**Abstract**

We propose a three step procedural map generation process capable of designing a pseudo-random map that is both navigable by units and generated in a timely fashion. The generated maps will be tested by an interactive zombie apocalypse simulation - Zombienation, for which the map generation process and output is optimized.

**1. Introduction**

Procedural map generation is the process of algorithmically defining the composition of a map, rather than via manual input. The most obvious benefits of this are saved time on the creation of playable material and the ability to introduce randomness into the game in question, allowing for a more unique experience with each play. Despite the potential for expensive computation, advances in hardware are reaching a point at which procedural techniques can be used to generate random material in real-time, as it is encountered by the user. The possibility of a seemingly infinite virtual world is an exciting prospect, and the primary motivation behind the proposed system.

The nature of Zombienation is such that our map generation procedure is limited to two dimensions, and does not need require any real-time calculation. We preserve the “on-the-fly” intention of procedural generation by initiating map creation immediately preceding use of the map. Various parameters have been set so as to maximize the aesthetic appeal and calculation time of the output, the results of which are discussed below.

**2. Related Work**

The Dead Linger

Real-time Procedural Generation of ‘Pseudo Infinite’ Cities (2003) – Stefan Greuter, Jeremy Parker, Nigel Stewart, Geoff Leach

Procedural Modeling of Cities (2001) – Yoav I H Parish, Pascal Mϋller

**3. Procedural Map Generation**

Three important aspects in the generation of any map are the creation of terrain, placement of roads, and organization of structures. These three elements are the primary components of our process, and by combining them we are able to create random maps that maintain structure and are usable by the apocalypse simulator.

In order to accomplish these sub-goals, we have implemented maze generation via Recursive Division to create roadways, Perlin Noise height maps for the placement of bodies of water, and randomized structure organization algorithms of our own devising.

**3.1 Recursive Division**

Recursive Division is the process of dividing a rectangle in half at a random interval with a particular orientation. Each child rectangle can be recursively divided with the opposite orientation until either the divisions are too small or a predetermined recursion depth is reached. Placing roadways along the boundaries of the final children yields either a maze with no set start or finish, or a sectioned city with interconnected roadways. The Recursive Division method of maze generation exceeds the usefulness of using Prim’s or Kruskal’s algorithm because it allows more control over the distance between roads and avoids the computational complexity of running searches on weighted graphs.

Our process uses two levels of Recursive Division. The first level divides the entire map into larger areas called Districts. To ensure relative similarity in the size of each District, the division interval (i) is limited to 1/3 < i < 2/3 the total length of the division axis. Districts are separated by double width roads to simulate highways that run through and outside the city. The second level of division occurs within each District, to further divide it into Sectors. Division ceases when the area of a Sector becomes too small or the preset recursion depth is reached. The effects of various depths on the overall calculation time are discussed below. Sector boundaries are marked with single width roads. To avoid an entirely rectangular city, some randomly decided Sector dividing roads are not placed. The final result is a city divided by highways, with interior roads that give the city a structured feel.

**3.2 Perlin Noise**

Perlin Noise is a method of generating noise by interpolating noise functions of various wavelengths to produce a smooth n-dimensional noise map. By creating two dimensional Perlin Noise, a can be extended into three dimensions by assigning altitude based on the noise value at each grid point. This can be used to create terrain, or for our purposes, assigning water tiles to every point on the grid below a threshold value. This method allows for generation of noise in a region with sides of length other than base two values (as is the case with the Diamond Square algorithm) and alleviates some of the complexity of calculating Simplex Noise.

The number of wavelengths used determines the smoothness of the resulting noise. We have set this, along with the depth threshold to accommodate realism and playability in generated water bodies and as such the effects of altering these values are not examined. In an attempt to spread water more evenly throughout the map, each District has its own Perlin Noise map calculated separately. The effects of generating water locally in this fashion are examined below. The final result of generating water in this fashion is a larger set of smaller water bodies distributed throughout the map that do not encroach on the dividing highways.

**3.3 Sector Distinction and Placement**

**4. Zombienation**

**4.1 Units**

4.11 Civilians

4.12 Workers

4.13 Rangers

4.14 Zombies

**4.2 Movement and Searching**

**4.3 Collision**

**5. Performance**

**6. Conclusion**

**6.1 Future Work**