (or at least in a classroom), whereas field observation would probably identify a surprisingly large range of physical contexts and postures adopted by end-user programmers.

The use of a non-standard interface device (the 3D mouse) alongside a regular mouse/trackpad introduced consistency problems of button assignment and command shortcuts between the devices. These are relatively familiar everyday annoyances for programmers, but more disconcerting for professional dancers who might be best described as "social" computer users in their everyday lives. In retrospect, we might also have been more aware from the outset of the embodied interpretations – more salient to dancers – of gestures such as using the middle finger to select points, or the fact that the 3D mouse and regular mouse both assigned the same function to "left" buttons, despite the fact that a dancer (and perhaps any normal person) mirrors actions from right to left. The native Field environment also incorporated some of the features of MIT-style command chords (use of shift/ctrl/meta modifiers) that were inappropriate to an end-user audience, however convenient in the context of professional exploratory programming.

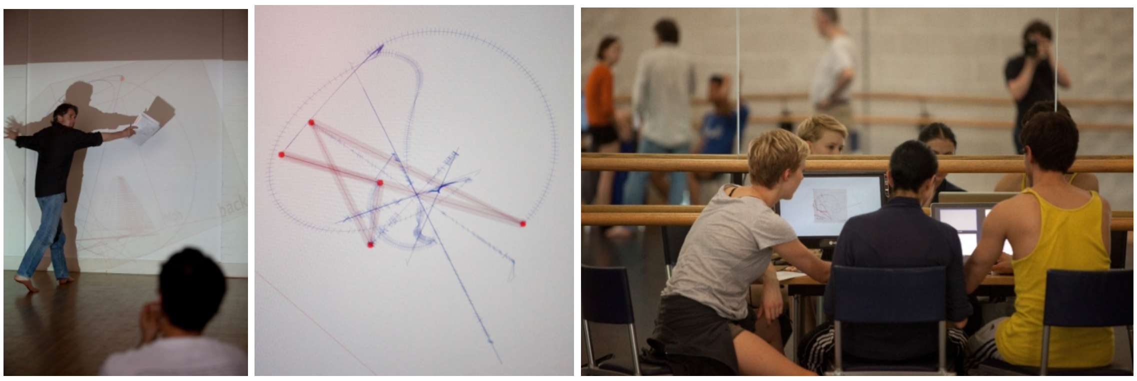

*Fig 7 – 3 day workshop*

The main period of research observation took place over a three day studio workshop, attended by all members of the company, together with the authors as instructors/facilitators, also providing technical support and observation (Fig 7). Some changes were made to the CLA operation at the end of each day – in particular, refinement of the model used to define the magnitude of the "speed-bumps", changing them from being relative to the previous speed (which involved fractional mental arithmetic to explain the results of successive speed bumps), to being expressed relative to an absolute reference value.

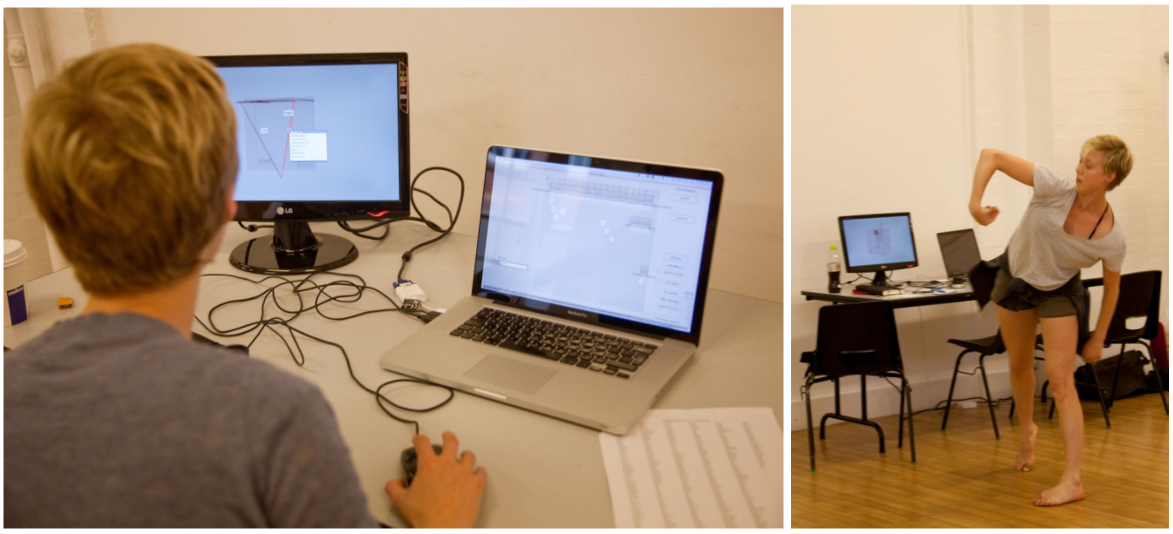

*Fig 8 – a) Experimental sketching of a new movement sequence, followed by b) dance exploration of the geometry created.*

The structure of the CLA sessions within the workshop was based around relatively long periods of experimentation with the CLA (from 90 to 120 minutes) (fig 8a), followed by extended periods (more than 60 minutes) in which the dancers worked to resolve the "problems" that they had set for themselves in the CLA animations, turning them into dance (fig 8b). The dancers worked in groups of 2 or 3 when interacting with the CLA (a constraint partly imposed by the number of workstations we were able to install in the studio, although group working was also seen as a valuable part of the creative process). Each dancer then developed a different interpretation of the kinematic and dynamic characteristics of the animation they had created. These interpretations could be as diverse as mapping joint configurations onto particular parts of the body (an arm, foot or head), mapping point motions onto the space of the room, or interpreting the dynamics of a swept line through jumping or rolling. Where groups of dancers worked together, this mapping-as-problem-solving became especially intense, as the dancers negotiated among themselves how the mapping might be achieved, and which aspects would be assigned for interpretation by which dancer.

On the second and third days of the workshops, those dancers who had become most confident in CLA operation had spent time overnight, planning new CLA programs that they would create the following day. The group activity then made a transition to these individuals as leader-operators, with other dancers observing and making suggestions to refine the animations. Given the status of the CLA animation as a creative sketch, in which the final "product" was the dance movements created, there was little desire to return to refinement of the CLA program after the transition had been made to working out the dance problem. In fact, even where there was a "bug" (as a programmer might call it) in the animation, dancers explicitly told us that they avoided the temptation to "fix" it, because the unintended behaviour was more valuable to them as an artistic challenge.

Observation of these sessions resulted in many small discoveries, refinements, and understanding of design decisions that we wished we had taken differently. The following discussion of these observations employs analytic concepts from the Cognitive Dimensions of Notations (Green & Petre 1996, Blackwell and Green 2003), with dimension names italicised in accordance with standard practice, as well as concepts from the Attention Investment model of abstraction use (Blackwell 2002) and Engelhardt's Language of Graphics (2002).

## 6.1. Version History

As with most version-controlled software development, our version history applied to the "code" of the transformation phrases, but not the "data" of the points arranged on the stage. Execution snapshots preserved data, but only if the code was executed. This distinction was of course invisible to our users, for whom a carefully constructed arrangement of points was just as valuable as the code that manipulated them – and thus a source of significant disappointment when it was accidentally deleted. Clearly, both code and data should be managed within a version history.

The version history itself was not as useful as we would have liked, because of the uniformity of the rendered nodes, making it difficult to return to interesting points. We added a facility to assign labels to nodes, but this requires the user to recognise in advance which versions are going to be the most interesting. This form of *premature commitment* is unrealistic in the context of an artistic process, especially considering the abstraction investment involved in stopping to think of a name just as the creative process is at its most exciting. Ultimately, we had a combination of *role-expressiveness* and *hidden dependencies*. Both the history tree, and "snapshots" of the 3D rendering, have related functions – to allow explicit reference to states within the creative process. Despite the fact that we were aware of this requirement, and designed these features to address it, we did not succeed in integrating them into a usable tool.

## 6.2. Sub-devices

As would be expected from Cognitive Dimensions analysis, we identified liberal use of sub-devices to enable offline processing or reduce *hard mental operations*. Some dancers copied pieces of geometry from the screen onto paper. Others composed explicit mappings from the screen onto the body of a dancer. *Juxtaposability* was a particular challenge – it was almost impossible for dancers to watch the screen while dancing. Some suggested that future systems might make the animated motion continuously visible via headset displays, or at least with wall-sized projections around the studio. In the absence of such equipment, substantial amounts of time during the dancing periods were spent walking over to the workstation, then watching the screen intently, attending to the specific aspects of the animation that constituted the current problem focus, and memorising them before returning to the dance floor. This hard mental operation was mitigated by the rendering of a blurry swept "sheaf" of lines that overlaid multiple frames over a second or so, thus allowing direction of movement to be seen at a glance (fig 2).

## 6.3. Visual feedback

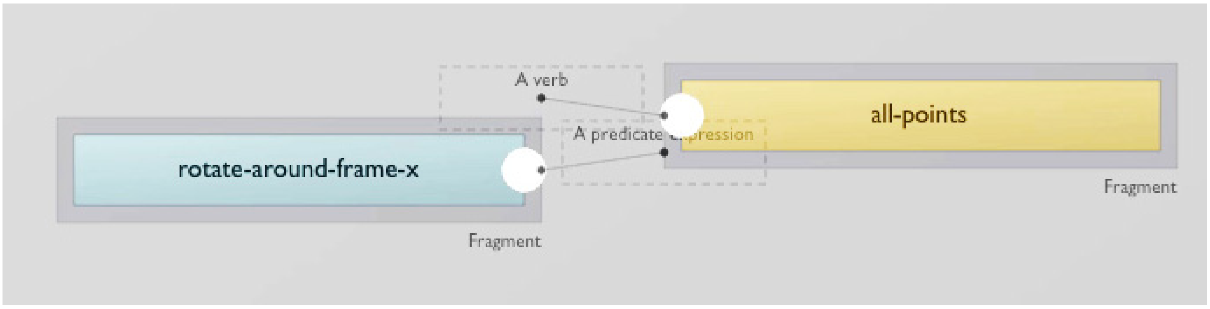

*Figure 9 – visually unobtrusive syntactic cues in CLA*

Although the visual style of CLA, like that of Field, is distinctively oriented toward the sensibilities of professional artists rather than engineers, this sometimes introduced unfortunate consequences for usability. The rendering of the phrase elements included a number of visual cues, generally designed to be unobtrusive and non-directive (fig 9). We were concerned that it should be possible to execute the program regardless of the degree of syntactic completeness (allowing *provisionality* and *progressive evaluation*), meaning that incomplete phrases were often present in the display area. There was a visual cue to indicate that a phrase was complete – the words underneath a complete phrase were highlighted with a continuous bar (fig 6), while incomplete phrase elements used highlighted open circles and whisker-hints to draw attention to the location of missing elements (figs

4 and 9). However, none of the dancers noticed this visual feedback until the second day, meaning that they were often unsure whether particular elements of the display would or would not have an effect on the execution behaviour.

## 6.4. Secondary notation versus verbal reference

Another unanticipated drawback of the visual notation (and one that is not directly reflected by a Cognitive Dimension) is that the collaborative nature of the studio work makes it necessary for dancers to discuss elements that they see on the display. In a textual programming language, it is always possible to simply read out a passage of text if it is necessary to refer to it in speech. With the visual notation of the CLA, we heard sentences such as "you need to hold that thing and go past that thing". To some extent, additional *secondary notation* would have helped with this problem of verbal reference – for example, changing the colour of one or more points so that they could be referred to as a group. However, any such visual detail carries the danger that users might be distracted by making something "pretty" – they observed that it was not always easy to predict, from the visual properties of the animation, whether it would be an effective starting point for dance. In particular, some animations that were visually beautiful offered little opportunity for further improvisation, while those that were complex and hard to read were "rich" as a source of potential movement cues.

## 6.5 Explicit representation of dynamics

Despite the fact that CLA is a domain-specific language, designed specifically to meet the needs of this group of dancers, there were some respects in which we seem not to have achieved the *closeness of mapping* that might be expected in a DSL. In reflecting on the three day workshop, many dancers spoke of the way in which they struggled to produce the particular animation effect that they wanted. They described this in terms such as "bouncing", "momentum", "energy" and "vroosh". In retrospect, these are all ways in which we might expect dancers to describe salient aspects of abstract motion (although they derive in part from a choreographic task used in the workshop, which involved interpretation of expressive motion terms). To some extent, the "problem-solving" aspect of the improvisation process depends on removing any obvious mappings – as in the avoidance of direct body representations. Nevertheless, one dancer who had found CLA least engaging said that it took at least 10 minutes to create anything he could relate to the body, and 40 minutes to make something interesting. It is possible that a language paying more attention to dynamics, as well as kinematics, might provide more expressive power in this respect.

## 6.6 Lack of syntactic constraint

Our focus in this project on creative arts practices offers a relatively extreme example of exploratory design activity. This has allowed us to observe some Cognitive Dimensions trade-offs that may not have occurred in a more conventional software engineering context. For example, syntactic constraints on drawing area manipulation were made to be absolutely minimal – the syntactic elements can be dragged anywhere at any time. In principle, this provided a high degree of *secondary notation* (with respect to location in the visual plane), and minimum *premature commitment*. However this extreme design choice unexpectedly resulted in high *viscosity* – because any element of a phrase could be dragged elsewhere at any time, it was then necessary to move the other components of the phrase to reassemble it. As with classically viscous visual dataflow languages, we watched CLA users spending a lot of time reassembling phrases in order to move them to different positions on the screen. However, whereas visual dataflow languages like LabVIEW present a tradeoff between *hidden dependencies* and viscosity in the routing of the dataflow connection paths, the CLA exhibited similarly high viscosity through its lack of syntactic boundary constraints. A more reasonable approach to visible syntax maintenance can be seen in the Scratch language – although it is possible to drag syntax tiles out of the expression they belong to (and sometimes to do this accidentally), most dragging operations affect the whole of a syntactically bounded expression. Many such languages seem to rely on an implicit virtual "syntactic physics" that simulate gravity, magnetism or adhesion to provide the operator with subtle interactive cues – ToonTalk provides another case study, in which manipulation can be *error-prone* due to lack of physical resistance to significant syntactic changes.

## 7. Conclusion

The context of the programming tasks in this project offers an intriguing counter to the usual practices of software engineering, in which a sketch might be made on paper or whiteboard before translating it into a code "product". At that stage in a software engineering project, there is certainly little desire among software engineers to return to preliminary conceptual sketches and revise them after the product has started to take shape. In CLA, we have created a programming environment whose purpose is conceptual sketching, and whose output is a transient representation to be discarded, rather than a final product.

In some ways, this might be compared to student programming exercises, which are also transient, and not the "final product" of the programming course – that product is a qualified programmer rather than a program. However, it is also useful to consider this transient form of programming within the context of live coding, where the program is improvised in front of an audience (Blackwell and Collins 2005).

Experimental languages of this kind are valuable in two respects. Firstly, they offer an opportunity for programming to become more inclusive, through creating languages to support users with different skills and work practices. Secondly, they help us to explore boundary cases that test general principles of programming language design outside of typical design parameters. The CLA project has, of course, also made contributions to choreographic practice and arts research (e.g. Blades 2012), and dance material developed in the workshops described here has become a component in new work for Wayne McGregor|Random Dance. However this research also extends beyond these immediate objectives, leading to new design-oriented programming languages, for contexts in both arts and business, that are now under development.

## 8. Acknowledgements

CLA was supported by Portland Green Cultural Projects. Luke Church's research was supported by Kodak. Nick Rothwell's work was supported in part by a Digital Futures in Dance residency at Dance Digital.

## 9. References

Barnard, P., May, J., Duke, D. & Duce, D. (2000). Systems, Interactions and Macrotheory. ACM Transactions on Human-Computer Interaction, 7, 222-262.

Blackwell, A.F. (2002). First steps in programming: A rationale for Attention Investment models. In *Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments*, pp. 2-10.

Blackwell, A. and Collins, N. (2005). The programming language as a musical instrument. In *Proceedings of PPIG 2005*, pp. 120-130.

Blackwell, A.F. and Green, T.R.G. (2003). Notational systems - the Cognitive Dimensions of Notations framework. In J.M. Carroll (Ed.) *HCI Models, Theories and Frameworks: Toward a multidisciplinary science*. San Francisco: Morgan Kaufmann, 103-134.

Blades, H. (2012). Creative computing and the re-configuration of dance ontology. In Proceedings of Electronic Visualisation in the Arts (EVA 2012), pp. 221-228.

DeLahunta, S., McGregor, W. and Blackwell, A.F. (2004). Transactables. *Performance Research* **9**(2), 67-72.

Downie, M. (2008). Field - a new environment for making digital art. *Computers in Entertainment* 6(4).

Engelhardt, Y. (2002). *The Language of Graphics. A framework for the analysis of syntax and meaning in maps, charts and diagrams* (PhD Thesis). University of Amsterdam

Green, T.R.G. & Petre, M. (1996). Usability analysis of visual programming environments: a 'cognitive dimensions' approach. *Journal of Visual Languages and Computing*, 7,131-174.

McCarthy, R., Blackwell, A.F., DeLahunta, S., Wing, A., Hollands, K., Barnard, P., Nimmo-Smith, I and Marcel, T. (2006). Bodies meet minds: Choreography and cognition. *Leonardo* 39(5), 475-478.