George Hanks P17173176

Procedural Level Generation and VR - Literature Review

Procedural generation is techniques within computer science used to create data algorithmically, it is commonly used to make textures and 3D models. Procedural generation has a long history with computer games, a popular example of its use is the game Minecraft (Mojang 2009), in which a world much larger that the earth is generated and stored in a file that is only several megabytes.

(Greuter et al., 2003) introduces a method for creating potentially infinite cities in real time, the goal of this project was to be able to generate visually interesting and complex environments, and the process should be self-contained. They use seeded random number generation to decide a building's parameters, the floor plans of these buildings are created in an iterative process from randomly generated polygons. This method removes lots of control from the designer and can lead to technically different but similar feeling outcomes, this feeling is emphasised by the fact that they only generate one type of building. They conclude that such a programme is ideal for large expansive environments, including the entertainment industry, but quality can still be strengthened.

(Greuter et al., 2003) has developed a method for developing large virtual worlds using procedural generation. They emphasize the need for these worlds to be generated at real time so that they can be used for video games and the entertainment industry. They use the example of large high-rise city to demonstrate the framework they have developed. They use an iterative process to design the floor plans of the buildings and then extrude these floor plans to create the facades of a building. They make use of view frustum culling to increase performance and store the building data within a cache. They conclude by saying that the worlds they can generate can be extremely large and do not make use of external data stored on the disc, and that the worlds they create are complex and diverse.

(Tutenel et al., 2009) presents a method of generating interiors for buildings within large urban environments. They state this would improve engagement in the games and contribute to opportunities for gameplay. They highlight the need for these interiors to be almost indistinguishable from hand-made counterparts, and the designs must be varied and look lived in. To do that, they use a rule-based design solution method where the software can identify all potential furniture positions within the space based on a set of guidelines, e.g. "The couch must be facing the television and within 5 metres." They conclude by saying that the need for methods like this is increasing with the size of game worlds and expecting artists and designers to make all the content would be unrealistic. This method allows the authoring of material on a large scale, but also leaves the control in the hands of the developers who can identify the rules and items in a space.

(Lopes et al., 2010) proposes a solution for generating interior floor plans, they justify the need for such a system as current solutions limit the power the designers and artist have over the outcome, and the floor plans that are generated generally do not adhere to consistency constraints. They make use of a grid-based layout to represent the rooms of a building, like architectural drawings, these rooms can then expand and shrink based on a set of rules and constraints. They summarise by saying that there proposed method delivers valid and plausible floor plans, that the designer has control over, and that the technique could easily be implemented into a larger system for generating urban environments.

(Smelik et al., 2010) presents a method of incorporating procedural generation techniques with designer authored content. They call this method declarative modelling and they use it to generate 3D worlds. The process starts with the designer describing a 2D worlds using sketches, these sketches could be road layouts, rivers, or elevation maps. A 3D world is then generation suing these maps, this world can then be edited afterwards by the designer to refine it. They emphasize this approach as it fixes some of the problems with other procedural generation techniques, and it can be a much faster approach compared to hand made alternatives. They summarise that getting procedural generation into the mainstream is still a hurdle that needs to be addressed however this approach is a step in the right direction.

(Cardamone et al., 2011) delivers an approach for generating racing tracks with player feedback and evolutionary methods. They use search-based procedural generation to create their racetracks, allowing them to generate racetracks based on a set of parameters with certain features and characteristics. They demonstrate that such approaches will increase the speed that developers can generate and validate content, creating a large amount of diverse content that remains enjoyable and increasing the game's replay value. They summarise by saying that on this system they have received positive feedback from users, but it still needs improvements.

(Togelius et al., 2011) presents their findings in the applications of evolutionary and metaheuristic search algorithms for procedural content generation. They justify the need for a survey as the demand from players increases along with cost for artists and designers, producing content using algorithms can alleviate the need for expensive artists, allow for much larger game levels, and allow funds to be focused elsewhere. They investigate multiple techniques such as parametric vs seed-based generation and weigh up the pros and cons of such techniques. They conclude by stating that there exist many suitable techniques for procedural content generation, for certain applications, but there are many areas of the field that still need to be explored.

(Parberry et al., 2014) proposes a method for generating realistic terrain for real time applications like games. The methods they use are a mixture of Perlin noise, a very fast noise generator, and real-world elevation data to build the height maps for the terrain. They justify the need for such techniques as they can produce more realistic terrain over other techniques, it can be run at real time, it is more varied and interesting, and it can be controllable by the designers and artists. They conclude by saying that their method of terrain generation can save processing time of Perlin noise methods, it allows intuitive designer control, and it can produce interesting and varied features, however for this method to be successful they need to extract interesting real-world terrain data.

(Parberry et al., 2014) present findings in the common uses of procedural generation techniques within the games industry. They justify their need for such a survey by saying that despite the abundance of procedural content generation techniques there are not commonplace with commercial products. They describe multiple benefits to such techniques such as reduced workload on designers and artists, large game levels and it can meet the increased demand of the consumer. However, they also describe the negatives to such a design philosophy such as less control in the hands of the designers, and it can be hard to match the quality of handmade content. They conclude by saying that existing procedural generation implementations are usually limited to certain aspects of games, but in order to reach consumer demand, further research needs to be done to expand procedural content generation applications.

(Van Der Linden et al., 2013) presents multiple techniques for generating environments. They illustrate how to use cellular automata and generative grammar to construct indoor and outdoor levels that can be scaled to create large and diverse experiences. Such environments include landscape and scenery such as trees, rocks and infrastructure. They emphasise that such strategies can take power from the developers, but it allows for fast content generation, which can save a developer time and money. They also conducted a survey of the methods they have been investigating and how they can be applied to certain aspects of the game, highlighting the popular methods and usage gaps. They conclude that such strategies are ideal for environments in the style of dungeons, and the next task would be to incorporate them into player-oriented objectives.

Procedural generation techniques can be used to create almost anything for video games, (Greuter et al., 2003), (Tutenel et al., 2009), and (Lopes et al., 2010) use it for architectural purposes, in order to create a more realistic and performant environment. Furthermore, it can be used to work in conjunction with authors and artist, (Smelik et al., 2010) and (Cardamone et al., 2011), present two different methods for achieving this.

Bibliography

- Mojang 2009, Minecraft, Windows, Mojang, Sweden.
- Greuter, S., Parker, J., Stewart, N. and Leach, G., 2003, February. Real-time procedural generation of pseudo infinite'cities. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (pp. 87-ff). ACM.
- Greuter, S., Parker, J., Stewart, N. and Leach, G., 2003, August. Undiscovered worlds–towards a framework for real-time procedural world generation. In *Fifth International Digital Arts and Culture Conference, Melbourne, Australia* (Vol. 5, p. 5).
- Tutenel, T., Bidarra, R., Smelik, R.M. and De Kraker, K.J., 2009, June. Rule-based layout solving and its application to procedural interior generation. In *CASA Workshop on 3D Advanced Media In Gaming And Simulation*.
- Lopes, R., Tutenel, T., Smelik, R.M., De Kraker, K.J. and Bidarra, R., 2010, November. A constrained growth method for procedural floor plan generation. In *Proc. 11th Int. Conf. Intell. Games Simul* (pp. 13-20). (Greuter etal., 2003)
- Smelik, R., Tutenel, T., de Kraker, K.J. and Bidarra, R., 2010, June. Integrating procedural generation and manual editing of virtual worlds. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games* (p. 2). ACM.
- Cardamone, L., Loiacono, D. and Lanzi, P.L., 2011, July. Interactive evolution for the procedural generation of tracks in a high-end racing game. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (pp. 395-402). ACM.
- 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 AI in Games*, *3*(3), pp.172-186.
- Parberry, I., 2014. Designer worlds: Procedural generation of infinite terrain from real-world elevation data. *Journal of Computer Graphics Techniques*, *3*(1).
- Hendrikx, M., Meijer, S., Van Der Velden, J. and Iosup, A., 2013. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *9*(1), p.1.
- Van Der Linden, R., Lopes, R. and Bidarra, R., 2013. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, *6*(1), pp.78-89.