Towards a Computational Reading of Emergence in Experimental Game Design

Simon Colton, Mark J. Nelson, Rob Saunders, Edward J. Powley, Swen Gaudl and Michael Cook
The MetaMakers Institute, Games Academy, Falmouth University, UK

metamakers.falmouth.ac.uk

Abstract

In any prolonged creative act, there may be moments when an interesting and/or surprising aspect of the artefact being created, or a related idea, emerges without prior knowledge of the creator. Such emergent properties can be capitalised on to drive the creative process. With the Gamika iOS app, we have made it possible to create novel casual game levels in minutes and hours rather than the usual days and weeks. This has enabled us to undertake and analyse game design sessions with a think aloud methodology, focusing on moments of emergence and how they influenced the level design. This has in turn led us to an initial computational reading of emergence in game design, where we imagine how an automated game designer could recognise and take advantage of unexpected changes in aspects such as aesthetics, gameplay and playing strategies which arise during the creative process.

Introduction

Making novel video games is technically quite difficult, and involves numerous skills, from graphic design to programming. We are interested in democratising game design, so that anyone and everyone can make simple games, much like they can make simple drawings and write simple stories, then learn to add sophistication to their creations. With the *Gamika* app, we have developed an iOS casual creator (Compton and Mateas 2015) that enables users to create full multi-level casual games directly on the device for which they are intended. The making of even fairly simple casual games is usually a time consuming process, with the effort often taking weeks and months. Where it is possible to use a handheld app to create games without programming, to the best of our knowledge, designers are limited to: skinning existing games, as per Playr (playr.us) or authoring levels in an existing game world, as per Sketch Nation (sketchnation.com). In contrast, Gamika enables rapid development of completely new casual games in minutes and hours, through a method of choosing an existing game level and then altering it until a new game emerges. As described in the next section, the user interface to Gamika provides access to drawing tools, generative art methods, search for variations via random mutation and the fine-grained setting of 284 numerical parameters to define the game mechanics for physics-based casual games.

In Gamika, we have defined a space of casual games via a breakdown of game levels into a set of numerical parameters. The idea of mapping a space of games has some similarities with systems such as VGDL (Schaul 2013) and PuzzleScript (puzzlescript.net). With these systems, the space of games is mapped to a space of hierarchical code structures, rather than a space of numerical vectors, as with Gamika. Gamika is also set apart from VGDL and PuzzleScript by its use of simulated physics. That is, whereas those systems define explicit movement rules for in-game objects, Gamika specifies only the physical properties of the objects and the environment, from which movement emerges. This reliance on *emergence* changes how the space is navigated: on one hand, it reduces the ease of finding a specific design that users may have in mind, but on the other, it increases the chances of the parameters combining in unanticipated and serendipitous (Pease et al. 2013) ways. The affordance of this emergence and how it could be used by automated game designers is the topic under investigation here.

The rapid development of game levels means that the creative process enabled by Gamika can be recorded and studied in short, encapsulated sessions. We describe below three case studies whereby a novice, intermediate and expert user of Gamika made levels for new games and used a think-aloud methodology to capture aspects of the process and their thoughts about the game being developed. The purpose of the sessions was to highlight those occurrences of emergence, where novel ideas for the game/level – which were specifically not imagined in advance – were found and used opportunistically to improve (or at least alter) the game.

Our ultimate aim is for Gamika to be accepted as an automated game designer (AGD), much in the same mould as the ANGELINA system (Cook, Colton, and Gow 2016). To this end, we provide some thoughts on a *computational reading* of emergence, based on insights arising from the design sessions. We imagine an automated game designer producing games in such a way as to opportunistically take advantage of emergent aspects of the game level it is producing. To do this, we specify some required components of the AGD, a setting within which end-to-end game design could occur, and a partial characterisation of moments of emergence, in terms of computational analogies of the human-centric moments seen in the design sessions. We conclude by discussing emergence from a design psychology perspective, and we describe future directions for the Gamika project.

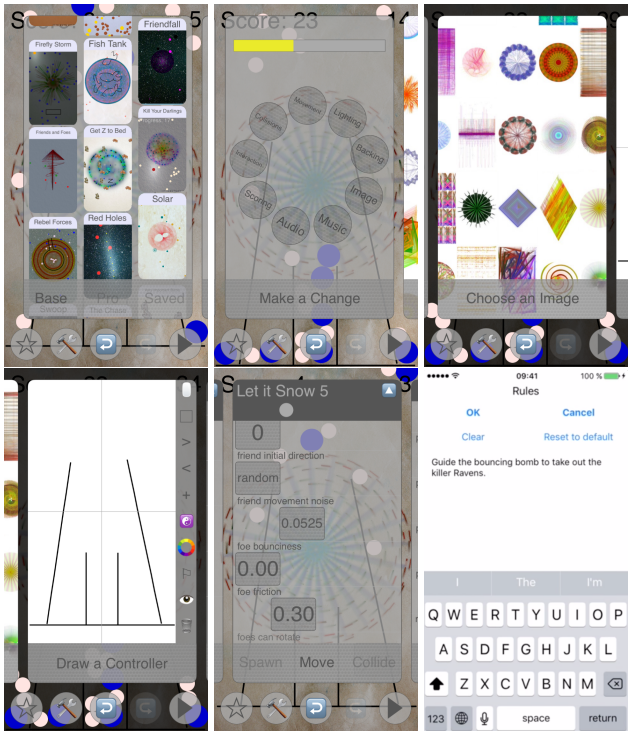

Figure 1: Base camp, mutation, generative art, drawing, parameter settings and game text design screens in Gamika.

The Gamika Handheld App

Gamika is an iOS application which enables users to make entire multi-level casual games directly on their device. Each level contains objects with simulated physical properties such as mass, restitution and velocity, within a simulated physics environment of various forces. There are three types of objects: friends, foes and the controller – these names simply provide useful handles as a lens through which the user can understand and remember properties and interactions. Determining how each type of object looks, is spawned, is attracted to other objects (or other locations), is affected by forces, and collides with other objects dictates the physical environment. The user can also specify how players interact with the objects, normally by tapping and dragging the controller, friends and foes. They can also specify how points for score, health and lives are accrued and lost, and how levels are completed, or failed. Finally, the user can supply some *game text* which provides a small narrative about how the level should be played, and gives background to the in-game characters.

To avoid a situation where there is an ominous blank canvas, users are expected to follow a pattern of: (i) choosing an existing preset game level from the *base camp* which is close to the kind of thing they might want to produce, then (ii) modifying it until it is either a new version of the level or an entirely different one. There are a number of design screens which enable the user to make these modifications, as depicted in figure 1. Firstly, the user can choose to make alterations to the level using a dial-like interface which performs random mutations on the genome of the game. The choice of button on the dial to rotate dictates which aspect of the level is mutated (e.g., lighting, collisions, movements, etc.) and the amount of rotation dictates how much the random mutation alters the current values. Secondly, the user can choose a piece of abstract art (Colton, Cook, and Raad 2011) and/or a hand-drawn motif for the controller through bespoke screens.

In a third set of screens, the user can specify the numerical parameters which dictate (a) the look of the level, e.g., background image, size/colour/image for the friends/foes, etc. (b) lighting effects, including ambient light and spotlights on various objects (c) how often and where friends and foes are spawned, and how many are allowed on-screen at any one time (d) how friends and foes move through attractive and repulsive forces (e) what happens when friends, foes and the controller collide with each other (f) six contributions to counters for each of: score, lives and health, and (g) how these counters should end the game in a win or a loss. Further details of Gamika, including the breakdown of game levels into component parts, and how it fits into the wider context of hand-held casual creators for game design, is given in (Powley et al. 2016). We concentrate here on some case studies of the usage of Gamika, and in particular how aspects of game levels emerge from user interaction with the design interface and the game itself.

Case Studies

Gamika empowers designers to produce games with wholly new game mechanics, interaction mechanisms and aesthetics. The speed at which this can be done means that entire game design sessions can be easily recorded and analysed. In the sessions described below, we recorded video of the iPhone/iPod screen where Gamika was being used, with a live audio narrative by the designer, encouraged to describe their ideas, intentions and motivations, i.e., via the *think aloud* protocol (Kuusela and Paul 2000).

Designers were particularly encouraged to be aware of unforeseen ideas that arise during the design/re-design and playtesting of game levels. This has no doubt biased the sessions towards a particular mode of creation. However, our aim was not to prove that emergence does happen, but rather to chart the different ways in which it can occur, in order to suggest computational equivalents. Hence, in this context, it was favourable to bias the designers towards seeking out and capitalising on novelties which arise during the design and playtesting of levels of a game. The three sessions below represent three levels of expertise in using the Gamika App: expert (author 1), intermediate (author 3) and novice (author 2). This setup was employed to explore the possibility of the computational equivalent changing strategies as it becomes more expert in using the casual creator.

Expert User

This design session was split into four sub-sessions of roughly 30 minutes, with an overall time of 2 hours and 10 minutes. The designer produced five preliminary sketch games and five levels of the final game, polished and tested

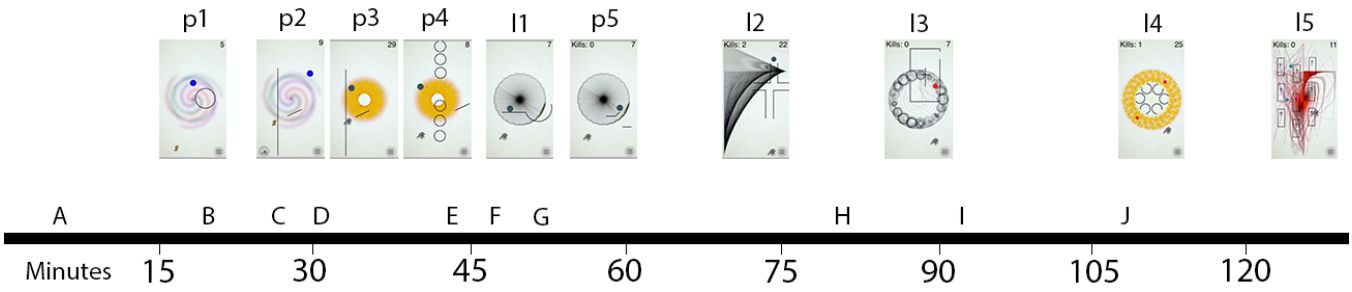

Figure 2: Approximate timeline for the expert game design session, showing when (p)rototypes and (l)evels of the final game were saved, and when emergent moments A to J occurred.

sufficiently to be able to show to others, as per the notion of *curation analysis* in (Colton and Wiggins 2012).

The designer originally aimed to produce a puzzle game, where the player has to work out how best to move the controller in order to bounce a ball into a particular position, similar to mobile games such as DropFlip (dropflipgame.com), where the aim is to bounce a ball into a bucket. However, towards the end of the second sub-session, this aim was abandoned in favour of an action game where the player must learn a skill of bouncing/catching/rolling/throwing and generally cajoling the ball with the controller, to get it into the right place at the right time. The designer lamented that this is usual and that they have not yet managed to design a satisfying puzzle game – we plan to improve Gamika in light of this criticism.

In addition to the gameplay, the game text also changed during the session. Originally the ball was labelled as food to feed a cute character known as a 'Bicho' (from the Spanish for little animal or little devil). However, at the start of the third session, a decision was made to change this to a narrative about the killing of ravens rendered in a Gothic style, and this was emphasised with a macabre theme running through the levels of the final game, i.e., with game controllers depicted as objects associated with death: a scythe, a cross, a coffin, some rosary beads and a graveyard. The following ten moments labelled A1 to J1 were recorded as examples were aspects of the game emerged without prior planning. These, along with the screenshots of the prototype and game levels produced by the expert designer are depicted on the timeline of figure 2.

A1 5m. While attempting to place the Bicho character at the bottom of the screen, the designer realised that it was placed half off the screen which was not desirable, and that it couldn't be put in the right place. This problem was solved by making the Bicho character attracted to the opposite side of the screen (the top), so that it moved very slowly.

B1 22m. Playing the game made the designer realise that the controller could collide with the Bicho character. This wasn't desirable, as it meant that the game could be too easily controlled that way. This led to a fix which uncovered a new game mechanic: that passing the controller through the Bicho character could be used to more easily solve the level.

C1 27m. The designer played the level in a new way, which was essentially bouncing the ball repeatedly. This gave him the idea for a new level (which was not eventually used) where the point is to keep the bomb away from the Bicho character, while it traversed the screen.

D1 30m. The designer played back the think aloud recording from the first sub-session and, on hearing the word "bounce" repeatedly, was given the idea of thinking of the ball character as a bouncing bomb, rather than an item of food. This solved the problem of the ball hitting the Bicho being rather a violent way of feeding them, and brought in a more suitable narrative, i.e., of destroying rather than feeding. This was capitalised on by changing the Bicho character to a Gothic raven character, making both the ball and the raven explode on collision, and altering the game text to talk about killing a raven rather than feeding a Bicho.

E1 44m. In the middle of attempting to design a level where the dropping ball rolled around a semicircular controller towards the raven, the designer realised that the process of catching the ball and then throwing and/or rolling it was more fun, as it afforded a higher level of control.

F1 46m. The designer realised that the game was too easy, as the player could simply catch the ball and move it to be on top of the rising raven character, hence avoiding the need to throw/roll the ball. To solve this, they added a line to the controller to stop it being dragged the requisite amount.

G1 49m. To fine tune the controller to encourage the throwing mechanic, the designer reduced the part-circumference of the semicircle. On viewing the altered controller, they realised that it looked like a scythe, and decided that this fitted the narrative of destroying the ravens well (as the grim reaper carries a scythe). The designer changed the game text to reflect this extra narrative element.

H1 86m. The level with a drawn coffin controller was deemed too easy, so the designer gave the ball some random noise and random initial direction; moved the entry point for the ball to the coffin; and removed the right hand wall, to increase the difficulty. To visually emphasise the randomness to the player, the ball colour was changed to red, and the bouncing bomb was portrayed with more autonomy in

the game text, i.e., as less of a collaborator/tool and more of something that had to be controlled.

I1 93m. The art image for the third level possibly gave the designer the idea of choosing a rosary bead motif for the controller in the fourth level (the designer wasn't sure).

J1 108m. Accidentally tapping the bomb character exploded it, which the designer decided was not desirable. However, when using the interface to turn this off, they saw that another option was that tapping could reverse (temporarily) the direction of the bomb. The designer tried this a few times and decided it was an interesting game mechanic, and kept it in the level. Tapping to reverse was made the main control mechanic in the final (fifth) level.

Intermediate User

This design session involved a single two-hour session, during which an intermediate designer – with some previous experience of making simple games from scratch with Gamika – managed to create a novel game with three levels of increasing difficulty. The designer started with the idea of not making a game but attempting to implement an aquarium-like environment as a meditative experience, or possibly a toy. The goal was not to create a faithful simulation of fish but rather something more akin to "seamonkeys", by relying on a Perlin noise field to move elements around. The early attempts to build an aquarium-like environment proved successful, in as far as Gamika allows, but trying to remove an obvious controller from the screen proved unsatisfactory. Attempting to use a minimal controller lead to a contemplative game-like experience and, after some experimentation with different ways of controlling the movement of the controller and scoring the accumulation of friends and foes on the controller, an enjoyable game emerged.

The final game, called "primordial scoop", consists of a large number of small elements moving around in a fluid environment. The friend and foe objects destroy each other on contact but are immediately replaced elsewhere at a random location. The player controls a single line that is similarly adrift in the fluid, and the player can only affect the orientation of the line as it is moved around by the currents. When one of the small elements touches the controller it sticks. The goal is to collect all of the small elements onto the controller within a time limit, in this case set at 2 minutes. At the end of the session, the designer had produced three levels with increasing numbers of elements, i.e., 20, 40, 60, to test how well the game could be introduced to a novice player and how enjoyable/frustrating the game becomes as the number of elements to collect increases. The following moments of emergence were observed.

A2 10m. The designer started with the general theme of an aquarium-like experience, not necessarily a game, but something more contemplative. Increasing the strength of the noise field produced a nice feeling similar to many small creatures floating in a liquid, but clearly not fish because they are being swept around by waves. Making all of the particles very small enhanced the feeling of having many small sea creatures, and significantly increasing the number of particles again made it feel like a sea of tiny creatures in some sort of "primordial soup". The designer noted that the idea of a "primordial soup" was interesting and decided to follow this direction, still with the intention of producing a display or toy, rather than a game.

B2 21m. The designer attempted to enhance the feeling of a "primordial soup" by making the friends and foes almost the same colour and allowing them to stick to each other to form large clusters. However, allowing the friends and foes to create clusters without limit quickly ran into problems as the physics engine became unstable. Setting collisions between friends and foes to destroy both created a pleasing display that continued to evolve over time without the problem of the clusters getting too big too quickly.

C2 27m. Changing the appearance of friends and foes to be as close as possible to the same colour created an interesting movement field which was pleasing in its uniformity. However, the designer noted that the controller wasn't doing anything and looked out of place. Trying to remove the drawn controller altogether resulted in a background artwork being shown, which was unexpected and not what the designer wanted. Not knowing any other way to make the controller disappear, the designer put it back and made it a similar size and shape to the friends and foes, noting that the controller still looked out of place, as it moved in a different way.

D2 46m. The designer added lights to some of the friends and foes, to make an interesting visual effect. However, the number of lights permitted was much smaller than the number of friends and foes on screen, which made the lights that are drawn seem arbitrary. The designer changed the behaviour of the friends and foes coming into contact with the controller to stick to it, and tried making only the stuck friends and foes light up. While this created an attractive spotlight around the controller, the limit on the total number of lights still made it seem that stuck friends and foes are being lit up at random. The designer decided to drop the idea of any form of lighting.

E2 59m. Turning off the lights but keeping the behaviour of the friends and foes sticking to the controller, the designer commented that it started to feel somewhat like a game. However, the round controller shape that the designer had been using up to this point seemed limited and he began experimenting with other shapes. He started with a rectangle, but settled on something very minimal and barely visible against the background, namely a thin straight line, as the controller almost relies on the friends and foes to attach to it to make it visible, and the line very clearly shows how the controller is moved around in the fluid environment.

F2 81m. The designer decided that moving the controller directly with the finger didn't feel right, given the floating feel of the rest of the screen. Moving the controller towards the finger didn't feel right either, as it felt like too direct control. Rotating the stick without moving it directly was satisfying, with not too much control, but enough to kind of

swim about, by bouncing off the walls, etc.

G2 96m. Having decided that gathering up the friends and foes through an indirect control mechanism had a good feel to it, the designer turned to the question of scoring. Some sort of countdown timer initially seemed appropriate. The designer started by counting the largest clusters, but this meant the goal was to create large clusters which again led to physics engine instability. While this can look interesting, the engine eventually slows down gameplay to a crawl, so a way incentivising the player to reduce the size of clusters was needed. The designer's first attempt was to change the scoring mechanism such that the game counted +1 for friends stuck to the controller and -1 for foes stuck to the controller, similar to another game designed with Gamika, called "Solar". The designer made the player's aim to be having a score of zero at the end of the time limit, which meant either building up perfectly equal amounts of friends and foes stuck to the controller, or trying to keep the controller clean by destroying friends with foes and foes with friends. The designer noted that this felt like a good way to play towards the strengths of the system in terms of the physics engine and appealed to the idea of keeping the game cooperative with the environment.

H2 111m. After playtesting the scoring mechanism, the designer found he didn't fully understand how scoring of clusters worked. Hence it was hard to figure out how to progress, because in many instances, picking up a new friend or foe onto a cluster on the controller didn't affect the score as expected. Changing the scoring mechanism to count all of the captured particles, and carefully controlling the number of friends on foes on screen, so that they are fixed from the beginning of the game and have to be "mopped up", made for much more satisfying gameplay.

I2 117m. Experimenting with different numbers of friends and foes produced a number of levels of increasing difficulty, suggesting that the game mechanic can be tailored to different skill levels.

Reflecting on the game, the intermediate designer noted that while the mechanic is simple and easily grasped with few elements on screen, the movement control required patience and working with the fluid game environment, rather than against it. The subtlety of control reminded the designer of some of the earliest arcade games, e.g., Lunar Lander, requiring a "light touch". The designer suggested a number of extensions to the game beyond increasing the number of elements, e.g., changing the shape and size of the controller, changing the stickiness of the controller, adding obstacles or increasing the number of types of elements on screen with different reactions between different types.

Novice User

This design session consisted of a single one-hour session, in which a novice designer produced a new game for the first time. The designer had previously used Gamika for one warm-up design session of 20 minutes, in which they produced a new level for an existing game, in order to gain basic familiarity with the interface. The designer started with the idea of trying to implement Flappy Bird within Gamika. It seemed unlikely that it would be possible to do so in a faithful way, but the idea served as a starting goal. As the initial attempts ran into difficulties, the game morphed and ended up not being at all similar to Flappy Bird, in fact much more dissimilar than Gamika actually requires (it is possible to implement something closer to Flappy Bird in Gamika, but that wasn't a hard goal, so was abandoned when other design directions presented themselves).

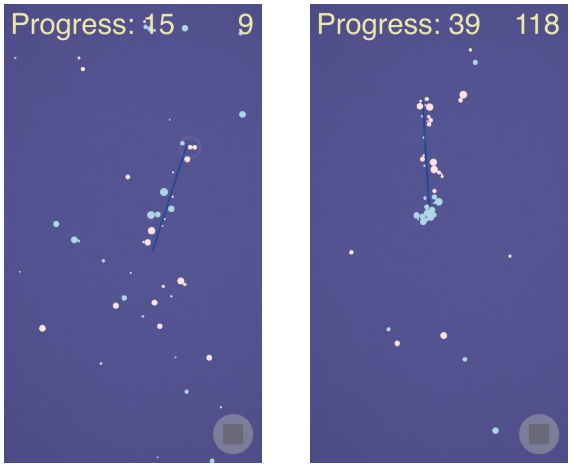

Figure 3: Screenshots from the final game design of the intermediate designer showing a typical state of play from the beginning and towards the end of a game.

The final game produced is dimly lit, with the player navigating glowing balls (which light the way) past obstacles without touching them. The main element retained from the Flappy Bird starting point is navigating left to right past gate-type obstacles without touching them. But rather than a fast-tapping style of gameplay, it has a steady-hand feel, as the player tries to avoid pulling the ball too close to a wall; the darkness meanwhile adds a second layer of challenge. The following moments of emergence were observed.

A3 15m. Through trial and error, the designer reached an interpretation of some features of Flappy Bird. Since Gamika has no side-scrolling, the original design of a single bird navigating a series of obstacles was not possible; instead, this interpretation had a fixed set of obstacles, with a series of balls spawning to navigate them in turn. The goal that obstacles should be avoided was implemented by having balls stick to the obstacles if they touch. However, there was not yet any control, and the designer was unable to figure out how to implement the core mechanic of Flappy Bird: gravity pulling the bird down while tapping flaps it upwards, at the same time as the bird moves at constant velocity horizontally. Instead, after browsing the options, he settled on controlling the ball in a non-tapping way, by having it pulled towards the finger touch position.

B3 21m. The designer noticed that having multiple balls on screen at the same time makes gameplay difficult, since they're all controlled with the same touch position simultaneously, and what's good for one may be wrong for the others. Options considered to address this were to either design levels around this synchronised control challenge, or limit the level to one non-stuck ball on screen at a time. The latter option was chosen, at least for a first level, reducing spawn rate so only one non-stuck ball is on screen at a time.

C3 36m. After experimenting with spawning positions, velocities, and obstacle design, the designer decided the game needed different aesthetics. Not sure what to do, he used Gamika's random mutation dial to try a different lighting arrangement. The game became dimly lit and the balls gave off light. The designer hadn't considered using lighting as part of the game's challenge (versus an aesthetic element), but the new lighting had the potential to make traversing the obstacles difficult in an interesting way, so he kept it.

D3 47m. Playing the game, the designer noticed that if a ball is stuck transitively (stuck to another stuck ball, but not directly to an obstacle), the game stops progressing, because no new balls are spawned. He looked to see if there was a way to excluded transitively stuck balls from the limit that inhibits spawning, but there didn't seem to be, so he removed the possibility of transitively stuck balls, by making ball-ball interactions produce a bounce.

E3 52m. The game became possibly too easy, since bouncing off stuck balls could provide a cushion to keep off the obstacles without having to really thread between them. The designer tried to compensate for this by making the level almost pitch-black, with only local light from the balls guiding the way, giving it an exploration-like element.

F3 60m. The designer converted a 15-second gameplay extract into an animated GIF to post to Twitter. He accidentally produced a version with heavy posterisation artefacts, due to using the default GIF web colour palette with a limited number of greys, as shown in Figure 4. This fortuitously produced a game (or rather, hypothetical gameplay video snippet) that much more closely matched the feel he had been hoping for with the dark level lit only by the aura of the ball: the discrete lit/unlit area and the cartoon-type shading works much better than the realistic lighting in the actual game. Gamika doesn't currently support such shader effects, however, so it was not possible to go back and implement this look in the game.

A Computational Reading of Emergence

The following is a hypothetical description of how an automated game designer (AGD) could use the Gamika software – either through alteration of the genome directly (a JSON file) or through the user interface – to produce a novel game, capitalising on emergent properties. To produce this, we have analysed the emergent moments in the case studies above, to provide potential computational analogies of them.

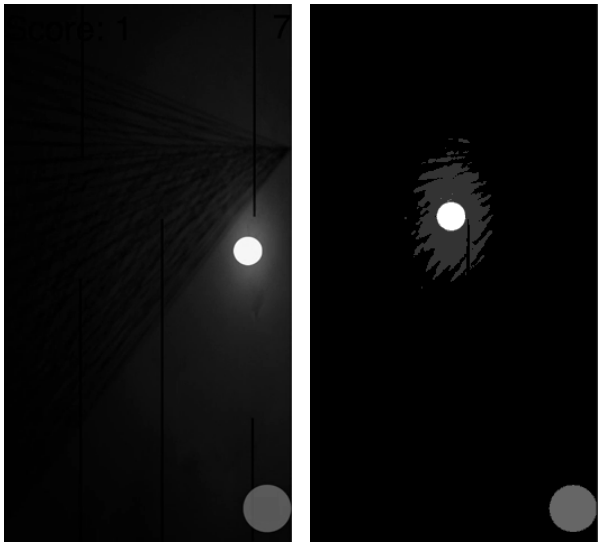

Figure 4: Screenshots from the final game design of the novice designer. Left as it appears in Gamika, and right after being accidentally posterised in a way fortuitously beneficial to the design.

Required Components of the AGD

We assume that the AGD has a suitable automated playtester (AP) which can try out various strategies for playing a game, and that the AGD has access to information about the complexity of each strategy, as suggested by (Nelson 2011; Nielsen et al. 2015). We also assume that the AGD has a parameter change predictor module (PCP) which can predict how parameter settings changes will alter gameplay in Gamika levels. Another requirement for the computational reading is that the AGD has a fun predictor (FP), with abilities to predict the level of fun that players might experience with certain game mechanics, levels and entire casual games. It is beyond the scope of this paper to describe a computational model of fun, but we would assume that it involves some level of novelty, some level of player control, some interesting physical attributes, etc. In the arena of casual gaming, the notion of 'juiciness' is related to fun, as per the 64th lens of gaming described in (Schell 2014). Our final requirement for the AGD is a game analyser (GA) which contains a machine vision module (MV) that is able to watch games being played, and analyse data from automated playtesting, to inform FP and PCP. We acknowledge that these requirements for an AGD are well beyond the current state of the art.

A Setting for End-to-End Game Design

The following is a rough approach to automated game generation, which stands as a best-case scenario (albeit not for emergence), wherein a designer could reproduce a well known game level in Gamika. This acts as a setting for the computational reading of emergence.

• **Choose an existing game.** A database of existing casual game levels, not necessarily implemented in Gamika, from

which one, EL, is chosen randomly or in some other systematic way. For instance, the database of Atari ROM games described in (Bellemare et al. 2013) would provide a suitable database.

• **Analyse the game.** The GA module analyses game code, assets, gameplay video, and machine-readable descriptions (e.g., VGDL) of EL, and, if possible, the AP module playtests the game to provide understanding of EL.

• **Choose a Gamika level as the basis for the new game.** Use the understanding of EL to compare to similar precalculated analyses of the levels in the Gamika base camp to choose the nearest neighbour, BC, to EL as the basis for the Gamika version.

• **Map game elements.** Produce a mapping from the game elements of EL to the friends, foes and controller objects of BC. Similarly map the initial placement of objects and the spawning regimes of EL to those of BC, in addition to the set of physical interactions such as collisions and movements. Finally, map the progress (score, health, lives) mechanisms from EL to BC, and do likewise for the visual and audio aesthetic elements.

• **Alter the game appropriately.** Where the mapped elements of EL and BC differ, choose the most appropriate parameterisation of the appropriate aspect of the level, and make the relevant change to BC.

Emergent Moments

In some cases, although probably quite rarely, it may be possible to start with an existing game idea and produce a suitably faithful version on Gamika. For instance, clones of asteroids, space invaders and frogger have been made by a human designer with Gamika. However, in our experience, producing even a poor version of an existing game is usually impossible because of the limitations of essentially finding a game in a space, rather than coding a specific solution. Hence, games designed with Gamika tend to emerge rather than being systematically constructed, so that the designer ends up with an interesting game, albeit not the one they had in mind originally. Below, we collate some of the emergent moments described above, and provide a computational equivalent, to provide some inspiration for a more formal model of how software could capitalise on emergence.

A1 A2 A3 Using GA and PCP to realise that aspects of EL are not possible, choose an alternative due to limitations in the expressivity of the levels allowable through the interface. Such a substitution might be done automatically through qualitative reasoning about space and movement (Forbus 1983; Cavazza et al. 2014).

B1 If inheriting from an existing Gamika game leaves residue parameters which produce unexpected gameplay, as assessed by GA, then: if FP predicts fun, keep the settings, otherwise turn off the settings which make that gameplay happen, as predicted by PCP.

C1 E2 Get the AP to undertake experimental gameplay not related to the objectives of the level. If FP predicts that this is more fun than currently, make that gameplay part of the objectives of the level.

D1 B2 Have the GA textually describe physical properties of game objects. Use these keywords to find facts or themes from an existing knowledge base, and alter game assets to fit the narrative. This could build on work in automated game skinning and theming (Nelson and Mateas 2008; Cook and Colton 2014).

E1 E2 Get the AP to once again experiment with strategies, and identify one which has a stronger level of control for AP or is predicted by FP to be more fun. Change the game to require the usage of that strategy, using PCP to suggest the alterations.

F1 G2 B3 E3 If AP tests of different strategies finds one which is too easy (in an obvious sense such as finishing quickly or maximising scores) or the level is too hard or obscure, e.g., it's not obvious how progress is being achieved, then use PCP to choose parameter changes to fix the problem – using AP to re-test the altered levels.

G1 Use MV to extract parts of a game visualisation that resemble some real-world object, then employ the AGD to add information about that object in the game text.

C3 Rely on serendipity to jump to a different part of the search space for some non-critical aspect of the game, e.g., the visual aesthetics, and use GA and FP to see if the changes alter the game mechanics and/or player enjoyment.

F3 Changes to the game's presentation layer (such as lighting and shading) modify the difficulty and experience, but only if the game is perceived visually via the screen; to notice this kind of effect, the AP module would need to play the games visually given only video of the screen, not given semantic information such as ball positions and velocity.

Discussion, Conclusions and Future Work

The Mechanic Miner module described in (Cook et al. 2013) as part of the ANGELINA project (Cook, Colton, and Gow 2016) encouraged emergence in automated game design by the software altering games at code level in order to assign a novel game mechanic to a player action, e.g., inversion of gravity. ANGELINA capitalised on this by building a level which could only be solved if the player employed the new mechanic (as checked by an automated playtester). In (Cook et al. 2013), the process stopped after a single new mechanic was introduced, but we could imagine an iterative process where more code changes were added and the game slowly emerged as a result.

The Gestalt psychologists recognised that emergence was an important part of human perception (Sternberg 2003). Studies of designers have explored the role of emergence in the design process, Gero included 'emergence' as one of the five core computational processes for creative design (Gero 1994). As a way of characterising emergence in design, Brown (1998) suggested that a reasonable definition of emergence in design might be that "An identifiable de-

sign property which has not been explicitly anticipated or explicitly represented in the current (partial) design can be said to be emergent." The requirement for a property to not be explicitly anticipated or explicitly represented goes some way to ensuring that an emergent property has the quality of being 'surprising' that is characteristic of emergence as typically used by designers when discussing their process.

In line with the Function-Behaviour-Structure framework for design processes, Gero has argued that there are three classes of emergence in designing: emergent structures, emergent behaviours and emergent functions. Emergent structures are the most obvious form, e.g., the perception of illusory contours in drawings suggesting structure that has not been explicitly represented. In the design sessions above, the emergence of visual features, such as the novice designer's accidental posterisation, could be reflected upon and integrated back into future game development.

Emergent behaviours are also familiar from the visual domain, becoming evident in the qualities such as groupings, alignment and symmetry, as studied by the Gestalt psychologists, as well as higher level features such as movement, rhythm and balance (Arnheim 1974). In the context of the game design sessions recorded, this might be recognised in the need of the intermediate designer to maintain a certain rhythm in the movement of the controller in relation to the friends and foes, the recognition of this constraint then formed the basis for the designer's choice of control mechanism for the player. Emergent functions require the discovery of unexpected uses, e.g., in the context of the sessions discussed above, this might be characterised as the use of the walls by the intermediate designer as a means of propulsion when playing the game, or the recognition of the symbolic significance of the scythe-like controller by the expert designer. Emergent functions brings with it a need for the design agent to ground their expectations of the game world in use, and possibly requiring them to bring in experience from outside the game world, and as such may require highly competent and flexible automated game players.

The computational reading of emergence in game design given here is currently very sketchy, and it will need to be fleshed out in the context of the emergent structures, behaviours and functions, and the context of serendipitous discoveries (Pease et al. 2013). As we develop Gamika as an automated game designer, we plan to add the modules required for emergence described above. We have already started work on the parameter change predictor, using decision tree learning to suggest parameter ranges for fine-tuning of the parameters. We have also implemented an automated playtester module which, in principle, enables any level to be played on the device, but in practice is not yet generic enough. We aim for Gamika to exploit emergent properties of games it is designing, to automatically produce interesting and engaging games of cultural and commercial value.

Acknowledgments

This work was funded by EC FP7 grant 621403 (ERA Chair: Games Research Opportunities).

References

Arnheim, R. 1974. *Art and Visual Perception: A Psychology of the Creative Eye*. University of California Press.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Brown, D. C. 1998. Functional emergence: A position paper. In *Proceedings of the International Conference on Artificial Intelligence in Design*.

Cavazza, M.; Hartley, S.; Lugrin, J.-L.; and Le Bras, M. 2014. Qualitative physics in virtual environments. In *Proceedings of the International Conference on Intelligence User Interfaces*, 54–61.

Colton, S., and Wiggins, G. 2012. Computational Creativity: The final frontier? In *Proceedings of the European Conference on Artificial Intelligence*.

Colton, S.; Cook, M.; and Raad, A. 2011. Ludic considerations of tablet-based evo-art. In *Proceedings of the EvoMusArt Workshop*.

Compton, K., and Mateas, M. 2015. Casual creators. In *Proceedings of the International Conference on Computational Creativity*.

Cook, M., and Colton, S. 2014. Ludus ex machina: Building a 3d game designer that competes alongside humans. In *Proceedings of the International Conference on Computational Creativity*.

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*.

Cook, M.; Colton, S.; and Gow, J. 2016. The ANGELINA videogame design system, parts I and II. *IEEE Trans. Comp. Intell. AI Games*.

Forbus, K. D. 1983. Qualitative reasoning about space and motion. In Gentner, D., and Stevens, A. L., eds., *Mental Models*. Psychology Press. 53–74.

Gero, J. S. 1994. Computational models of creative design processes. In Dartnall, T., ed., *AI and Creativity*. Kluwer. 269–281.

Kuusela, H., and Paul, P. 2000. A comparison of concurrent and retrospective verbal protocol analysis. *American Journal of Psychology* 113(3):387–404.

Nelson, M. J., and Mateas, M. 2008. An interactive game-design assistant. In *Proceedings of the International Conference on Intelligent User Interfaces*, 90–98.

Nelson, M. J. 2011. Game metrics without players: Strategies for understanding game artifacts. In *Proceedings of the AIIDE Workshop on Artificial Intelligence in the Game Design Process*, 14–18.

Nielsen, T. S.; Barros, G. A. B.; Togelius, J.; and Nelson, M. J. 2015. Towards generating arcade game rules with VGDL. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*.

Pease, A.; Colton, S.; Ramezani, R.; Charnley, J.; and Reed, K. 2013. A discussion on serendipity in creative systems. In *Proceedings of the International Conference on Computational Creativity*.

Powley, E.; Colton, S.; Gaudl, S.; Saunders, R.; and Nelson, M. 2016. Semi-automated level design via auto-playtesting for handheld casual game creation. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*.

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

Schell, J. 2014. *The Art of Game Design: A Book of Lenses*. CRC Press, 2 edition.

Sternberg, R. 2003. *Cognitive Psychology*. Thomson/Wadsworth.