41

Simulation Principles from *Dwarf Fortress*

Tarn Adams

41.1 Introduction
41.2 Principle 1: Don't Overplan Your Model
41.3 Principle 2: Break Down and Understand the System
41.4 Principle 3: Don't Overcomplicate
41.5 Principle 4: Base Your Model on Real-World Analogs
41.6 Conclusion
References

41.1 Introduction

Dwarf Fortress is a game built on simulation, generating unique game worlds for the player to experience through procedural content generation (PCG) [Dwarf Fortress 14]. Over the development, four guiding principles kept the game simulation robust and easy to work with. This chapter shares those principles with you.

Dwarf Fortress procedurally generates the entire world from scratch. The process begins with an elevation map, generated based on a randomized fractal. Next, it creates several map layers, which include temperature, rainfall, drainage, vegetation, and salinity. With the raw components of the map defined, the next step classifies each tract of land into different biomes (plant and climate types, such as woodlands or savannas). In the next phase of the simulation, temporary rivers wear down the mountains, followed by permanent rivers that flow from high ground to low ground. Plant and animal populations are then introduced, followed by world-specific fantasy creatures. At this point, the world creation is complete and the simulation of civilization begins. Through this process settlements are raised, trade routes are formed, and wars are waged. At a designated point in time, the simulation stops and the player begins the game in a unique living world.

41.2 Principle 1: Don't Overplan Your Model

Developers often make the mistake of overplanning their model. It's a problem with mechanics in general, but especially with simulation-based PCG, since you can't always predict the results and you don't necessarily want to. It can be extremely rewarding to play your own game and enjoy what emerges from the simulation. As a general strategy, it's best to get something running as soon as you can and then work from there until you're satisfied. Rather than overplanning the model, build it up through iteration.

41.3 Principle 2: Break Down and Understand the System

If you just focus on the end result, it's not clear what's required to make it look and work correctly. Instead, break down and understand the system you are modeling in terms of its basic elements and interactions. Not only will you develop a richer interplay of objects, but certain problems solve themselves. As a simple example, when creating terrain, it is tempting to spawn particular biomes or allow a fractal to directly define the biomes. However, *Dwarf Fortress* achieved much better results by handling fields separately: temperature, rainfall, elevation, drainage, etc. The interplay of those fields determined the final biome, resulting in a more natural, internally consistent solution.

41.4 Principle 3: Don't Overcomplicate

There's no reason to have 50 variables modeling one aspect of a game's behavior if they don't all have a meaningful impact on the game. Operate at the level of what the player sees or one layer below. Don't get carried away and create a lot of worthless noise. Not only do uselessly complicated systems hinder tuning, but they can have a paralyzing effect during development. Use complexity only where it's needed, otherwise strive for simplicity.

41.5 Principle 4: Base Your Model on Real-World Analogs

It's helpful to base your simulation on reality. If you have a real-world analog in mind, you can correct defects using a broader understanding of the fundamentals. In *Dwarf Fortress*, for example, the world maps improved greatly when rain shadows were taken into consideration (mountains block rain-producing weather systems, casting a shadow of dryness on the other side). Oddly enough, drainage was another nonobvious consideration that helped smoothly delineate forests from swamps. We know the real world works, so if you fall back on reality, you can usually get the simulation to work as well, given adequate memory and CPU time.

41.6 Conclusion

Regardless of what you're simulating, the preceding four principles should help you craft the game you actually want. When starting out, don't overplan the model. Build the model up over time, break down the components, and try to fully understand the underlying system. Base the model on real-world analogs so that you can benefit from systems that are

known to work. Allow the individual elements to combine, producing results which can be varied, surprising, and yet still responsive to your adjustments. By keeping it simple, you can avoid black box simulations and ultimately stay in control of your game.

For further information and a great wealth of PCG tips and tricks, please see the chapter "Procedural Content Generation: An Overview" in this book or visit the PCG Wiki [PCG Wiki 14].

References

[Dwarf Fortress 14] Dwarf Fortress. 2014. http://www.bay12games.com/dwarves/ (accessed September 10, 2014).

[PCG Wiki 14] Procedural Content Generation Wiki. 2014. http://pcg.wikidot.com/ (accessed September 10, 2014).