Mixed Reality Meets Procedural Content Generation in Video Games

Sasha Azad, Carl Saldanha, Cheng Hann Gan, and Mark O. Riedl
School of Interactive Computing; Georgia Institute of Technology
sasha.azad, csaldanha3, gan, riedl@gatech.edu

Abstract

The use of artificial intelligence and procedural content generation algorithms in mixed reality games is an unexplored space. We posit that these algorithms can enhance the gameplay experience in mixed reality games. We present two prototype games that use procedural content generation to design levels that make use of the affordances in the players physical environment. The levels produced can be tailored to a user, customizing gameplay difficulty and affecting how the player moves around the real-world environment.

Introduction

As the scale of games increases and video games are played by diverse players in more diverse environments, there has been a corresponding increase in the need for computational systems to replace the manual effort involved in generating gameplay assets and adaptation. Procedural Content Generation (PCG) is the use of algorithms to automate the production of various aspects of computer games, such as levels, missions, or rewards. Instead of game designers manually placing individual structures, enemies, or other elements in game environments; these elements and their relationships are encoded and used to generate a game automatically. PCG algorithms use these encodings to create multiple customized game elements. This approach has been used successfully in the past, with games such as Discoverie, Spelunky, Bioshock Infinite, and No Mans Sky using PCG to generate rooms, caves, cities, and planets (respectively) for the player to explore.

Recent hardware developments in Virtual Reality (VR) headsets, such as Facebooks Oculus Rift, Augmented Reality (AR) headsets, such as Google Glass or the Daqri Smart Helmet, and Mixed Reality (MR) headsets, such as Microsofts HoloLens and the Magic Leap, make it increasingly likely that commercial augmented and mixed reality games will become available in the near future. VR environments take the users visual experiences entirely into the digital world. These environments allow the designers complete control over the environment. In contrast, AR takes the users view of the real world and layers digital information on top of it. MR combines both, allowing digital and physical objects to coexist and interact with each other in real time. AR experiences are expected to adapt and change significantly based on the configuration of the players physical environment, and MR experiences additionally also have to adapt to the interaction between physical and virtual objects. An example of an MR game is Microsofts Young Conker game for HoloLens, which has the player controlling a virtual avatar walking across table and floor surfaces (Microsoft, Asobo Studios, 2016). By changing rooms or furniture arrangements, the user can participate in varying gameplay experiences. This evolution from virtual to mixed reality, or transmogrified reality, has been described as one of the biggest changes in the gaming industry in the last 30 years (Falstein, 2015).

Historically, procedural level generation has been researched in the context of fully virtual game environments (Shaker, et al. 2015). Thus, artificial intelligence approaches to PCG treat the creation of game content as an optimization problem. In mixed reality our algorithms are constrained by the presence of objects in the real world. Added virtual objects need to be presented with realistic integration into the physical world. Acceptable occlusion, object identification, and other relationships between the real and virtual objects must be made. We intend to apply existing procedural content generation techniques while keeping in mind these constraints of the domain.

In this paper, we explore two game concepts with the design constraints of an MR environment. Currently, as we do not have an MR device yet, we simulate two games in a virtual 3D environment on both a PC and an Oculus Rift. The games are being developed as prototypes which explore the ways in which procedural content generation can enhance MR games. The games are loosely inspired by the popular platformer games, Super Mario Bros. and Lemmings.

With Mixed Reality Mario, the player controls a virtual avatar that runs and jumps across real furniture, virtual platforms and on top of enemies on a procedurally generated track. The route of the track is affected by the heuristics of our PCG algorithm. With Mixed Reality Lemmings, we move away from the single linear track. Instead, we allow players to interact with their environments using virtual boxes. This forces our player to interact with the virtual and real surfaces in order to move the lemmings across the furniture. Our algorithm detects playable surfaces and uses the negative space to create a compelling game. To do this, we limit and direct player movement of the lemmings across surfaces with virtual walls. Both games were designed to be played in any real-world room-style environment. We intend to demonstrate how different arrangements and pieces of furniture in a room can impact the level being generated. We do this by having the underlying PCG algorithm take into consideration the quantity and position of surfaces and obstacles (i.e., non-playable surfaces) in the environment. Next, we look at the technical implementation of the prototypes as well as other potential evaluation functions that can be used for our MR playground.

Related Work

We present two games designed for a mixed reality environment. Our games utilize a real time generation of levels using a generate-and-test PCG technique for path selection. Togelius et al. (2011) enumerate many of the uses of procedural content generation in games and coined the term search-based PCG to distinguish a special class of procedural content generation problems that can be solved using generate-and-test methods such as genetic algorithms and simulated annealing.

Cook (2015) used a PCG algorithm and computer vision to evolve a level with constraints based on visibility of game elements in a virtual game environment. Tutenel et al. (2009) generated virtual levels based on a set of rules and the high-level semantics associated with objects in a game world. For example, one rule could be keep adding cupboards until the sum of all Storage Volume properties exceeds 1.3 cubic meters. The placement of objects in the scene was based on the virtual world presented to the algorithm. One of our games is based on the popular Super Mario Bros. game, and uses a similar set of rules for the placement of virtual enemies. There have been a number of research projects on generation of levels for Super Mario Bros. and similar platformers, including, but not limited to: Shaker et al. (2012) and Guzdial and Riedl (2015). To date, work on level generation in platformer games has assumed a virtual environment whereas mixed reality games would need to consider the additional constraints imposed by a physical environment.

Further work has been done on the positioning of elements in a 3D scene. Specifying angle constraints, Gosele and Stuerzlinger (1999) determined potential positions of virtual objects in the scene by specifying their relationships to each other. For example, a chair must be placed on the floor relative to a table already in the scene. Both works are useful for placing virtual assets that correctly interact with the real world in a mixed reality experience. However, unlike a virtual environment, this creates an opportunity for future algorithms to make use of data from the real elements while generating candidate solutions.

Level Generation Preliminaries

Level generation for both games occurs in two steps. First, we sense the environment the player is in. Next, we use a generate and test methodology to select the optimal route for the avatar to follow using a fitness heuristic.

Environment Mapping

During the initial set up of the game environment, we require the player to create a map of the environment by walking around the room using a Kinect 2.0 with Kinect Fusion. We use this to create a 3D model of the room. We then identify usable horizontal surfaces as a set of polygons. The polygons are clustered together using the Union-Find algorithm to create distinct concave hulls that act as playable surfaces. Each playable surface is represented as a node in a graph. Edges connect nodes whenever it is possible to travel between the surfaces. Movement between surfaces can be impeded by physical obstacles in the path; for instance, jumping from one sofa to another may be stopped by the back of the sofa in between. If the distance between two surfaces is greater than or equal to a threshold distance for the virtual character, this also indicates that it is impossible to travel between the surfaces without the help of a virtual platform or bridge. The distance between two surfaces is measured as the distance between the two closest points in the point cloud of each surface.

We currently register each of the surfaces being scanned during the room-mapping phase with an ID to help us in tracking the surfaces that have previously been visited by the player controlled avatar. Unlike mixed reality games that allow the player to control the avatars motion in 3D and choose the order to visit surfaces (such as in Microsofts Young Conker), our mixed reality level generator constrains the player to operating on a specified sequence of surfaces. By limiting and directing the possible paths the player can take, the algorithm can control the length of gameplay. In the future, this can be used to create rhythms (an important part of the platformer genre) (Smith et al. 2009) or challenge progressions (Shaker, Yannakakis, and Togelius 2010; Harrison and Roberts 2013, Zook and Riedl 2015). Since at times, the player may have access to only a limited number of playable surfaces (as identified by our algorithm) based on the environments furniture arrangements, we can allow for novel puzzles or challenges using the same surfaces.

Level Generation and Evaluation

There may be many routes across the surfaces of a room between any two points, some of which are easier than others. We rely on a search-based PCG approach using a generate-and-test method to test each route generated against our evaluation function to choose the best route. The procedural content generation algorithm has the responsibility for delivering the all the game elements to the player.

We use a combination of heuristic functions $h_{1\ldots n}$ that evaluate each of the possible paths produced, normalizing their result to a value between 0 and 1. We pass a weight control w_i $(0\ldots 1)$ to each one of the heuristics h_i, which tells the heuristic to favor paths which return values closer to the weight control. This is done using $v_i = 1 - abs(h_i - w_i)$. The value of each path is given by $\sum_{i=1}^{n} v_i$.

We randomize the weight controls $w_{i\ldots n}$, to produce different paths. describe the heuristic functions we used to tie

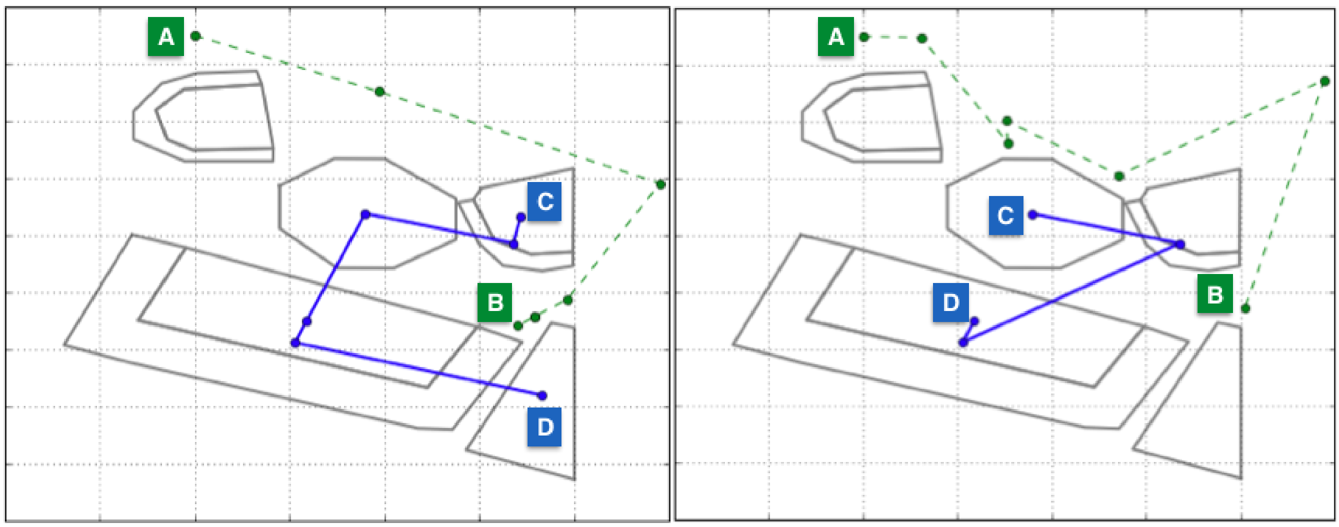

Figure 1: Two possible virtual character tracks (blue, A-B) and how they affect player movement profiles (green, C-D).

the selection of virtual content to the affordances of the real-world environment:

- Length of Gameplay (h_{length}): The length of the path is selected as a weight for the function. We choose a path closest to the weight passed to the function. We control the length of gameplay by generating longer or shorter paths according to the weight passed to the length heuristics. Longer paths produce a higher weight.
- Proportion of surfaces used ($h_{surfaces}$): The percentage of surfaces to be visited is selected as the weight. A percentage is used since the number of surfaces cannot be known a priori. Paths that go to more surfaces have a higher weight according to this heuristics.
- Player physical movement (h_{RRT}): We know the players' position in the virtual space from the position of the camera. The amount of movement required from the player is computed by finding the shortest route through negative space for the player to remain within reach of the lead agent (Figure 1). This is done by using the rapidly-exploring random tree algorithm to predict possible paths to the virtual agent. Longer paths are given a higher score and the weight control allows us to place the level based on the distance the player travels based on the RRT. Figure 1 shows two possible tracks using different furniture surfaces and some likely routes the player can take to minimize his or her distance from the virtual character while keeping it in view. The required physical movement of the player can be very important if the user has limited mobility or if physical fitness is a motivation.
- Target difficulty ($h_{difficulty}$): The difficulty heuristic is defined independently for each game we created. In MR Mario, the difficulty can be varied by changing the number or length of the jumps required on the path, or the number of enemies Mario can encounter. In MR Lemmings, difficulty can measured as the frequency of player interventions (number of virtual boxes or jump pads used) required. Target difficulty can be specified as a target number or as a function (e.g., monotonically increasing).

Path	$v_{RRT,w=0.2}$	$v_{length,w=0.8}$	$\sum v_i$
[7, 6, 5, 0, 2, 1]	0.8571	0.9888	1.8459
[0, 2, 6, 5]	0.7561	0.8404	1.5965
[2, 6, 7]	0.4064	0.8921	1.2985
Path	$v_{RRT,w=0.5}$	$v_{length,w=0.5}$	$\sum v_i$
[0, 2, 6, 5]	0.9439	0.9999	1.9438
[2, 6, 7]	0.7064	0.9886	1.695
[7, 6, 5, 0, 2, 1]	0.5571	0.8749	1.432

Table 1: Change in the values of the normalized heuristic with the weights chosen and the impact on the selected path

In Figure 1 we show two possible tracks created by changing the weights of the heuristics. The normalized heuristics generating the paths have been detailed in Table 1 above. The path value in the table contains a list of the detected surface indices traversed by the virtual character. In the left image a $w = 0.8$ for the path heuristic allows for a longer generated virtual character path (i.e. $[7, 6, 5, 0, 2, 1]$). while minimizing the human movement since it has a lower heuristic weight (i.e. $w = 0.2$). We see that the RRT predicted the movement of the human player from their starting location of point A to point B (in the left image) since all surfaces traversed by the virtual character can be easily accessible from point B.

Game Specific Level Generation

Mixed Reality Mario

Mixed Reality Mario is a mixed reality platformer loosely inspired by the popular Super Mario Bros. game. Prior PCG work on this game has assumed that the generator has full control over all aspects of the level. In mixed reality, the

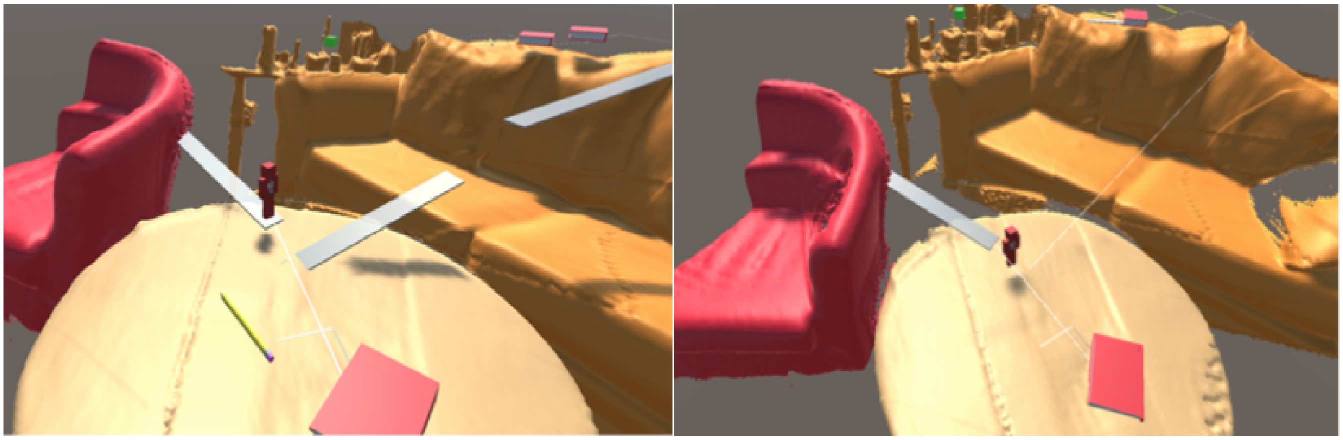

Figure 2: Two possible MR Mario tracks given the same furniture configuration. Note for clarity the image is shown as rendered in a VR headset.

algorithm is constrained by the configuration of furniture, walls, and interactions with the physical environment. By generating a single linear track that crosses physical furniture surfaces (the floor is lava), the algorithm can control the players exploration of the environment for customization of the gameplay experience while responding to the selected evaluation function.

We use the aforementioned level generation algorithm with a given start and finish point. Varying the difficulty parameter in the heuristic affects the number of enemies placed on the path and the length of each level. Surfaces are connected with virtual platforms that the player can walk on to traverse. Virtual platforms are generated between surfaces that have been calculated to be more than the maximum jumpable distance for the Mario character by our surface mapping algorithm.

In the typical Mario style, enemies are generated on long straight paths for the player to jump over or kill. Our PCG algorithm is able to place enemies on the path precisely using a set of rules that define the movement of these enemies on the surfaces. For instance, a static paper tack can be placed on shorter paths and sharp turns; in contrast, a pencil follows the virtual character along longer straight paths in order to stab the character but takes a longer time to turn corners.

In MR Mario, the player must control and move the virtual character from one side of the room to the other following a track across the furniture. We posit that (curiosity notwithstanding), left to themselves, players are likely to continually choose paths that are either too difficult or too easy to move through the room. Choosing paths that are challenging beyond the players current ability can lead to frustration, whilst the easier paths can lead to boredom. To increase the duration of enjoyment and to motivate the player to more deeply explore the mixed reality playground, we constrain the player with a single track. The single track allows us to tightly control the the track, even along the edges and backs of furniture, allowing for adaptation to the players level of difficulty. The player is removed from the responsibility of planning the avatar's path ahead while navigating the real world obstacles in the players path. Instead, he can dedicate all his skill to successfully interacting with the system to deal with the virtual obstacles and enemies the system generates. With future play testing, we plan to compare the discrete elements of the players movement (for instance, running or jumping) to refine our heuristic algorithm to maximize the players enjoyment in the game.

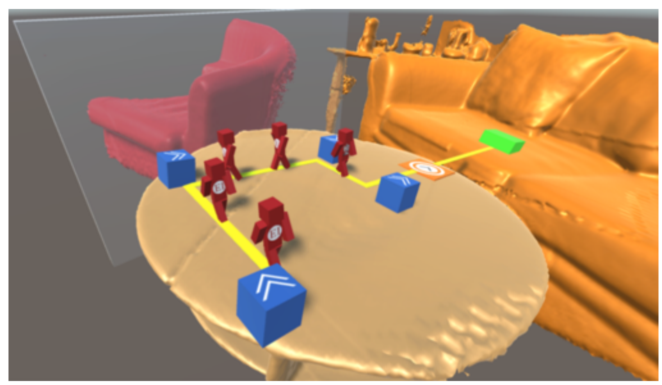

Figure 3: *A Possible MR Lemmings path generated with virtual wall constraining jumps. The path the agents will take (normally hidden from the player) is shown in yellow.*

Mixed Reality Lemmings

Our second game is a mixed reality puzzle game inspired by the popular Lemmings game. The game spawns a line of virtual avatars known as lemmings. These lemmings must be directed by the player safely to the final goal. The lemmings walk in a straight line until they fall off a surface to their deaths, unless the player intervenes. Virtual walls are generated around the world to restrict the Lemmings movement between surfaces by causing the lemmings immediate death on collision. The player can interact with the lemmings in two different ways. First, the player can place a virtual box which directs them left, right or backwards. He does this by clicking on a point in the virtual space which places the box. Again by clicking a point in the virtual space jump pads are generated that launch lemmings over gaps. Different jump pads can launch lemmings different distances and heights.

The heuristics are used to generate an open-world puzzle for the level. Every pair of surfaces is considered as a start and goal. For every pair, we enumerate all possible routes through the graph. The solutions for all pairs of start and goal nodes are then ranked according to the evaluation function described above. The procedural content generation algorithm also creates virtual walls (Fig. 3) to constrain and direct the player to certain routes. The player can move through these walls, but the lemmings cannot. Using the walls, we can control the type of experience that the player has in the game by restricting the possible solution paths for the lemmings.

The walls are generated by traversing the nodes of the winning route and placing a wall between any two surfaces on a sub-optimal path (as defined by the game heuristics). For instance, if the level designer is searching for a route to maximize human movement, walls could be generated to force the human to weave through furniture instead of allowing direct Pythagorean movements or shortcuts.

In contrast to MR Mario, MR Lemmings constrains the player to specific surface sequences but allows the lemmings to have continuous movement within the surfaces themselves. By not limiting the motion of the virtual characters to a single linear track, we allow the user to experiment with the placement of virtual boxes and jump pads to explore the room more freely. However, constraining the surfaces ensures that each level can vary and not be repetitious while allowing the levels to become more challenging over time. This difference in the granularity of the route constraints in both games allows for maximum reusability of the playable surfaces within the real-world environment of the player while keeping the game challenging and maintaining player interest.

Conclusions and Future Work

In the future, we plan to continue working on player interaction with the mixed reality objects and evaluate the gameplay experience. The player data we will acquire on level completion and the avatar death rates can then be used to model the level difficulty and challenges faced by the player. Additionally, we plan to integrate a neural net to identify the type of room the player is in. This will allow us to generate room-specific interactions and objects. For instance, in the MR Mario game, enemies in a kitchen could be virtual knives, while a power-up in a living room could be a virtual book on a table.

Currently we are able to simulate the environment virtually on a computer running a Unity Environment. In the future we plan to integrate a mixed reality device into the scene in order for the player to be able to interact more naturally with the environment, place boxes or direct the virtual characters to move.

References

Cook, M., 2015, September. Would You Look at That! Vision-Driven Procedural Level Design. *Proceedings of the 2015 AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.*

Falstein, N., 2015. Transmogrified Reality. *Session delivered at the Smartphone & Tablet Games Summit, Game Developer Conference, San Francisco, CA.*

Gosele, M. and Stuerzlinger, W., 1999. Semantic constraints for scene manipulation. *In Proceedings of the 1999 Spring Conference in Computer Graphics.*

Guzdial, M. and Riedl, M.O., 2015. Game Level Generation from Gameplay Videos. *Proceedings of the 2016 AAAI Conference on Artificial Intelligence for Interactive Digital Entertainment.*

Harrison, B. and Roberts, D.L., 2013. Analytics-driven dynamic game adaption for player retention in Scrabble. *Proceedings of the 2013 Conference on Computational Intelligence in Games.*

Microsoft, Asobo Studio, "Microsoft HoloLens - Young Conker." 29 Feb. 2016. Web. 31 May 2016.

Shaker, N., Liapis, A., Togelius, J., Lopes, R. and Bidara, R., 2015. Constructive generation methods for dungeons and levels. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*

Shaker, N., Togelius, J., Yannakakis, G.N., Weber, B., Shimizu, T., Hashiyama, T., Sorenson, N., Pasquier, P., Mawhorter, P., Takahashi, G. and Smith, G., 2011. *The 2010 Mario AI championship: Level generation track. IEEE Transactions on Computational Intelligence and AI in Games, 3(4), pp.332-347.*

Shaker, N., Yannakakis, G., Togelius, J., Nicolau, M., and O'Neill, M., 2012. Evolving Personalized Content for Super Mario Bros Using Grammatical Evolution. *Proceedings of the 2012 AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.*

Smith, G. and Mateas, M., 2010. Tanagra: A Mixed-Initiative Level Design Tool. *In Proceedings of the 2010 International Conference on the Foundations of Digital Games.*

Togelius, J., Yannakakis, G.N., Stanley, K.O. and Browne, C., 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games, 3(3), pp.172-186.*

Tutenel, T., Smelik, R.M., Bidarra, R. and de Kraker, K.J., 2009, Using Semantics to Improve the Design of Game Worlds. *In Proceedings of the 2009 AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.*

Zook, A. and Riedl, M.O., 2012. A Temporal Data-Driven Player Model for Dynamic Difficulty Adjustment. *Proceedings of the 2012 AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.*