2017-01-01

Mixed-initiative approaches to on-device mobile game design

Nelson, MJ

https://pearl.plymouth.ac.uk/handle/10026.1/20915

CEUR Workshop Proceedings

Mixed-Initiative Approaches to On-Device Mobile Game Design

Mark J. Nelson
Simon Colton
Edward J. Powley
Swen E. Gaudl
Peter Ivey
Rob Saunders
Blanca Pérez Ferrer
Michael Cook

The MetaMakers Institute
Falmouth University
Penryn, Cornwall, UK
metamakersinstitute.com

Abstract

Playing casual games is a wildly popular activity on smartphones. However, *designing* casual games is done by a smaller group of people, usually on desktop computers, using professional development tools. Our goal is to bring these activities closer together, in terms of who does them and how they do them. Our *Gamika Technology* platform is a 2D physics-based mobile game design environment. It comprises a 284-dimensional parametric design space, and poses mobile game design as the problem of navigating this space. We have built three mobile apps thus far to experiment with on-device, mixed-initiative navigation of the Gamika design space and some of its subspaces. We describe these apps here in terms of the initiatives that go into making a game with them, and how these are split between people and underlying AI software. Our overall goal is to democratise game design, so that anyone and everyone can make casual games directly on their mobile phones or tablets.

Author Keywords

Mobile games; mixed-initiative interfaces; automated game design; automated playtesting; design spaces.

ACM Classification Keywords

H.5.m. Information interfaces and presentation: Miscellaneous

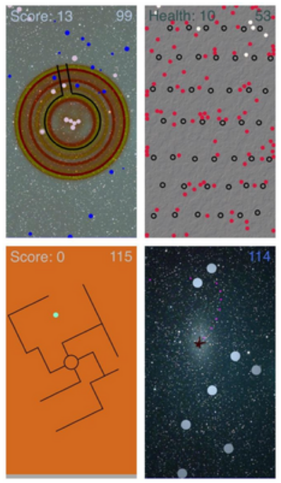
Figure 1: Four Gamika games

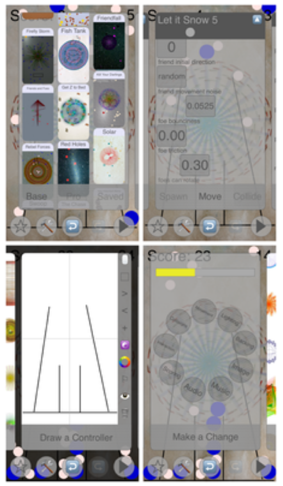
Figure 2: *Cillr* design panels

Introduction
At The MetaMakers Institute and its associated spinoff company MetaMakers Ltd. (metamakersinstitute.com), we are building apps for on-device, mixed-initiative design of games for mobile devices (smartphones and tablets). We explain here our overall approach in building the Gamika Technology platform that provides the basis for our work, and three apps built on that platform: *Cillr*, *Wevva* and *No Second Chance*.

Our goal is to allow users to create mobile games directly on the devices that they play these games on. Today, many people play casual games, but a much smaller number of people design such apps, and they tend to do so on traditional computers, using environments such as Unity, XCode or Android Studio that are entirely dissimilar to the context in which games are played (and much less accessible).

This goal of on-device creation is why we are building mixed-initiative, co-creative design tools [5]. On-device design of mobile games must balance two issues: giving users enough control to feel ownership of their output, but automating enough aspects of the design exercise so that making games on a smartphone feels like an enjoyable, empowering exploration of design possibilities, not cumbersome small-screen programming. Hence our goals are more aligned with the genre of creative apps dubbed "casual creators" [2] than with visual-programming tools.

Our approach begins by parameterising 2D physics-based games. The basis of the Gamika Technology platform is a 2D game engine parameterised by 284 features that we have identified as core to a diverse range of casual games. This set includes parameters controlling the physics engine, player interactions and scoring/win-conditions. Physics parameters expose common features of a 2D physics engine: object spawn rates/locations, collision responses, attractive/repulsive forces, etc. Interaction parameters specify how players interact with the physics world, such as when and how objects respond to the player tapping or dragging on the screen. Scoring and win-condition parameters specify how events impact the game outcome (the more narrowly conceived "rules" of the game). A more detailed parameter overview is given in [4, Section III].

Games for the Gamika platform are encoded in 284-parameter *chromosomes* (the term is borrowed from evolutionary algorithms, as automated game generation is a goal), augmented with data such as graphical and sound assets. Given a chromosome, the Gamika platform can run the game via an interpreter that allows run-time changes to the game specifications (see Fig. 1 for a few examples).

Cillr: Navigating the full design space
Given the Gamika Technology platform, the problem of on-device navigation of a high-dimensional game-design space is still far from trivial. But compared to the open-ended problem of full game design, we believe it serves as a more suitable starting point for the affordances of mobile devices.

Equally importantly, design spaces can be both *manually* and *automatically* navigated, allowing for mixed-initiative design. We are working on both interface- and automation-oriented solutions to on-device design-space navigation, and experimenting with using the two together. *Cillr*, our in-house app for building Gamika games (Fig. 2), implements baseline versions of both.

The simplest way of manually navigating a 284-dimensional design space to give the user 284 sliders, with which they can set each parameter. While simplistic approach, this does work fairly effectively in *Cillr*. The 284 sliders are grouped into categories with related functionalities to make them more discoverable (spawning-related sliders, collision-related, etc.).

The simplest way of automatically navigating a large parameter space is to randomise the parameters. However, we have found that this produces too low a yield of playable games, and hence *Cillr* mutates subsets of parameters from existing games instead. Randomly mutating multiple sets to produce a new random game, and then trying to figure out what it is, can be a fun interaction loop. If you aren't a researcher interested in design spaces, however, the proportion of playable games remains too low for the mutation approach in *Cillr* to be ready for end-user consumption.

Besides producing Gamika chromosomes (both manually and with randomisation), *Cillr* includes editing tools for graphical elements such as sprites, level layout, and lighting, so complete games can be produced, including games with level progressions and multiple levels of difficulty. We have used the interface to produce clones of classic games like frogger, asteroids and space invaders, as well as a variety of novel casual games (a narrated set of design sessions is reported in [1]).

We do not, of course, claim that an on-device mobile game design tool with 284 sliders and parameter randomisation is the solution to the problem of democratising game design. But as an initial baseline, *Cillr* is perfectly usable, at least by experts. Its main drawback is that it is complicated to navigate, and requires some time hunting for which slider to change to make something specific happen. Furthermore, even after having found the desired parameter, it can be difficult to understand why the game didn’t change as expected.

In a preliminary user test with game-design undergraduate students, we found them somewhat frustrated by the experience of using *Cillr* to make games. Interface complexity was one issue, but more importantly, the difficulty of understanding the high-dimensional design space made it hard for these initial testers to grasp what they wanted to do in the app, and how they would begin to do it. Therefore, rather than focus initially on improving *Cillr’s* interface, we have instead focused on producing design tools for more cohesive, lower-dimensional design subspaces, still on top of the overall Gamika Technology platform, but not exposing the entire design space at once.

Carving out cohesive subspaces

The next phase of our research has looked at restricting the larger Gamika design space to more cohesive subspaces, which exposes more comprehensible on-device design spaces by specialising interfaces and automating generative aspects to navigate the subspace.

Despite all games being 2D and physics-based, the Gamika space is heterogeneous, with very different kinds of games available within its parameters; some puzzle-like, others meditative, others arcade-style action, etc. Cohesive subspaces share enough features such that navigating the design space feels more akin to designing game levels, or game variants, with more understandable relationships between parameter changes and changes in gameplay behaviour (though often still with complex and emergent aspects).

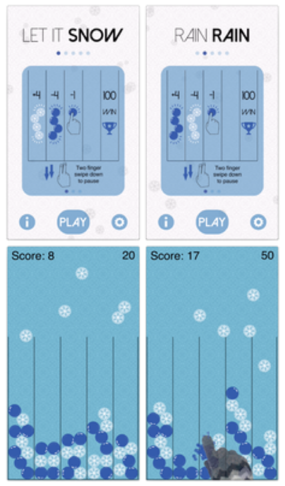
Figure 3: *Wevva* rules (top) and gameplay (bottom)

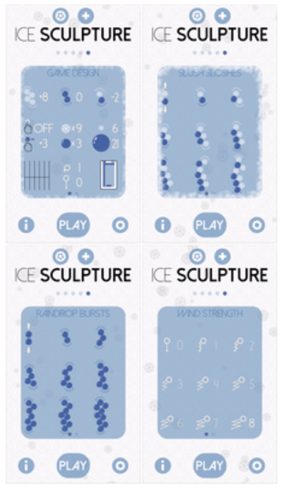
Figure 4: *Wevva* design panels

Once we've identified a design subspace, the research question then becomes: given this design subspace, can we understand its space of variation well enough to build user-interface and generative components that match with its salient features, and employ those to build an enjoyable, mixed-initiative app for designing (and playing) games or levels in that subspace?

Below, we describe the first two subspaces we've investigated, and the corresponding mobile game-design apps, namely *Wevva* and *No Second Chance*.

Wevva

Using *Cillr*, we made a relatively addictive four-in-a-row game called *Let It Snow*, where snow and rain pour down from the top of the screen (as white and blue balls respectively). When four or more white balls cluster together, they explode and the player gains a point for each in the cluster. Each white ball that explodes is replaced by a new one spawned at the top, with a maximum of 20 on screen at any one time. Likewise with blue balls, except the player loses points for them. Players can interact with the game by tapping blue balls to explode them, losing one point in doing so.

While the game rules are straightforward, we have found it to be difficult and require puzzle-solving strategies as well as quick reactions. There is a grid structure which collates the balls into bins, and the best way to play the game involves trapping the blue balls in groups of twos and threes at the bottom, while the whites are exposed and are continually refreshed through cluster explosions. Occasionally, when all blues are trapped in small clusters, only whites will spawn, which is akin to snowing (hence the game's name) and is a particularly pleasing moment to aim for.

We used *Cillr* to produce three variations of *Let it Snow* called *Rain Rain*, *Jack Frost* and *Slush Slosh*, each requiring different tactics and skills. These winter games will be paired with games representing additional seasons, for release as an iOS game, *Wevva* (Fig. 3). This app further includes two aspects that are not common in casual games: (a) an AI player for each game that can assist novice players, and (b) a design screen enabling players to generate levels in a semi-random way, and tweak them to get balanced variations. The AI player appears on-screen as a gloved hand that taps the blue balls to keep clusters of four from forming (Fig. 3, bottom right), implementing one part of a winning strategy. A slider lets the player change the level of AI assistance. At 50%, it feels like having an in-game partner helping out. At 100%, the game is quite different, as the AI player takes care of one aspect of the game (avoiding losing points), freeing the player to concentrate purely on gaining points.

The design screen (Fig. 4) exposes the following elements of the game design to the player: (a) the sizes at which clusters of balls explode (b) the scores attached to clusters exploding and the player tapping (c) the size of the balls (d) the maximum number of balls of each type allowed (e) the design of the grid, (f) physical properties of the environment, namely bounciness and noise, (g) spawning regions for both types of balls, and (h) what happens when the player taps the balls – both actions and scoring consequences.

There is a random generation button which will set these parameters in a varied way, but designed so that the clustering explodes are balanced in terms of expected score. We achieved this by running online simulations of novice players and recording the number of times

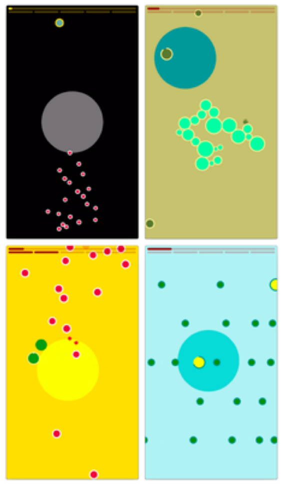

Figure 5: *No Second Chance*

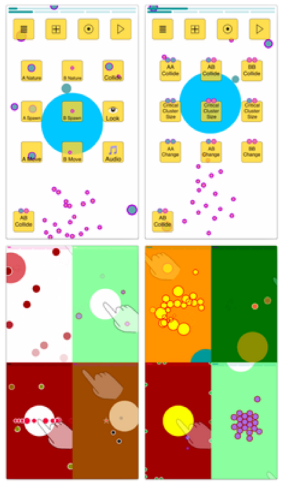

Figure 6: Design interface (top) and auto-playtester (bottom)

that clusters of each size and type occurred. Initial experiments with the design screen have indicated that the space exposed by the above parameters, while vast, does not contain hugely varied game types. However, we have used it to make games which differ substantially from the four preset games, e.g., involving juggling balls, or trapping and tapping them, etc.

No Second Chance

Again using *Cillr*, we designed a game of patience and concentration, *Pendulands*. Here, balls move in a pendulum-type motion and annihilate each other if they collide; the player must catch five of them by hovering under them with a large round target until they stick.

By varying parameters within this theme, we discovered that a whole set of *Pendulands* variants (or levels) can be created. The fixed elements defining this subspace of Gamika games are: the player always controls the target by dragging, and must catch five balls on the target. Within these parameters, very different types of challenges can be created (see Fig. 5 for examples).

No Second Chance is our third app, built around this space of games. The name comes from a meta-game mechanic: players can send games to each other in such a way that they are deleted if the receiver doesn't beat the game on first playing (in five minutes). This emphasises the "disposable" nature of games in a generative space, where part of the challenge is exploring the space of games and figuring out how each one works when first encountering it.

As with *Wevva*, a design screen (Fig. 6, top) lets players make new *No Second Chance* games. It is laid out as a hierarchical menu, with submenus allowing visual style and a variety of physics parameters to be changed. Since what is fixed about *No Second Chance* games is the control and scoring mechanism, new games are made by varying physics, spawning and scoring options, which can produce very different game dynamics and mechanics.

To demonstrate the types of games that can be produced (and to provide an initial challenge), the app comes with 100 games we designed using this interface, which we've categorised into three primary types of challenges: *skill* games, where the primary challenge is dexterity, *ingenuity* games, where the primary challenge is figuring out a specific trick or strategy, and *patience* games, which involve waiting for the right situation to arise and capitalising on it accordingly.

The generation button creates a new game via an evolutionary process. In particular, pairs of existing games' chromosomes are randomly crossed over, then filtered using static heuristics to reject clearly bad candidates. The first four candidates that pass the filter are auto-playtested on the device in a split-screen view (Fig. 6, bottom) that plays them at 8x speed for 5 seconds, the equivalent of 40 seconds of game time. We want games to be playable but not too easy, so the app chooses the game that the playtester was able to catch the most balls on, without being able to catch all five. (This split-screen visualisation of playtesting isn't strictly necessary, but we are exploring the entertainment value of "Hollywood AI" that visually externalizes to users what the apps' AI components are doing.)

We have been conducting playtests with the beta app to improve both the design interface and generator. A series of playtests in a local school (Camborne Science and International Academy) have been particularly helpful, as the students have proven adept at mastering the app and providing useful suggestions. Taking a more explicitly learning-technology turn, since game design in *No Second Chance* is essentially modification of physics parameters to produce new types of gameplay dynamics, we have also written a series of companion lessons that introduce game design and basic physics through *No Second Chance* design exercises.

Conclusions and future work

Our goal is to democratise mobile game design by building on-device design tools, so players can design new games in the same setting in which they play them. Our view is that doing so requires building tools for mixed-initiative navigation of design spaces, where player/designers have control over their designs but also enjoy the benefits of automated and semi-automated exploration of these design spaces.

To summarise our design strategy: (a) The Gamika Technology platform parameterises game design so that it becomes navigation of design possibilities rather than programming (b) we carve out coherent design-space subsets in which the relationship of parameter changes and game design is more intuitive, and (c) we build mixed-initiative apps mapping design interfaces, automated game playtesters, and game generators onto each subspace. Our first two apps built on such subsets, *Wevva* and *No Second Chance*, are discussed in this paper, as is *Cillr*, an internal prototype app that targets the entire Gamika design space.

Modular, reusable automated playtesters are a component worth elaborating on. They take initiative in varying ways: as evaluators during game generation (in *No Second Chance*), as AI assistant players (in *Wevva*), as fixers for broken player-designed games [4], and even as performers in standalone art installations in which the autoplayer both designs and plays new games (as per the installation called *I Create, You Destroy* shown at the first Arts as Games/Games As Arts festival). Since the auto-playtesters serve multiple roles, and often need to play in a way that is *readable* by users, we currently handcraft them out of modular heuristics in each domain, rather than using off-the-shelf but black-box general game playing algorithms such as MCTS or deep learning. We are interested, however, in automatically inferring these kinds of modular, readable heuristics, along the lines of [3].

Acknowledgements

This work is funded by EC FP7 grant 621403 (ERA Chair: Games Research Opportunities). We are grateful for the feedback provided by our alpha/beta testers.

References

1. Simon Colton, Mark J. Nelson, Rob Saunders, Edward J. Powley, Swen Gaudl, Michael Cook. 2016. Towards a computational reading of emergence in experimental game design. In *Proc. CCGW 2016*.
2. Kate Compton and Michael Mateas. 2015. Casual creators. In *Proc. ICCC 2015*, 228-235.
3. Fernando de Mesentier Silva, Aaron Isaksen, Julian Togelius, Andy Nealen. 2016. Generating heuristics for novice players. In *Proc. CIG 2016*, 158-165.
4. Edward J. Powley, Simon Colton, Swen Gaudl, Rob Saunders, Mark J. Nelson. 2016. Semi-automated level design via auto-playtesting for handheld casual game creation. In *Proc. CIG 2016*, 372-379.
5. Georgios N. Yannakakis, Antonios Liapis, Constantine Alexopoulos. 2014. Mixed-initiative co-creativity. In *Proc. FDG 2014.*