Redesigning Computationally Creative Systems For Continuous Creation

Michael Cook[1] and **Simon Colton**[1, 2]
[1]The MetaMakers Institute, Falmouth University, UK
[2]Computational Creativity Group, Goldsmiths, University of London, UK
www.metamakersinstitute.com ccg.doc.gold.ac.uk

Abstract

Most systems developed for Computational Creativity projects are run for short periods of time, which provides enough space to be creative, but limits the long-term growth and development of software both internally and in the wider world. In this paper, we describe the notion of *continuous creativity* by describing how ANGELINA, an automated game designer, was rebuilt to be an 'always-on' system. We review the history of ANGELINA and contrast this new approach with earlier versions. We introduce the term *presence* to describe the impact a computationally creative system has on its environment, and vice versa, and discuss how continuous creativity can contribute to a system's presence, providing greater creative independence, opportunities for framing, and space for the system to grow.

Introduction

Automated game design is a frontier challenge for Computational Creativity research. Composing music, writing stories, conceptualising visual aesthetics, inventing systems of rules – designing a videogame involves solving many distinct creative problems, and ensuring all of those solutions pull together towards the same ultimate goal. (Liapis, Yannakakis, and Togelius 2014) describes videogames as the 'killer app' for Computational Creativity, as they offer not only a variety of creative challenges, but also additional problems of co-operation and integration between creative tasks and potentially between different creative individuals.

ANGELINA is an automated game design project which has been developed, over several iterations, to explore ideas relating to Computational Creativity. The project's aim has been twofold: to solve the hard, technical problems of developing software capable of designing games automatically; and to investigate the social and cultural aspects of game design and try and understand how, if at all, an AI system can take on a role in this space. As a result, our work on ANGELINA encompasses studies of evolutionary computation and code synthesis, as well as user studies, exhibitions, and evaluating the cultural impact of the software. Each version of ANGELINA is designed to focus on a particular subproblem within automated game design, but shares a common core structure and engineering approach (Cook 2015).

Rebuilding and reassessing the system over the course of many years allowed us to refocus the project on new problems, as well as embrace emerging technology and new platforms. However, it also led to a lack of continuity between the different versions of ANGELINA, and had a negative impact on the perception of the system as a long-term creative entity. Observers found it hard to make sense of a system which changed so often, and ANGELINA had little persistence outside of a single creative act. Reflecting on these issues, we designed and built a new version of ANGELINA, with an emphasis on long-term creative existence. We called this new approach *continuous automated game design*, or CAGD, but more generally it expresses an approach to computational creativity we are calling *continuous creativity*.

In this paper, we discuss the process of redesigning ANGELINA in the continuous creativity paradigm. We begin by giving an overview of the history of the project and discuss its limitations, introducing the concept of *presence* to describe the aggregated long-term legacy of a computationally creative system. We then outline the new structure of ANGELINA, explaining the changes necessitated by a shift to continuous creativity. Finally, we place this new version of ANGELINA in the context of related automated game design research, and then discuss future work. The rest of the paper is organised as follows: in *Background* we cover the history of ANGELINA as a project; in *Presence In CC Systems* we critically reflect on the project and introduce the notion of *presence*; in *Designing For Presence* we describe the new ANGELINA, focusing on how the structure of the system has changed in response to the demands of continuous creativity; in *Discussion and Future Work* we discuss the potential problems and new directions exposed by continuous creativity; in *Related Work* we place ANGELINA in the context of automated game design research.

Background

ANGELINA is an automated game design system, and has been in development in some form since 2011, first presented at ICCC in 2013 (Cook, Colton, and Gow 2013). Although version numbering is somewhat obscured by project forks and anonymised systems, there are five distinct historical versions of ANGELINA, each with a different focus, but their broad structure remains the same across most versions. In this section we give a brief overview of the structure of the software, and then dissect some of its short-

comings. We also provide an overview of related work in terms of automated game design and computationally creative systems. Throughout this paper we occasionally refer to a specific version of ANGELINA with a subscript like this: ANGELINA$_1$. For a discussion of ANGELINA's various versions to date, see (Cook, Colton, and Gow 2017).

Each version of ANGELINA can be thought of as running through three distinct phases: predesign, design and postdesign, as described below.

Predesign

The system begins in a **predesign** phase. For early versions of the software this predesign phase was almost nonexistent, simply loading in parameters, but for later versions of the software this is where it would lay the foundation for the design problem it was about to tackle. ANGELINA$_3$ read online newspaper articles to decide on an inspiring article to make a game about, and would then search for online media like images and sound effects to use in its game design. In ANGELINA$_5$, the system was provided with a short phrase or single word before running, and broke the input down using online concept and word association databases.

Predesign was a flexible space which we could use to add in new functionality during the setup for the system. Because the design phase which follows it cannot be interrupted until it is finished, the predesign phase was the only way for the system to make creative decisions before work on the game began. Any expansion to ANGELINA's creative capacity typically had to fit into this phase somehow: for example, the system was later given the ability to scrape social media to assess how people felt about a particular notable figure, which then became a parameter that influenced other media searches. This easily fitted in to the predesign phase, because it ultimately only affected the media that would be fed into the design phase. However, we were unable to expand the system in more complex ways – responding to a serendipitous discovery while designing the game by going back and searching for more information online, for instance.

Design

The **design** phase is the largest and most important part of the system. All major versions of ANGELINA employ the same technique in this process, namely *cooperative coevolution* (Potter and Jong 1994). Unlike a normal evolutionary system, where a single population of solutions is evaluated and recombined until termination, a cooperative coevolutionary system is composed of several separate evolutionary subsystems solving their own part of a larger problem, in this case designing a game. When an evolutionary subsystem evaluates its population, instead of evaluating it in isolation it uses high-ranking exemplars from the other subsystems to synthesise a larger artefact, in this case a game, and then evaluates that larger artefact instead. Thus, the fitness of a population member is not just based on local evaluation, but on evaluation in the context of a larger solution.

Different versions of ANGELINA were built to break the game design process down into different kinds of subsystem – ANGELINA$_1$ had Level Design, Layout Design and Ruleset Design, for instance. Our intention was to replicate the way a small game developer might distribute the creative task of game development, where level designers, musicians, writers and so on would work independently, but share their work together to evaluate their progress towards a common goal. Another desirable feature this technique had was being omnidirectional. In many generative systems there is a clear line of steps that the software always moves through, in the same order. A cooperative coevolutionary system does not work in this way, because all parts of the artefact are being evolved simultaneously. If, for example, the Level Design subsystem struck on a particularly good design, this could influence the fitness landscape of the other subsystems.

Postdesign

After the design phase has concluded and the game is not going to be changed further, we enter the **postdesign** phase. The most important part of this phase is compiling the game – in many cases, ANGELINA would not create an executable when it was finished. ANGELINA$_2$ modified Actionscript files and then ran a compiler to produce a game, while ANGELINA$_5$ required the manual moving and arrangement of files so it could be compiled into a finished binary executable. Although this may sound like a minor aspect of the system compared to coevolutionary systems and creative evaluation, the requirement for the intervention of a person is a major weakness, both from the perspective of the perception of observers, and the autonomy and independence of the system. We believe that it is crucial that the system can release its work on its own, so it can control how and when it disseminates its work. This is a big part of 'closing the loop' creatively – allowing the system to decide when it is finished.

Another aspect of postdesign was the preparation and compilation of framing information (Charnley, Pease, and Colton 2014). Across all versions which employed framing this took the form of textual commentaries, which ANGELINA would construct using templates filled with data about decisions made by the system. Because of the dense nature of the design phase, the framing information never referenced the development of the game itself – instead, it discussed intentions and motivations, and the origins of the media used in the games. The main reason for this was that it was hard to convey meaningful things about the design phase because players never knew what it was or what took place in it, they only ever saw the finished game. Almost everyone we spoke to asked about the design process itself, highlighting how little we communicated about it.

Presence In CC Systems

ANGELINA began as an abstract system with little emphasis on creative decision-making, and evolved over time to take into account issues like real-world context, self-evaluation, and framing. However, this did not change the fundamental structure of the system, or the AI techniques it used to achieve its goals, and this caused problems as the project developed. In this section, we highlight some of the

most common issues we identified in the design and execution of ANGELINA, and then introduce a common thread which ties them together: the concept of *presence* in a computationally creative system.

Opacity Of The Design Phase

A working definition of Computational Creativity makes reference to 'unbiased observers' who assess software as being creative or not based on how it behaves (Colton and Wiggins 2012). The most important aspect of ANGELINA's process, where the game is actually designed, was not only impossible to observe, but would also be impossible for many to understand even if they could observe it (Cook, Colton, and Gow 2013). As such, they can only guess at how ANGELINA develops games in the Design Phase.

We found this was a particular problem for ANGELINA, because observers were unable to distinguish work done by ANGELINA from work done by us in building the system. For example, for ANGELINA$_3$ we built a template game for the system to modify. This meant that aspects of the game like how the camera moved, the player's appearance or the control scheme were all out of ANGELINA's control. Players frequently attributed this to ANGELINA, however, because they didn't know enough about the design process to know what the system was actually responsible for.

Short Term Impact

The second major problem, which we believe may have been exacerbated by the opaque nature of the design phase, is that people were unimpressed or confused by the higher-level structure of the system. We believe that because they were unable to assess ANGELINA effectively by examining its design work, they instead looked to other aspects of the system for evidence of creative autonomy, such as how the software operates not when creating a single artefact, but across its entire lifespan. Common questions asked by both journalists and the general public included:

- Can the system learn new things?
- How does the system decide when to make a new game?
- How many games has it made?
- Can it play other people's games and learn from them?

None of these questions refer to the act of designing a game: instead they touch on the long-term growth of the system; whether the system has creative independence; what the system's legacy is; and whether it can engage with the existing culture of videogames. It speaks to a higher-level thinking about AI, one that is willing to accept that the system can perform certain tasks, but now wants to know what those tasks are in aid of, and whether they are in the context of a wider environment.

Previous versions of ANGELINA lacked good answers to these questions. We decided when to run ANGELINA, and what it should make a game about. The act of creation left almost no long-term impact on ANGELINA – in a rare case, ANGELINA$_3$ would remember past topics and respond slightly differently if it came across them again. It could not engage with other creators (the closest we came was having other creators engage with it, when its game jam entries were judged by its peers). It did not develop over time, and it had no long-term goals – it was designed to run as a blank slate, create something, and then stop. Our use of co-operative co-evolution also hampered us, because during co-evolution every aspect of the game's design was constantly in flux, *and* constantly dependent on every other aspect of the design, which made it hard to extend the design phase or add systems that modified or adapted the design process.

Different Creative Modalities

Besides the opacity of the design phase and the lack of long-term structure, there was also a technical problem we encountered when designing previous iterations of ANGELINA, namely that we found it hard to balance high-level design work and fine-grained discovery work. ANGELINA$_4$ used metaprogramming to invent new game mechanics, but in order to do so, it would exhaustively simulate an existing game with specific objective functions. This was intensive work just to discover a new game mechanic in a fairly stable search space. On the opposite end, ANGELINA$_5$ used parameterised game mechanics and simple level design algorithms so it could rapidly prototype and test games, allowing it to explore a higher-level design space more quickly. Combining these in a single system would be difficult. There would be no way to have both activities working simultaneously in a cooperative coevolutionary system because a small change in the high-level design would seriously disrupt the low-level search for new game mechanics.

At the same time, discovering game mechanics, or any other kind of detailed design knowledge, felt like something that should happen outside of an ordinary game creation loop. Yet there was no obvious place to put this, because ANGELINA was only run when we intended for it to design a game, and there was no clear plan for how newly-discovered design knowledge would be fed into ANGELINA's normal game design process. Discovering new game mechanics produces no immediately consumable creative artefact, yet seems like an important part of the system's growth. Who should decide what kind of task it undertakes, or when it undertakes them? It felt difficult to build a system in the way we had been, while enabling all these different motivations and modalities for creativity.

Product, Process and Presence

(Jordanous 2015) notes that "traditionally within computational creativity the focus has been on... [a] system's Product or its Processes" – by which they mean the artefacts produced by software, and the way in which those artefacts were made. In reflecting on our work on ANGELINA, we propose a third element to this line of thinking, which we call *Presence*. Presence is the impact a computationally creative system has on its environment, and the impact the environment has on that system in return. It accumulates over time, and encompasses both tangible things (such as a system's knowledge of its past work) and intangible things (the perception the public has of the system). To put these three elements in context: *product* relates to a single artefact, at

the moment it is consumed by an observer; *process* relates to the means by which that artefact came into being; *presence* relates to the impact of the system's history and environment on the process being undertaken, and the impact the resulting product will have on the future of the system and its environment. Presence is not merely the sum total of the system's output – it also includes things such as how the system influences and is influenced by its peers and critics; how the system relates to and is perceived by its audience; how the system sets and achieves goals for itself; how it learns and grows through creating.

At the time of writing, many systems in Computational Creativity have a presence, but it is almost entirely sustained by the involvement of the system's designers. For example, ANGELINA's presence is sustained by talks given by the authors about the system; by a website of projects that is manually maintained by the authors; by public events like entering game jams or releasing games that are chosen by the authors. It is not detrimental to the system's creativity to have people contribute to a system's presence, indeed it may never be possible to fully separate a system's creator from that system's legacy. However, the software must also have some responsibility in creating and managing its own presence, as a step towards us handing over creative responsibility to a system, and enabling software to have creative autonomy not just over what they make, but on their place in the wider world, and any creative communities they may exist within the context of.

In redesigning ANGELINA, our intentions were to find a way to increase this sense of presence in the system. We aimed to do so not simply by adding features to the software, but by designing the structure of ANGELINA in such a way that it would not need to be rebuilt as often as previous versions, and in a way that encouraged future additions to the system to preserve and expand the system's presence. In the next section we describe how we went about doing this.

Designing For Presence In ANGELINA

We designed the latest version of the system, ANGELINA$_6$, to take into account the conclusions of our project review, and the identification of presence as a lacking element in the system to date. The result is a design which we hope will not only produce better games, and frame them in richer and more compelling ways, but also a system with more control over its own presence, and a better foundation on which to build new features and do more research in the future, without rebuilding the system again. In this section, we provide a high-level overview of some of the system's most important features, before discussing lessons learned from this process in the following section.

Overview

ANGELINA$_6$ maintains a database of active game designs that it is working on, each of which has a metadata file which tracks important statistics about the project and tasks that need to be completed. When the system has completed a task, it checks this database and selects a project that has active tasks and is not on hold. After ANGELINA$_6$ loads the project by parsing the game's project file (written in a domain-specific game description language, described below), it selects a pending task on the project's to-do list and passes control to a module designed to complete that particular task. When ANGELINA$_6$ has completed its current task, it will modify the project file, updating the game if the task was completed successfully, and making notes in the metadata file for future work. If the game is ready to release or needs to be abandoned, it may perform additional steps here, otherwise it files the game back in the database and begins the cycle of selecting a new game to work on.

Continuous Creativity

One of the most important changes in this version of ANGELINA$_6$ is that the system does not have a defined start or end point. Instead of being turned on, creating a game, and then stopping, ANGELINA$_6$ is designed to constantly cycle through a database of active game projects with associated lists of pending tasks. It can also choose to start a new project to add to this list, or declare a project as abandoned or released to remove it from the list. Theoretically speaking, ANGELINA$_6$ can now run indefinitely, moving between creative tasks and producing games forever. In practice, we do not actually run ANGELINA$_6$ perpetually for reasons of energy conservation and hardware strain, but the system resumes exactly what it was last doing when it is restarted. The decision to design ANGELINA$_6$ as a continuous system was one of the earliest decisions we made when redesigning the software, and it forms the core of what we call *continuous creativity*, a way of building software that we describe in detail in (Cook 2017).

Making the system continuous is the most important design decision made in the new version of ANGELINA$_6$. A continuous structure gives us the ability to have the system change the order in which it performs creative tasks, or even change which creative projects it works on. It also raises questions about how these systems should be upgraded, how often – if at all – a system should be reset, and how the data within them should be structured. By forcing ourselves to commit to the notion that this software is always working, always existing in the world, we change our relationship with the software as creators, and put the long-term presence of the software above short-term research goals. The software is now in control of what it does and when it does it, it decides when to start work on something and when to change to something else or stop entirely. This shifts the relationship between the public, the system and us as its programmers, and puts more emphasis on the system's autonomy and independence.

Task-Driven Design

Prior versions of ANGELINA$_6$ used cooperative coevolution to simultaneously design all aspects of a game together. The main advantage we perceived this as having was that all aspects of the design could be solved simultaneously, and therefore any part of the design could 'lead' and influence other parts. However, this approach came with many drawbacks, including a higher complexity for observers and overcorrection between subsystems. The new ANGELINA$_6$ es-

chews this approach, and instead breaks up each part of the design into its own separate task – thus, when ANGELINA$_6$ is designing a level now it is only designing a level, and nothing else is happening at the same time. We provide ANGELINA$_6$ with a catalogue of tasks for different purposes, which can be parameterised to specialise a task to a particular game or phase of development. Current tasks include designing rulesets, sketching level concepts, designing levels, and assigning art and colour schemes to the game's content. Each task employs its own process for completing its work: for example, level design uses a mix of evolutionary design and MCTS for testing levels (Browne et al. 2012), while ruleset design uses abductive reasoning and answer set programming (Gelfond and Lifschitz 1988).

The most immediate benefit from this is clarity and transparency: it's now simple to express to observers what the system is doing at any given time, because it is only ever doing one thing. We also gain a new kind of nonlinearity to the way the system works, despite giving up the simultaneous nature of the coevolutionary approach. Currently, ANGELINA$_6$ is given a loose structure in how it designs games: design a ruleset; experiment with level design to confirm the ruleset's potential; design several larger levels to fit the game; release the game. However, in the future as ANGELINA$_6$'s task catalogue expands, we plan to give the system the autonomy to dynamically change its task queues to fit a particular game. For example, it might discover that it cannot design many interesting levels, and so schedules a task to extend the ruleset and make it more complex. After doing this, it will schedule further level design tasks, as well as another task to evaluate the older levels and confirm they still work in the context of the new ruleset.

This is all possible because the task system is entirely modular and written with clear interfaces that ANGELINA$_6$ can use. For example, the Level Design task can be customised to change aspects such as the size of the level, the complexity of the desired solution, and the depth with which to search for solutions. This means ANGELINA$_6$ can easily adjust the same modular task to accommodate exploratory design, simple level design, and deep ruleset exploration. In the future, we hope this will lead to ANGELINA$_6$ having a lot of autonomy over how it works, and provide it with opportunities to refine its work and go back and improve on tasks that are already completed.

Longer Design Cycles

Continuous work shifts the emphasis of the software away from producing a single game and towards growth as a game designer over many creative acts – in other words, it emphasises *presence* over *process*. An individual game project is now just a section in the long-term existence of the system, rather than the target outcome of running the system. This also removes the need to generate a game by a deadline – previously we would want ANGELINA$_6$ to produce a game relatively quickly because the system could not save its work, and thus had to create a game in a single execution. A continuous system doesn't need to work in this way, and so we are using this as an opportunity to build a system that spends weeks producing an artefact rather than hours.

One of the reasons people were fascinated by how long it took ANGELINA$_6$ to produce a game is that AI, particularly creative AI, can seem mysterious to the general public. Even though ANGELINA$_6$'s games were not blockbuster-quality, the idea that it only took four or six hours to make one seemed impressive. One of the benefits of changing the timeframe of ANGELINA$_6$ is that it shifts its work from being on the scale of software to being on the scale of humans. This isn't just a perceptual benefit, however. Working more slowly means we have more opportunities for observers to engage with the process – ANGELINA$_6$ can tweet about a game idea it has had, and blog about the development process over multiple weeks, culminating in the release of the game. This allows people to see development and growth during creation, not just after the fact as has been the case before. This is a new approach to framing for the project.

This also opens up opportunities for ANGELINA$_6$ to work with people more directly. Game developers frequently collaborate with others to complete a game, and also send their game to playtesters to get feedback. Up until now, ANGELINA$_6$'s short timescale has meant that it has had to play its own games, and acquire pre-existing art and music from online sources. But working over weeks means ANGELINA$_6$ can send out games to testers and wait for feedback, or send commissions to artists and musicians and wait for responses. These are exciting opportunities for research into human-software collaboration, and longer timescales make it feel like a natural part of the process.

Custom Engine & Description Language

Past versions of ANGELINA$_6$ used game templates designed by hand which they then modified and exported. This restricted the systems more, but made it easier to build them in the first place, and much easier to disseminate the finished games which was always a key objective. This version of ANGELINA$_6$, much like ANGELINA$_5$, is built in the Unity game development environment, but unlike ANGELINA$_5$ its output is a text file, not a Unity project. This text file contains the entire game described in a custom description language we have made, inspired by VGDL (Schaul 2013) and Puzzlescript (Lavelle 2014). The text files act like game cartridges or ROMs, in that they are fed into another application which we have created, which interprets the language and runs the game. The interpreter uses a custom game engine which we built in Unity, meaning that both ANGELINA$_6$ and the interpreter use exactly the same code to run games.

The immediate advantage to this approach is that it makes it easier to distribute games, and easier for ANGELINA$_6$ to release them. Almost all prior versions of ANGELINA$_6$ needed some manual work by a person to compile and distribute its games, but now it can upload that text file to game marketplaces, or send them via email. Using a description language also has additional benefits though, primarily that it allows other people to easily write games that can be interpreted by ANGELINA$_6$. This means that for the first time ANGELINA$_6$ can play games designed by other people and learn design knowledge from them, or evaluate them and give feedback to the designer. We intend to explore this in future work and investigate how ANGELINA$_6$ can work

```
{
    "trigger": "OVERLAP enemy playerpiece",
    "code": [
        "DESTROY $2",
        "SFX punch2",
    ]
},
{
    "trigger": "ENDTURN",
    "code": [
        "DO_AI_HUNT enemy playerpiece"
    ]
}
```

Figure 1: A code snippet from a game description.

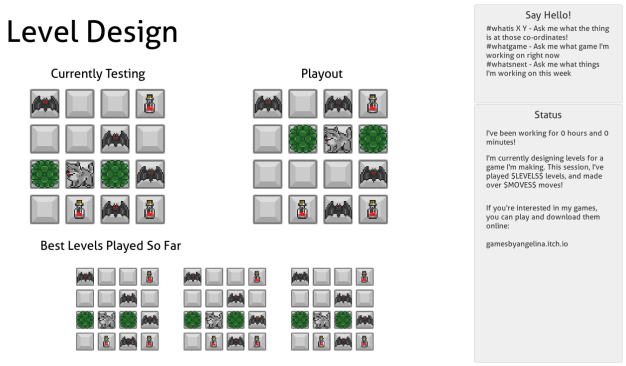

Figure 2: A screenshot of ANGELINA's workspace, on the Level Design view.

with and learn from other people.

Our decision to use a custom language rather than an existing one is partly down to other languages not quite fitting our needs – Puzzlescript is quite abstract for software generation, and we felt the VGDL was too prescriptive. The most important reason, however, is that we wanted a language which was flexible enough to enable ANGELINA$_6$ to extend it in the future. Figure 1 shows part of a game description, to illustrate this. The part shown defines two rules in the game, each structured as a trigger condition followed by a list of things that happen when the condition is met. The top rule says that when an enemy overlaps with the player, the player piece is destroyed. The second rule says that when a turn ends, enemies move towards the player.

We've designed the description language so that ANGELINA$_6$ can engage with it at different levels depending on the kind of design work it is doing. At the highest level, it treats the entire code in Figure 1 as a single game concept that it can add into a game without modification (it adds enemies which chase the player and kill them). ANGELINA$_6$ has a catalogue of these mechanics that it can use to rapidly develop games with concepts that are known to be useful. It can also create its own rules, using the language to design triggers and lists of effects. This is a lower-level action that would probably be performed outside of a game design, in a prototyping phase where it experiments with new game ideas. When it finds useful or interesting mechanics, it can add them into its catalogue to use later in higher-level design tasks. Finally, it can work at an even lower-level, and use metaprogramming and code generation techniques to add new keywords (like `DESTROY`) to the language. We aim to extend our previous work in mechanic discovery to do this (Cook et al. 2013). These new keywords could then be used in low-level mechanic design, and ultimately filter up into high-level catalogues of mechanics. Being able to work at different levels fits in with the overall philosophy of continuous creation and growth.

Discussion and Future Work

The notion of presence is relevant to all areas of Computational Creativity, and we believe that many of the engineering decisions made in the latest version of ANGELINA$_6$ also have wide applicability to most of the domains that Computational Creativity has been applied to. As with *process*, there is no step-by-step guide to emphasising *presence* in a system. Nevertheless, we believe the following three features of ANGELINA$_6$ were useful in helping drive us towards building a system with more presence:

- **Continuous** – The system has no beginning or end, and seamlessly moves between tasks and projects, recording its progress, starting and stopping projects as it sees fit.
- **Modular** – The system selects from several tasks in order to advance a project, and tackles a single activity at a time.
- **Long-Term** – The system is built with long timeframes in mind; a single project can take a long time to produce, and a single project is less significant than the impact it has on the system's creative development.

These features help us think about the system beyond a single creative act, and about how the system will change over time, what interfaces it presents to the outside world, and the ways in which it can be extended in the future.

Continuous creativity and exploring the notion of presence opens up a lot of future work – investigating how to motivate long-term systems, how to build more complex framing profiles using historical data, as well as raising questions about how a system should be tested, how it should be reset, and whether unofficial execution of the software, such as during development, constitutes as part of the system's official 'history'. For now, we identify two key areas of future work that we intend to pursue:

Long Timescale Visualisation

ANGELINA$_6$'s creative process is now much more accessible, because it only works on one task at a time, and organises itself in a way that is perhaps closer to how people organise large creative tasks (such as maintaining lists of short-term goals, or evaluating progress as a project moves towards completion). This means that people can now watch ANGELINA$_6$ as it creates, something which has been trialled before by software such as The Painting Fool (Colton and Ventura 2014). However, the continuous nature of the software changes the tenor of this experience, as people can

now observe the system over longer periods of time, which allows them to notice changes in a particular artefact being created, as well as growth in the system itself.

We plan to explore this by having ANGELINA$_6$ livestream its creative process on online streaming sites such as Twitch.tv. This will allow people to watch ANGELINA$_6$ as it works, and we have designed a visual frontend to the software that tries to represent ANGELINA$_6$'s creative activities in a way that faithfully represents the underlying algorithmic activity. Figure 2 shows one of the design screens. At the time of writing we have completed some trial streams, and also had ANGELINA$_6$ exhibit at a major games expo, which we hope to report on in a future publication.

Richer Framing

Because ANGELINA$_6$ records its progress in such detail, including maintaining lists of tasks and projects, version history for its games and notes on the success or failure of tasks, the system has a huge amount of data at its disposal about the creative process. This is somewhat necessitated by the continuous, modular nature of the system – since it has to be able to suspend projects and transfer data between modules, it has to keep meticulous records and copious metadata about each creative project it starts. This, combined with the slower, long-term nature of the system, opens up powerful new ways for the system to frame its work to observers.

We also have opportunities for framing *during the creative process* which is not something we believe has been attempted before in Computational Creativity. Because the creative process aims to last days or even weeks, we can have the system comment and reflect on its process while it is still working on a project. This provides even greater opportunities than before to have a system remark on changes in direction, leaps in progress, and crucial decisions. Even though many computationally creative systems exhibit these properties, they are rarely highlighted during short, intense generative processes. Continuously creative systems, however, provide natural points between tasks where a system can reassess its work and identify next steps, or make notes about the process for later framing.

In tandem with our livestream experiments, we are also developing ANGELINA$_6$ to engage in more active forms of framing, by allowing viewers of the livestream to ask ANGELINA$_6$ questions using a limited set of phrases the system understands. This allows viewers to retrieve framing information from the system dynamically, at different stages of development. Examples of these questions include: asking what project the system is working on currently; asking what other tasks they have to do next; or asking what a specific game piece does in the game being worked on. We plan to study the impact of this active framing on the perception of the software as creative; we anticipate it will have a positive impact and help connect observers more closely to the creative process during creation, rather than only allowing engagement after the fact.

Related Work

Automated game design is a growing area of study, and is beginning to fork into a set of subproblems that share a common core. One of these is the generation of test cases for general game playing – unseen games, or games designed to specifically test a particular area, would help research into general game playing, and also has benefits for certain kinds of competitive human play. Research that closely links general game playing to game design, such as (Khalifa et al. 2017) and (Bontrager et al. 2016), are forging a link between these problem domains, as is the emergence of a game design track in the General Video Game AI competition (Liebana et al. 2016). This challenge-first approach can be traced back to work by (Togelius and Schmidhuber 2008), for example, who designed rulesets for games based on how hard they were to learn.

Another application area is automated design as support for other game designers. The Sentient Sketchbook (Liapis, Yannakakis, and Togelius 2013) is a tool that assists in the level design process, with an innovative interface that helps sort and visualise important information and opportunities to the user. In a similar vein, (Shaker, Shaker, and Togelius 2013) present Ropossum, an interactive level design tool for physics-driven games like Cut The Rope. While these tools don't try to take on the entire game design process, they use very similar techniques and show how AI tools can assist in a variety of different game design contexts.

Similarly, work by Osborn proposes that this broader goal of 'discovering game design knowledge' should be one of the field's objectives (Osborn, Summerville, and Mateas 2017), something that echoes a paper by Smith, one of the earliest game design papers at ICCC, which proposed the concept of a machine that discovered game design knowledge through experimentation (Smith and Mateas 2011). We are already seeing work aimed at discovering or translating game design knowledge, for example through machine vision and interpretation (Guzdial and Riedl 2016) (Guzdial, Li, and Riedl 2017), or reasoning about game design knowledge using formal methods (Martens et al. 2016).

Many other systems exist simply to further the broader goal of building software that can design games. Sometimes this is focused on a narrow genre, such as Barros et al's work on mystery puzzle games (Barros, Liapis, and Togelius 2016), while others attempt broader systems that target a less complex but also less fixed structure, such as the Game-o-Matic, possibly the most successful automated game design project to date (Treanor et al. 2012). These systems often tackle the hard problems of cultural knowledge, too, such as Nelson and Mateas' system which built simple games from plain text descriptions (Nelson and Mateas 2008).

Conclusions

In this paper, we described a new version of ANGELINA, rebuilt to reflect our changing ideas about computational creativity and automated game design. We introduced the notion of *presence* in computationally creative systems, to complement well-established notions of *process* and *product*. We showed that past versions of ANGELINA lacked presence, and how redesigning a new version of the software to be continuously creative helped guide us towards a design that can take more responsibility for its own presence and long-term growth. Finally, we laid out our immediate next

steps for ANGELINA$_6$, and some next steps for those looking to incorporate these ideas into their own systems too.

Acknowledgments

This work is funded by EC FP7 grant 621403 (ERA Chair: Games Research Opportunities). The authors wish to thank the reviewers for their feedback on the paper; the Max Planck Institute for Software Systems; the automated game design community for many years of discussion that shaped ANGELINA up to this point; and Chris Donlan, for asking the right questions.

References

Barros, G. A. B.; Liapis, A.; and Togelius, J. 2016. Murder mystery generation from open data. In *Proceedings of the International Conference on Computational Creativity*.

Bontrager, P.; Khalifa, A.; Mendes, A.; and Togelius, J. 2016. Matching games and algorithms for general video game playing. In *Proceedings of the Conference on Artificial Intelligence in Interactive Digital Entertainment*.

Browne, C.; Powley, E.; Whitehouse, D.; Lucas, S.; Cowling, P. I.; Tavener, S.; Perez, D.; Samothrakis, S.; Colton, S.; and et al. 2012. A survey of monte carlo tree search methods. *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI*.

Charnley, J.; Pease, A.; and Colton, S. 2014. On the notion of framing in computational creativity.

Colton, S., and Ventura, D. 2014. You can't know my mind: A festival of computational creativity. In *Proceedings of the International Conference on Computational Creativity*.

Colton, S., and Wiggins, G. A. 2012. Computational creativity: The final frontier? In *ECAI*, Frontiers in Artificial Intelligence and Applications.

Cook, M.; Colton, S.; Raad, A.; and Gow, J. 2013. Mechanic miner: Reflection-driven game mechanic discovery and level design. In *Proceedings of the EVOGames Workshop, Applications of Evolutionary Computation Conference*.

Cook, M.; Colton, S.; and Gow, J. 2013. Nobody's a critic: On the evaluation of creative code generators. In *Proceedings of the International Conference on Computational Creativity*.

Cook, M.; Colton, S.; and Gow, J. 2017. The ANGELINA videogame design system - part I. *IEEE Trans. Comput. Intellig. and AI in Games* 9(2):192–203.

Cook, M. 2015. *Cooperative Coevolution For Computational Creativity: A Case Study In Videogame Design*. Ph.D. Dissertation, Imperial College, London.

Cook, M. 2017. A vision for continuous automated game design. In *Proceedings of the Experimental AI and Games Workshop at AIIDE*.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. 1070–1080. MIT Press.

Guzdial, M., and Riedl, M. O. 2016. Game level generation from gameplay videos. In *Proceedings of the Conference on Artificial Intelligence in Interactive Digital Entertainment*.

Guzdial, M.; Li, B.; and Riedl, M. O. 2017. Game engine learning from video. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*.

Jordanous, A. 2015. Four pppperspectives on computational creativity. In *AISB 2015 Symposium on Computational Creativity*.

Khalifa, A.; Green, M. C.; Liebana, D. P.; and Togelius, J. 2017. General video game rule generation. In *IEEE Conference on Computational Intelligence and Games*.

Lavelle, S. 2014. Puzzlescript. http://www.puzzlescript.net/.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013. Sentient sketchbook: Computer-aided game level authoring. In *Proceedings of ACM Conference on Foundations of Digital Games*.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2014. Computational game creativity. In *Proceedings of the International Conference on Computational Creativity*.

Liebana, D. P.; Samothrakis, S.; Togelius, J.; Schaul, T.; and Lucas, S. M. 2016. General video game AI: competition, challenges and opportunities. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Martens, C.; Summerville, A.; Mateas, M.; Osborn, J.; Harmon, S.; Wardrip-Fruin, N.; and Jhala, A. 2016. Proceduralist readings, procedurally. In *Proceedings of the Experimental AI and Games Workshop at AIIDE*.

Nelson, M. J., and Mateas, M. 2008. An interactive game-design assistant. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*.

Osborn, J. C.; Summerville, A.; and Mateas, M. 2017. Automated game design learning. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*.

Potter, M. A., and Jong, K. A. D. 1994. A cooperative coevolutionary approach to function optimization. In *Proceedings of the International Conference on Evolutionary Computation*.

Schaul, T. 2013. A video game description language for model-based or interactive learning. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*. IEEE Press.

Shaker, N.; Shaker, M.; and Togelius, J. 2013. Ropossum: An authoring tool for designing, optimizing and solving cut the rope levels. In *Proceedings of the Conference on Artificial Intelligence in Interactive Digital Entertainment*.

Smith, A. M., and Mateas, M. 2011. Knowledge-level creativity in game design. In *Proceedings of the International Conference on Computational Creativity*.

Togelius, J., and Schmidhuber, J. 2008. An experiment in automatic game design. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*.

Treanor, M.; Schweizer, B.; Bogost, I.; and Mateas, M. 2012. The micro-rhetorics of game-o-matic. In *Proceedings of the Foundations of Digital Games Conference*. ACM.