The Q*bird Level Designer: User-assisted procedural level design in augmented reality

Yi Fei Cheng
Swarthmore College

Aline Normoyle
Swarthmore College

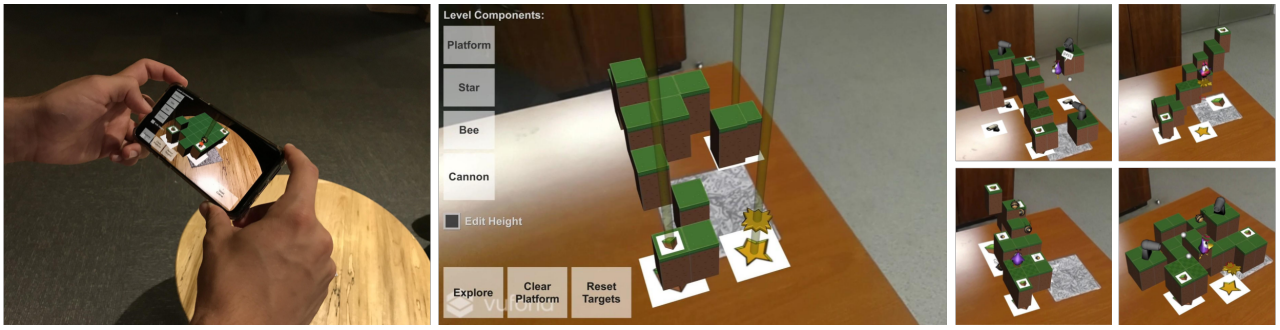

Figure 1: The Q*bird Level Designer. Left to right: Using the designer on a phone. Screenshot of the design tool. The buttons on the left serve as a pallet for available cards. The yellow handles show the locations of generated content on top of cards. Screenshots of different levels created with our system.

ABSTRACT

Augmented reality (AR) gaming is becoming widely available thanks to improvements in hand-held devices such as phones and tablets. In this work, we describe our system for generating levels for the AR game, Q*bird. In Q*bird, the player must visit every cell in the level while avoiding bees and cannon balls, similarly to the 1982 arcade game, Q*bert. To create a new level, designers place game elements using virtual cards. The system then generates the remainder of the level, ensuring that it's navigable. Designers can edit these levels by dragging and dropping the created geometry. To test, the designer can drop a character into the level and play it. This system aids playtesting and level design by allowing levels to be quickly specified and tested in the same environment in which the game is played. Furthermore, this system offers an example of how the design of AR levels can also be performed in AR.

CCS CONCEPTS

• **Computing methodologies** → **Mixed / augmented reality**; • **Human-centered computing** → *User interface design.*

KEYWORDS

augmented reality, procedural content generation, level design, user-assisted design tools

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MIG '19, October 28–30, 2019, Newcastle upon Tyne, United Kingdom
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6994-7/19/10.
https://doi.org/10.1145/3359566.3364686

ACM Reference Format:
Yi Fei Cheng and Aline Normoyle. 2019. The Q*bird Level Designer: User-assisted procedural level design in augmented reality. In *Motion, Interaction and Games (MIG '19), October 28–30, 2019, Newcastle upon Tyne, United Kingdom.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3359566.3364686

1 INTRODUCTION

Augmented reality (AR) gaming is becoming widely available thanks to improvements in hand-held devices such as phones and tablets. In this work, we describe our system for generating levels for the AR game, Q*bird (Figure 1). The goal of this system is to make the process of generating and evaluating AR levels easier and faster. Level design is an iterative process, where the quality and enjoyment of a level can only be assessed by playtesting. Thus, level design is time-consuming. Playtesters must repeatedly play through a level to identify problem areas. Furthermore, AR games are typically created using a desktop development environment. Thus, the environment where the level was designed may not match the hardware or environment where it is played. This system offers an example of how the design of AR levels can also be performed in AR.

Q*bird is inspired by the 1982 arcade game Q*bert. In Q*bert, the player must visit every block in a pyramid while avoiding enemies. While the original Q*bert levels were a pyramid of stacked blocks, Q*bird can have an arbitrary terrain of blocks. Q*bird supports Q*bert-inspired enemies – a bee and a cannon – which can push the player off the platform. To help players escape enemies if cornered, single-use hover disks can be used to escape. The player wins by visiting every block and loses by falling off the level.

To create a new level from scratch, designers drag and drop virtual cards representing game elements onto a surface, such as a table. Our system then generates a navigable level based on the given cards. This approach is inspired by the use of paper for prototyping game designs [Schell 2014]. Our system supports cards for

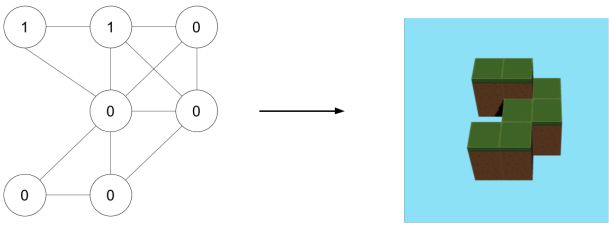

Figure 2: Level representation. The navigable space is represented as a graph (left). Each node has a height associated with it and maps to a cell (row, column) in a 2D grid.

representing waypoints, bees, cannons, and hover disks. Typically, many different levels are valid for a set of cards. To help designers explore different possibilities, the system offers an *explore mode* which generates a series of visually different options. For small edits to an existing level, the system offers a *tweak mode*. In tweak mode, the designer can move cards and change block heights by clicking and dragging them. The relationship between the virtual objects, the real world, and the game are one-to-one. No special camera controls are needed: the user walks around the object to see it from the other side as if looking at a hidden objects through a window. Once happy with a configuration, the designer can enter *play mode* to drop a character and test the game mechanics. Generating a level for playtesting can be completed in seconds.

Exploration and tweaking are implemented using an evolutionary algorithm to search the space of possible levels, inspired by [Shaker et al. 2010]. Evolutionary algorithms are a popular technique in procedural content generation (PCG) because they place few constraints on the representation of the level or the fitness objective [Shaker et al. 2016]. Thus, they allow us to incorporate the unique requirements of Q*bird into the generative process . Furthermore, the stochastic nature of procedural algorithms can produce innovative examples which the designer may not have previously considered. Lastly, a human can influence the output generated by a PCG algorithm [Secretan et al. 2011]. In our system, when users edit the level directly through tweaking, the changes are propagated forward into future searches.

2 EXPLORING AND TWEAKING LEVELS

The main goal of the level designer is to help a user search the large space of possible levels given a placement of cards. Levels are represented as graphs superimposed on a 2D grid of cells (Figure 2). The system maintains two representations of the level. One is a genome representation, which is used for evolutionary search. The other is a 2D array of heights, which represents the geometry (e.g. the phenotype in evolutionary search terminology). Given a genome, the system generates a level using an algorithm based on depth-first-search. Similarly, if the user edits the terrain's height, the system updates the genome so it stays consistent.

Users can iterate through a series of potential levels (*explore*), or edit a level directly (*tweak*). When exploring, we found that generating levels randomly can result in many similar-looking levels. Therefore, when exploring, we use evolutionary search to ensure the system shows visually distinct examples. We quantify

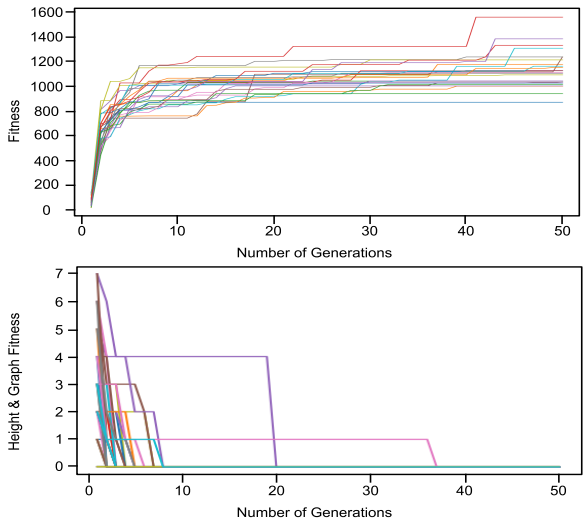

Figure 3: Convergence. Top, in explore mode, our system maximizes novelty. Bottom, in tweak mode, we allow users to move cards while minimizing the difference between the remaining parts of the level. Each line represents a computed example. The fitness (Y axis) of the population improves with the number of generations (X axis).

visual differences in terms of height changes, graph differences, and quantities of empty space. To tweak a level, the system supports moving cards and changing cell heights. When moving cards, we use evolutionary search to minimize the difference between the shared regions. When changing heights, we modify the heights on the 3D map directly and update the genome.

3 RESULTS

We implemented our system on a Microsoft Surface Pro (Intel Core i5, 2.50 GHz, 8 GB RAM) using the unity3D game engine and Vuforia. In Figure 3, we show the convergence of the fitness objective for explore mode (generating novel examples) and tweak mode (moving cards). In both, we generate 50 random levels of size 5x5 cells, initialized using 2 platform cards. Explore performs at 1 second at 25 cells (5x5) to 4 seconds at 100 cells (10x10). Moving cards performs at 1 second at 25 cells (5x5) to 5 seconds at 100 cells (10x10). Changing heights performs at less than 0.2 seconds at 25 cells (5x5) and less than 1.5 seconds at 100 (10x10). There is no relationship between the number of generations needed to converge and the size of the grid.

REFERENCES

Jesse Schell. 2014. *The Art of Game Design: A book of lenses.* AK Peters/CRC Press.

Jimmy Secretan, Nicholas Beato, David B D'Ambrosio, Adelein Rodriguez, Adam Campbell, Jeremiah T Folsom-Kovarik, and Kenneth O Stanley. 2011. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation* 19, 3 (2011), 373–403.

Noor Shaker, Julian Togelius, and Mark J Nelson. 2016. *Procedural content generation in games.* Springer.

Noor Shaker, Georgios Yannakakis, and Julian Togelius. 2010. Towards automatic personalized content generation for platform games. In *Sixth Artificial Intelligence and Interactive Digital Entertainment Conference.*