Automating Game Design In Three Dimensions

Michael Cook and **Simon Colton** and **Jeremy Gow**[1]

Abstract. We describe *ANGELINA-5*, a new iteration of the *ANGELINA* framework for investigating and building software which automates the process of videogame design. *ANGELINA-5* is the first automated game design tool that produces 3D games. We outline here the system's structure, the challenges inherent in building an automated game designer in a modern game engine, and we discuss the future research directions for the project.

1 INTRODUCTION

Procedural content generation [8, 6] is a staple of both games development and research, but automating the process of designing an entire game is territory we are only just beginning to explore. Many new challenges arise when attempting to design not just one element of a videogame, but all elements simultaneously, and many problems that were easily ignored when generating a single piece of content must be dealt with directly when part of a larger design challenge.

Unlike procedural content generation, automated game design does not have obvious applications to the modern games industry. While procedural content generators can act as sources of surprise and excitement for players, or help designers in producing larger games more quickly, the goal of automated game design is to produce a system which does not help anyone, nor need anyone to help it. Nevertheless, as an avenue of research it asks many interesting questions about how and why we design games, and can be a catalyst for novel procedural generation research at the same time. By forcing a system to design everything, we reveal the less obvious design tasks we might otherwise take for granted, or not consider getting a piece of software to do itself.

We describe here the development and structure of *ANGELINA-5*, the newest iteration of *ANGELINA*, an automated game designer which employs computational evolution. Previous versions of *ANGELINA* have examined different aspects of automating game design, explored different technologies and genres, and investigated the problem both from the side of games research and the side of Computational Creativity research. *ANGELINA-5* marks a new phase for the research, making a step forward technologically, but also philosophically too. By building *ANGELINA-5* in Unity, a flexible and modern development environment, we are laying the foundation for *ANGELINA* to develop as a platform for automated game design, rather than a system custom-built for designing one specific type of game.

The rest of the paper is organised as follows: in section 2 we briefly describe *ANGELINA-5*'s background, and the choice of Unity as a development platform; in section 3 we describe *ANGELINA-5*'s internal structure as an evolutionary system; and in section 4 we describe future directions for the research project. In section 5 we draw some brief conclusions.

[1] Computational Creativity Group, Goldsmiths, University of London. ccg.doc.gold.ac.uk

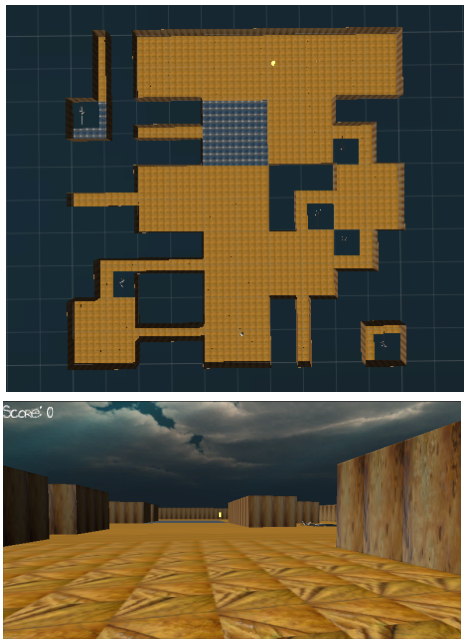

Figure 1. Images from *Hit The Bulls-Spy*, a game designed by *ANGELINA-5*. Top: A view of the entire game map. Bottom: A screenshot of the final game while running.

2 ANGELINA and Unity

ANGELINA is a cooperative coevolutionary [7] system for automating the process of videogame design. There have been several different versions of *ANGELINA* in the past [2, 3, 4], each tackling a different kind of game design problem often on different platforms or game engines. The latest version of the system, *ANGELINA-5*, represents a large step forward and a considerable shift in the platform that *ANGELINA* is built upon.

The research aims of the project are concerned with automating game design and the procedural creation of content, but also the addressing of issues in Computational Creativity. Later versions of *ANGELINA* investigated questions of thematic control, context and framing of design decisions, and whether *ANGELINA* could discover new game mechanics without additional game design knowledge [5].

ANGELINA-5 is built as an extension to the Unity game development environment[2]. Unity is an extremely popular, versatile and powerful game engine that ships with a comprehensive development environment that is also highly extensible. Unity games can be deployed to web browsers, to all major desktop operating systems as

[2] http://www.unity3d.com

native applications, to every modern games console and handheld device, and most smartphone operating systems including iOS, Android and Blackberry.

This versatility means that distribution of *ANGELINA-5*'s games is much simpler than before, and the games can be distributed to a wider variety of people, hopefully increasing the success of future user studies, as well as improving the dissemination of our results. Unity also supports both three- and two-dimensional game development, meaning that for the first time we can begin to investigate the automation of fully-3D game design. This allows *ANGELINA-5* to explore a wider variety of game types, and also strengthens the image of *ANGELINA-5* as a game designer in terms of using contemporary technology. This is important in terms of Computational Creativity, since the perception of a piece of software as being creative is determined by many factors besides simply what the software outputs [1].

3 System Structure

In this section, we describe the architecture of *ANGELINA-5* by detailing the steps the system goes through to produce a game, from the beginning input theme to compiling and exporting a final game.

3.1 Predesign Phase

ANGELINA-5 begins by being given a word or phrase which acts as a theme for the game it is about to design. This method of defining a starting point for a game design is derived from *game jams*, i.e. game development contests where people congregate to design games in short timeframes, constrained by commonly-shared themes. Examples of themes might be fairly straightforward, such as 'fishing', or more abstract, such as 'alone'. In some cases the themes are intentionally very unusual or restricting in order to stimulate creativity – the theme for the 2013 *Global Game Jam* was the sound of a heart beating. Developers are encouraged to incorporate the theme into their game in whichever way they can, such as through the ruleset, the narrative, the visuals or some other way.

When an input theme is given, if it is longer than a single word, *ANGELINA-5* will first attempt to isolate a single word most likely to be a suitable theme. Single words work better than phrases for our current methods of media acquisition and framing, because many of these processes are based on querying web services that expect singular queries. However, it should be noted that this single-word approach is not a long-term solution, and better theme parsing is a point of future work. In order to choose a single word from a phrase, *ANGELINA-5* first removes all words which are not nouns, and then uses a frequency analysis against a large corpus of English text[3], in order to find the least common word in the input phrase. This approach was developed by analysing 150 game jam themes by hand and running similar filters on them. We found that the most prominent theming information tended to be in more specific words, particularly, nouns. The exception to this rule is where the theme includes meta-references to the game itself, such as 'build the level you play'. Once it has a chosen theme word, it then begins a predesign phase which prepares it for the act of designing a game.

Once *ANGELINA-5* has a theme word it attempts to expand the theme using word association databases[4]. We plan to replace this technique with a more relevant topic association approach in future, but for most applications word association provides a reasonable set

[3] http://www.kilgarriff.co.uk/bnc-readme.html
[4] http://wordassociations.net

Figure 2. Two textures from *ANGELINA-5*'s repository of over 600.

of words relating to the source theme word. These word associations are combined with the theme word to provide a list of possible words relating to the game's overall theme. For example, the theme word *secret* would lead to a list of words including *secret*, *spy* and *mystery*. A typical list of associations runs to about thirty words. These associations are then used to perform a series of media searches, one for each association, in order to build a database of assets for use in theming the final game. *ANGELINA-5* downloads public domain fonts from DaFont[5], 3D models from TF3DM[6] and sound effects from FreeSound[7]. These media are archived as they are downloaded, so that they can be retrieved quickly for future designs which include the same associations.

ANGELINA-5 also generates a *zone plan* which defines a number of themed zones for use within the game design. A zone is a collection of a floor texture, a wall texture, a 3D model for use as scenery, and a sound effect. The sound effect and scenery model are both randomly selected from the media downloaded from the associations list. In order to select the texture, *ANGELINA-5* searches through a list of 622 tagged texture files for ones which are related to one or more of the association words. A relationship can be established in one of two ways: first, it can compare the associations with the filename or folder name of the textures, which are categorised roughly according to their type (such as 'clouds' or 'paper'). Secondly, it can call on a database of word associations mined using crowdsourcing via Twitter.

ANGELINA-5 regularly posts random untagged texture files to its Twitter account[8] and asks its followers to provide single words which they associate with the image. These are retrieved and recorded in a database file, and used as a secondary means to relate associations to textures in the case that the filename match fails. Figure 2 shows two examples from the database of textures. If no matches are found through either method, *ANGELINA-5* selects textures randomly for the zones.

Once *ANGELINA-5* has selected two textures and randomly chosen a 3D model to act as scenery (we describe scenery later) and a sound effect for each zone, the zone map is complete. Before it proceeds to the main design phase, *ANGELINA-5* will generate a title for the game, and select a piece of music. The game's title is generated using a rhyming dictionary and a corpus of popular culture references, including famous examples of media such as music and books, as well as idioms and common sayings. It attempts to create puns using these resources and the list of source word associations, using a similar approach to the one described in [4].

[5] http://www.dafont.com/
[6] http://tf3dm.com/
[7] http://freesound.org/
[8] http://www.twitter.com/angelinasgames

To select a piece of music, *ANGELINA-5* attempts to choose a suitable mood for the game. It first takes the main theme word, and passes it to Metaphor Magnet[9][9] to obtain feelings people express in relation to the theme word. Metaphor Magnet is a tool for exploring a space of metaphors, mined from Google N-Grams. It has an array of features that are built on top of this concept, including the ability to show feelings people commonly express about a topic, such as poetic or metaphorical qualities of something, with the knowledge that these feelings are backed up by concrete examples in the N-Gram corpus.

As an illustration, if we submit the word *winter* to Metaphor Magnet, we are presented with a number of possible metaphors for winter, such as a 'frightening night' or a 'refreshing spring'. By selecting one of these, *ANGELINA-5* can use feelings that Metaphor Magnet has corpus evidence for - e.g., winter in the context of a frightening night is commonly described as 'frightening'. This feeling is chosen as the base mood for the music chosen by *ANGELINA-5*. It now has to relate this emotion to a piece of music. The music database *ANGELINA-5* currently uses is Incompetech[10], which categorises its music pieces according to twenty different moods. In order to relate the mood discovered through Metaphor Magnet with an appropriate tagged mood in Incompetech, we use DisCo[11] to rate the semantic similarity between each of the 20 known emotions and the one discovered emotion. The most similar emotion is used as the search mood for music, and a piece of music is randomly selected from the resulting pieces.

3.2 Design Phase

As with *ANGELINA-3* described in [4], *ANGELINA-5* is composed of several evolutionary systems that work in tandem with one another to cooperatively evolve a game design. Each evolutionary system has two aspects to its fitness function – internal, objective rules that are considered to be unchanging regardless of the overall game design, and external, subjective rules that take into account what properties the current most fit game design has and adjusts its fitness evaluation accordingly. In order to evaluate these subjective rules for a given member of a population, *ANGELINA-5* takes the most fit example from every other evolutionary process, combines them together to form a game, and then simulates playing that game in real-time. For more details on coooperative coevolution, see [7]. For more details on our specific use of CCE in *ANGELINA*, see [2].

In *ANGELINA-5*, there are currently four separate evolutionary processes, or species. We briefly describe them below:

- Level Design – which forms a basic layout of solid space in the game world. The top image in Figure 1 shows a birds-eye view of a level designed by *ANGELINA-5*. Level designs are currently built out of smaller tiles which are selected from a library of hand-designed tiles and arranged into a variable-size array. In Figure 1, the size of the map is five tiles wide by five tiles high. A tile is a ten by ten array of integers denoting solid ground, empty space or scenery. Scenery regions are impassable to the player, and when the game is exported they are replaced with large, static 3D models for theming purposes.
- Zoning – which describes the visual and aural qualities of different regions of the game world. Zones are defined in the predesign phase, and during evolution a *zone map* is evolved: an array of integers relating each tile in the Level Design to one of the premade zones.
- Placement – which describes the start position of the player, and the position of the level exit. The primary objective in all of *ANGELINA-5*'s games is currently to reach the exit. In addition, a Placement defines the number and starting position of the game's *entities*. Entities are objects which are placed in the game world and given code to execute to play a role in the game's systems and rules. A Placement contains a list of starting positions for each type of entity – currently all games by *ANGELINA-5* include exactly two entity types.
- Ruleset – which describes the set of *behaviours* possessed by each entity. In Unity, 'behaviour' is an overloaded term used to describe any piece of code which implements a particular interface. In the current version of *ANGELINA-5*, we have supplied a stock of behaviours which can be attached to the entities in *ANGELINA-5*'s games to form a basic ruleset. These behaviours include providing motion for the entity (such as random walks, or wall following) and adding mechanical rules (such as killing a player, or providing score when collected). Expanding this set with automatically generated code is a point of future work, see [5] for details.

Each of these four processes evolve their populations in isolation, according to various fitness criteria, normally expressed as parameters which can be easily varied, so as to give *ANGELINA-5* the ability to alter its own fitness functions in the future. Currently, all parameters are set through experimentation to find values which produce an interesting variety of outputs. The fitness criteria are as follows:

- Level Designs are selected to maximise the size of the largest contiguous island, whilst simultaneously avoiding overfitting by limiting fitness to a maximum island size. This encourages level designs in which the tiles join up to form a single level space, but avoids the situation where the entire level is one open expanse by penalising levels which are too full of solid tiles. A level design is further penalised if the player or exit start position is in open space.
- Zone Maps are selected to maximise connectedness in zones of the same type. This means that a zone map which has two Zone 1 zones separated by a Zone 2 scores lower than a zone map which has a single contiguous Zone 1 zone and another single Zone 2 zone. This is done to provide consistency in when and how often a zone is encountered by the player.
- Placements are selected to maximise spread of entity placements across the map, but are penalised for any placements, including player or exit placements, which are not on solid ground. Placements are also selected to maximise the distance of the path from the start position to the exit position, with a penalty if no such path exists.
- Rulesets are selected to maximise the number of rules fired in a simulation of a game. *ANGELINA-5* records which rules fire during an execution of the game, using a simple player controller which attempts to follow a direct path to the exit. Rulesets are penalised if there is no way for the player to gain score or die, but does not guarantee both score gain and death are in the game.

It should be noted that many of these fitness criteria are in place only to complete *ANGELINA-5* as a game design system. We intend to replace all of these over the course of *ANGELINA-5*'s development by giving the system the ability to create its own fitness criteria. These might therefore be considered baseline criteria for producing a complete game design.

A typical evolutionary setup for *ANGELINA-5* consists of a population size of 40 for each of the four evolutionary species, and a run of

[9] http://ngrams.ucd.ie/metaphor-magnet-acl/
[10] http://www.incompetech.org
[11] http://www.linguatools.de/disco/disco_en.html

50 generations for the system as a whole, meaning that each species undergoes 50 generations of evolution itself. We utilise one-point crossover and single-element mutation for all four species, since representation is almost entirely array-based. Selection is elitist, and we carry forward the parents of the previous generation, something which we found useful in previous versions of *ANGELINA* due to the volatile nature of cooperative coevolutionary systems.

3.3 Postdesign Phase

When *ANGELINA-5* has completed the set number of generations and completed a game design, the game export process begins. Unity games are meant to be developed inside a single project which contains all the art and audio assets for the game, the data, the levels, the code and logic. Unity has export features that compile these various components together into a single package for a chosen platform (such as iOS). However, in the case of *ANGELINA-5*, it is *ANGELINA* that is the Unity project, not any single game that it develops. This means that the asset folders contain databases of models used in the past, music that has been downloaded, metadata and information about *ANGELINA-5* as a system, and so on. Exporting the games as-is is therefore not possible, as Unity cannot be told to avoid exporting certain resources, and would attempt to export gigabytes of data for each small game developed.

For this reason, and because of a desire to archive games designed by the system, we have *ANGELINA-5* export all the relevant information about a game design into a separate folder. This includes a text file describing the level design and the locations of various resources, as well as the asset files such as 3D models and texture files. This folder can then be read by a standalone Unity project that only imports the necessary resources, and can then export executable game binaries. This means that the games can be archived successfully, since the assets are contained within a single named folder, and also that only relevant game assets are exported by Unity, keeping the size of the finished games to a minimum.

4 FUTURE WORK

The version of *ANGELINA* set out in this paper represents a foundation on which many individual strands of research will be built, each extending the system in a new way and exploring new issues in automated game design. As such, many elements of the system as it stands today are subject to change. In this section, we briefly describe some of our aims.

- **Code Generation for Entity Behaviour** In [5] we used metaprogramming techniques to invent new game mechanics for a 2D platform game by inspecting a game's codebase, generating new code, and automatically playing the game. We intend to explore this approach further, this time in the more challenging but equally more promising Unity platform. Behaviours for entities in *ANGELINA-5*'s games, which currently provide the basis for the game's ruleset, will be automatically generated by *ANGELINA-5*. This will probably take place outside of a normal game design evolution, producing game mechanics which *ANGELINA-5* records for use in future games.
- **Modular Fitness Functions** The current fitness criteria are simplistic and designed to provide a skeleton implementation for *ANGELINA-5* to build upon. In future, we want to explore the system's capacity to produce its own fitness functions, expressing preferences for content of a particular type, both for gameplay aims as well as aesthetic ones. This approach will first explore basic parameterisation, with an emphasis on the system's ability to find interesting niches within the evolutionary space, and then look towards full generation of code for fitness functions.
- **Richer Game Design** To date, all versions of *ANGELINA* have created games with single levels. *ANGELINA-5* will develop multi-level games which introduce progressions that introduce the player to game mechanics gradually, increase in difficulty, and involve simple narrative arcs. In addition, we also want to improve *ANGELINA-5*'s ability to understand input phrases and themes, and read them in more creative ways. Simple improvements like the use of parts-of-speech taggers and word sense disambiguation would greatly improve the accuracy of *ANGELINA-5*'s theme analysis, but we also want to investigate the system's ability to relate a theme directly to its gameplay and rules, not just its visual and audio content.

5 SUMMARY

We have introduced *ANGELINA-5*, the latest iteration of *ANGELINA*, an automated game designer. We have motivated the work in the context of the modern games industry as well as computational creativity research, and given an outline of the current capability of the system, as well as a general overview of its internal structure. We discussed *ANGELINA-5*'s place in the context of other work in automated game design and procedural content generation, and looked ahead to how we intend to develop the system in the future.

ACKNOWLEDGEMENTS

The authors wish to thank Tony Veale for helpful discussions and insight, and the reviews for this paper which offered constructive feedback on the project. This work has been funded by EPSRC grants EP/L00206X and EP/J004049.

REFERENCES

[1] Simon Colton, 'Creativity versus the perception of creativity in computational systems', in *AAAI Spring Symposium: Creative Intelligent Systems*, (2008).

[2] Michael Cook and Simon Colton, 'Multi-faceted evolution of simple arcade games', in *Proceedings of the IEEE Conference on Computational Intelligence and Games*, (2011).

[3] Michael Cook and Simon Colton, 'Initial results from co-operative coevolution for automated platformer design', in *Proceedings of the Applications of Evolutionary Computation*, (2012).

[4] Michael Cook, Simon Colton, and Alison Pease, 'Aesthetic considerations for automated platformer design', in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference*, (2012).

[5] Michael Cook, Simon Colton, Azalea Raad, and Jeremy Gow, 'Mechanic miner: Reflection-driven game mechanic discovery and level design', in *Proceedings of 16th European Conference on the Applications of Evolutionary Computation*, (2013).

[6] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup, 'Procedural content generation for games: A survey', *ACM Trans. Multimedia Comput. Commun. Appl.*, **9**(1), 1:1–1:22, (February 2013).

[7] Mitchell A. Potter and Kenneth A. De Jong, 'Cooperative coevolution: An architecture for evolving coadapted subcomponents', *Evolutionary Computing*, **8**(1), (2000).

[8] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne, 'Search-based procedural content generation: A taxonomy and survey', *IEEE Trans. Comput. Intellig. and AI in Games*, (2011).

[9] Tony Veale, 'From conceptual "mash-ups" to "bad-ass" blends: A robust computational model of conceptual blending', in *Proceedings of the Third International Conference on Computational Creativity*, (2012).