International Conference on Artificial Reality and Telexistence
Eurographics Symposium on Virtual Environments (2017)
T. Huang (Editor)

Holo Worlds Infinite: Procedural Spatial Aware AR Content

Louise M Lawrence[1], Jonathon Derek Hart[1], Mark Billinghurst[1]

[1]Empathic Computing Laboratory, School of ITMS, University of South Australia, Australia

Abstract

We developed an Augmented Reality (AR) application that procedurally generates content which is programmatically placed on the floor. It uses its awareness of its spatial surroundings to generate and place virtual content. We created a prototype that can be used as the basis of a city simulation game that can be played on the floor of any room space, but the approach could also be used for many other applications.

CCS Concepts

***•Human-centered computing** $\rightarrow$ **Mixed / augmented reality; •Computing methodologies** $\rightarrow$ **Mixed / augmented reality; •Information systems** $\rightarrow$ Multimedia content creation;*

1. Introduction

In this paper we describe Holo Worlds Infinite, a project that utilizes a method which procedurally generates virtual content for Augmented Reality (AR) scenes. This addresses two important problems for AR applications; (1) rapidly generating AR content, and (2) generating AR content that dynamically fills the available real space. Procedurally generated content for virtual worlds is not new, and algorithms which generate virtual content have been shown to reduce time and cost in creating content. However, procedurally generating content for AR is a challenge, because AR involves superimposing virtual objects over the real world.

We address the challenge of procedurally generating content that fits around any existing environment, without relying heavily on user input. The goal was to use a hardware device that scans any room that the user is in and then starts creating and placing virtual objects in the real environment, without users manually placing objects.

To achieve this, there are two aspects to the problem that were solved. One was procedurally creating content based on the surrounding space. This was done in the form of a procedurally generated road system for a virtual city. The second aspect was the placement of virtual objects, by filling the spaces between the roads with houses. Although demonstrated with a virtual city, our approach could be adapted to many other types of application content.

Our main contribution is combining procedural generation with spatial awareness of the surrounding space. We adapted existing procedural generation algorithms to work with depth sensed room scans, and developed a prototype of a system that turns the floor of any room into a game space. Figure 1 shows our procedurally generated city fitting into a room scanned by a Microsoft HoloLens.

Procedurally generated content for AR is uncommon, with an exception being level generators [ASGR16], which rendered everything with VR equipment instead of using a mixed reality device or AR device. This is similar to a couple of VR applications that attempt to use the real world environment to shape virtual game worlds [SX16] [SGJSM16]. Instead of replacing the real world with a rendered reconstruction, either partial or complete, our prototype uses an AR headset, the HoloLens, to view the generated content overlaying the actual environment, as demonstrated in Figure 1.

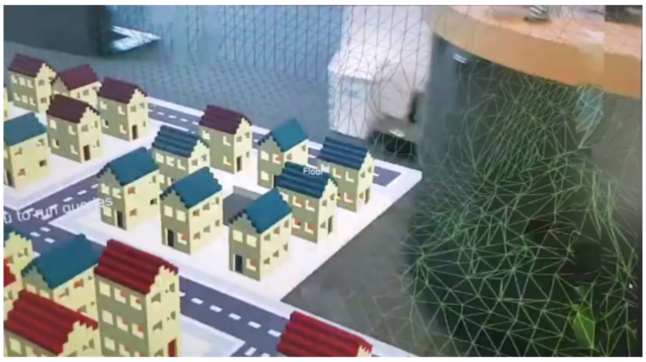

Figure 1: *Screenshot of the prototype showing the procedurally generated content overlaying the real world.*

2. Prototype

To demonstrate how the procedural AR content generation worked, we built a simple virtual city simulation that could be viewed on the HoloLens. To do this, we used the Unity game engine and Microsoft's HoloToolkit Unity plugin.

DOI: 10.2312/egve.20171373

We utilized the spatial capabilities of the HoloLens by using the Spatial Understanding feature of the HoloToolkit. We created a method that takes x and y values of a real location and a tile size value and determines if a tile can be placed at that location.

The procedural generation has two parts, road generation and building placement. The road generation uses a L-system [PM01]. The building generation uses the Unity engine's inbuilt Perlin noise function and a location to decide which building model belongs where. Any 2D square-grid based road generation system should work with our system. The main procedural algorithm uses a while loop that will look at a potential road and then decide whether it meets local constraints. The pseudo-code for the road generation is listed below:

```
Let P = PriorityQueue<Road>()
//Accepted roads meeting local constraints
Let S = List<Road>()
P.EnqueueAll(StartingRoads)
While (P.Count > 0)
    Let R = P.DequeueSmallest()
    Let A = CheckLocalConstraints(R)
    If (A==false) continue
    S.Add(R)
    P.EnqueueAll(GlobalGoals(R))
Return S
```

The steps are as follows:

1) We have six starting roads, two roads that span horizontally and vertically across the extents of the scanned area from where the player initially stands, and four roads that span around the edge of the scanned area.

2) We generate the rest of the roads using the while loop. The local constraints simply return true whether the road is at a minimum parallel distance to all other roads. The global goals method generates potential roads at random sizes all around the newly placed road like an L-shape. This produces a road network from which we remove any road tiles which are is not feasible.

3) Four Booleans determine which road tile model should be displayed at a given grid location. These represent if there are tiles north, south, east or west of the current tiles. The compass directions are set based on the tile's orientation within the Unity coordinate system.

Once the roads were generated, buildings were placed adjacent to them, using simple checks to see if tiles were unoccupied and if they were adjacent to roads. We kept track of the state of each tile by a set of maps (2D arrays of Booleans), one for road and another for buildings. Selecting a building model was achieved by a 2D Perlin noise function, which was part of the default Unity engine. The Perlin noise would determine the index of the building selected from an array of all the building models.

3. Conclusion

We have successfully combined AR procedural generation with the Microsoft HoloLens' depth sensing capabilities. The procedural generation algorithms, using L-systems and Perlin noise, were modified to be aware of the surrounding environments. The main benefit of our approach is that it allows procedural real-time creation of AR content that can be adapted to fit any real world environment. Our prototype has not been optimized, yet runs at an acceptable rate on a mobile, wearable computer, the HoloLens. Although we demonstrated our application with a virtual city simulation, it could be used for generating a wide range of AR content, such as virtual landscapes, forests, carpets, virtual tiles or wallpapers.

We intend to make Holo Worlds Infinite into a city simulation game that can be played on multiple HoloLens. We also plan to generate content on walls and ceilings as well as on floors which is done currently. Our future plans include further optimizations and building a small library of helper methods that supports tile based games. This would make procedurally placing virtual objects easier. The prototype itself needs improved collision detection, as the current queries are not checking that there is enough height above the space for the tiles with building models. In addition, it is possible to develop the prototype to take into account walls, ceilings and furniture. We also intend to continue developing the prototype into a simple city building simulation game. This will include a more sophisticated user interface which would allow the user to interact with the generated content. We plan to create more models and rules for the prototype to generate. One of our plans was to develop the final game to infinitely generate content, and infinitely scan the rooms around it. Whether this is possible remains to be seen, but our present prototype has established the groundwork for future research in this space.

4. Acknowledgements

We would like to acknowledge the Australian Commonwealth for financial support for our research, which made this project possible. We would also like to thank the Empathic Computing Lab at the University of South Australia for providing hardware and moral support.

References

[ASGR16] Azad S., Saldanha C., Gan C., Riedl M. O.: Mixed reality meets procedural content generation in video games. *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference* (2016), 22–26. 1

[PM01] Parish Y., Müller P.: Procedural modeling of cities. *28th annual conference on Computer graphics and interactive techniques* (August 2001), 301–308. 2

[SGJSM16] Sra M., Garrido-Jurado S., Schmandt C., Maes P.: Procedurally generated virtual reality from 3d reconstructed physical space. *Proceedings of the 22nd ACM Conference on VRST* (2016), 191–200. 1

[SX16] Sing K. H., Xie W.: Garden: A mixed reality experience combining virtual reality and 3d reconstruction. *Proceedings of CHI 2016 CHI Extended Abstracts* (2016), 180–183. 1