Procedural Map Generation in Video Games

Is this the end of level design as we know it?

Jack Fletcher

AINT354 Design for Entertainment Systems
Jackfletcher@turtledogstudios.co.uk
turtledogstudios.co.uk

ABSTRACT

Content creation within video games is steadily becoming more resource intensive, with AAA games typically containing large maps that have to be filled with a consistent level of content to deliver an immersive experience. With indie or solo developers, this may not always be a viable methodology due to time or budget constraints. This paper highlights key implementations of procedural content generation (PCG) within game development and investigates the effectiveness of their chosen solutions within the domain of map generation. Later, this paper discusses perlin noise, fractal landscapes and cellular automata to understand their advantages and disadvantages.

Keywords

Perlin Noise, Seeds, L-Systems, Fractal Landscapes, Simplex noise, Billow noise, Ridged noise, Worley Noise, Domain Warping, Gradient noise.

1. INTRODUCTION

Video games typically feature some form of world space which the player interacts with, whether this is by directly moving their character (*Monster Hunter World*, Capcom, 2018), or by modifying the terrain (*Minecraft,* Mojang, 2009). In game development, creating a world can consume a large portion of development time within the life cycle of the project, adding significant cost overhead, with many AAA games venturing into hundreds of millions of dollars to produce (Downward Thrust, 2016). These AAA games, often with higher budgets, create worlds such as those within *Skyrim* (Bethesda Game Studios, 2011) or *Red Dead Redemption* (Rockstar North, 2010). They are relatively small, with maps of 15 square miles and 28 square miles respectively (See figure 1), while being resource intensive to develop.

As there is such variation in what type of games are created, there is not a 'best' methodology for world creation. PCG would not have worked effectively in *The Witcher 3* (CD Projekt Red, 2015), due to its reliance on the fictional world created by Andrzej Sapkowski in '*The Witcher*' (Gollancz, 2008) series.

During the 2017 Games Development Conference (GDC), Sean Murray explains the issues Hello Games faced while developing *No Man's Sky* (Hello Games, 2018). They tried different methodologies of procedural content generation, each having their own disadvantages and advantages, before selecting a variation of perlin noise.

During their 2019 GDC presentation, Insomniac Games compared their design pipelines for one of their previous games, *Sunset Overdrive* (Insomniac Games, 2014), and their newest game, *Marvel's Spider-Man* (Insomniac Games, 2018). They discovered that hand-crafting modular building templates as they did previously, would not work for the vastly larger world that *Marvel's Spider-Man* (2018) required, and had to find a different solution.

The remainder of this paper is split into sections in the following order: Section 2 discusses PCG as a concept and some use cases such as character dynamics. Sections 3-3.3 introduces key players in PCG, explaining and comparing implementation and discussing their unique selling points. Section 3.4 covers other related technologies and approaches.

Figure 1. A comparison between typical AAA game map sizes (Wong L, 2015)

2. FIELD

The cost of development is minimized through the implementation of PCG. PCG is the concept of using random, or pseudo random procedures to create game content that is not hand crafted, or persistent through each playthrough of the game. This can be as simple as adding details such as grass to handcrafted elements, or as complex as developing an entire solar system. Kate Compton describes an interesting problem known as the '10,000 bowls of oatmeal problem' (Compton K, 2016). This problem theorizes that regardless of how effective your PCG algorithm is at generating infinitely new areas, if the player does not see a perceivably new difference, there is no real incentive to explore as it is not perceptually unique. The problem is explained as such,

> 'I can easily generate 10,000 bowls of plain oatmeal with each oat being in a different position and

different orientation, and mathematically speaking they will all be unique. But the user will likely just see a lot of oatmeal.' (Compton K, 2016)

3. KEY PLAYERS

While games companies are not likely to give away their algorithms for fear of clones being created, they often talk about the type of technologies that were used within their algorithm. As such, the information contained within this section is unlikely to show any clear examples of their implementation. However, it will give some insight into how they created it.

3.1 No Man's Sky

No Man's Sky (Hello Games, 2016) is an open world, action-adventure survival game, in a self-proclaimed 'infinite procedurally generated galaxy' (*No Man's Sky* Press Kit, 2016). To create this, they procedurally generated almost all components of the game - to provide 18 quintillion planets, a functionally infinite number of possibilities. Due to the scope of this project, only terrain generation will be covered, however in section 4 a link to presentations by Innes McKendrick and Grant Duncan will be provided for further context.

The overworld terrain of *No Man's Sky* (2016) is generated using what Sean Murray describes as 'Uber Noise' (Murray S, 2017 *Building Worlds Using Math(s)*). This is comprised of several layers of perlin noise (as discussed in Section 3.4), including ridged and billowed noise to create sharp ridges and rolling mountains respectively. An analytical derivative is utilized to measure the rate of change within the noise to determine whether the voxel is on a slope. This is then used to modify the amplitude of octaves responsible for minute details to create erosion within terrain features such as mountains. Uber noise generates a height map similar to that shown in figure 2, which uses colour to show the variation between high and low points within the noise map.

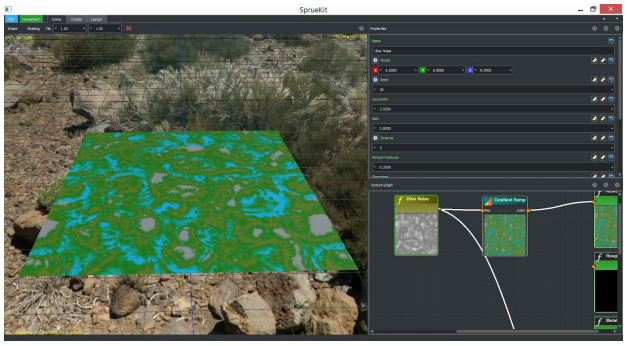

Figure 2. A heightmap based on 'Uber Noise' (u/PickedChicken, 2017)

3.1.1 Sid Meier's Civilization 1

Sid Meier's Civilization (MicroProse, 1991) is a turn based strategy game, where the player aims to either reach Alpha Centauri (building a spaceship) or obliterate all other factions, thereby winning the game. It utilizes PCG through a set of four generation stages, each taking an input and creating the land mass, size of biomes, variation of biomes and then creates the minute details. While it cannot be confirmed, this is likely using Moore's neighbourhood technique within cellular automata, due to how it generates each chunk, shown in *Civ1 Map Generation Explained* (DarkPanda, 2013).

3.1.2 Minecraft

Minecraft (2009) is a sandbox based game originally developed by Mojang that utilizes a form of 3D perlin noise (see Section 3.4) to create its expansive terrain. *Minecraft* (2009) generates terrain in what is informally known as a 'chunk', which is a 256x16x16 (Height x Width x Depth) segment of a world that is generated when the player spawns for the first time, or enters a new chunk.

Minecraft's (2009) chunk generation occurs in a set of stages, with each stage replacing parts of the last. In the first stage, the world's shape is decided by using several noise map layers, to create terrain roughness, elevation and local detail. Water is then added to all empty spaces below or equal to the y value of 62 using a seperate noise map. The top layer of the stone is then replaced with the relevant surface material such as dirt, grass or sand. Next, caves are created by using the noise values from a perlin noise map to highlight where blocks should be removed. Ores are then added to the chunk; this is done after the caves are formed to be less computationally expensive. Ore can only replace a block that was previously not air, and creating caves results in more air blocks throughout the chunk. Finally, foliage is added to the world.

To place plants and terrain within the *Minecraft* (2009) world, a Whittaker Biome Diagram (Whittaker RH) is used to determine whether plants or terrain features like ice, sand or snow are used in place of water, dirt or stone.

3.2 Summary

In *Building Worlds Using Math(s)* (2017), Sean Murray states that *No Man's Sky's* (2016) terrain generation took the team several years to complete, throughout each of its iterations. While this is suitable for a large scale operation with no definitive time limit, this research has to define PCG that can be viable for a project with a three month time limit. Due to this, an algorithm to the scale of *No Man's Sky* (2016) is not feasible, and if a similar approach is taken, it would have to use a cut down methodology, such as only using a few variations of perlin noise. Furthermore, *No Man's Sky* (2016) creates worlds only during runtime, whereas during the project, using procedural generation as an editor tool may be more effective.

Sid Meier's Civilization (1991) does not give any useful insight into how to develop a viable PCG, however it does give an insight into how to use it to raise the skill ceiling of a game.

Minecraft (2009) emphasises a logical ordering to your generation algorithm, only sampling the voxels that are required to make the algorithm more performant.

Table 1. Types of PCG compared by their potential uses.

Game	Type of PCG	Potential Uses	Insight
No Man's Sky	'Uber Noise' (Perlin noise derivative)	3D world generation, asset generation, fauna/flora generation	A derivative of perlin noise is a likely candidate for small scale operation
Civilization	Custom	2D and 3D map generation	Using PCG to generate new maps raises the skill ceiling of games
Minecraft	Trilinear filtered low density perlin noise field	3D world generation, island/capsule world generation, cloud and texture generation	Effective sampling of voxels will greatly improve performance

3.3 USP

In *No Man's Sky* (2016), the team developed a unique world generation algorithm, which fits their exploration game and created functionally infinite universes. 'It was calculated it would take you about five billion years to explore every 2 to the power 64, or 18 quintillion, planets that this game has to offer.' (Jelle, 2017). This in itself is what attracted player's to it, as it does not offer any new or even innovative iterations on previous game mechanics, it simply allows the player to explore indefinitely.

Minecraft (2009) is a similar style game to *No Man's Sky* (2016), all terrain is procedurally generated and it is functionally infinite. However, *Minecraft*'s (2009) main selling point is its ability to be used as a sandbox, where the limit is the player's creativity. For example, you can build small villages or hunt for resources, or create redstone contraptions that mimic real life electronics, such as a quad core computer. (legomasta99, *[Minecraft Computer Engineering] - Quad Core Redstone Computer v5.0,* 2018*)*

*Sid Meier's Civilization 1 (*1991*)*, is a turn-based strategy 4X game, where the main focus is on creating an empire. Due to this focus, being able to create fresh maps for each experience means that there are no 'meta' positions to begin your empire and it is not easy to predict other player's movements or actions. This introduces a higher skill ceiling into the game, where players have to weigh the resources surrounding a potential city site against inferred knowledge. Therefore, while PCG is not the main selling point of this game, it is integral to the replayability of the game, whether this is in singleplayer or multiplayer. This is affirmed by Johnson et al ,'First, having an in-exhaustible source of new maps means that levels become less predictable, which contributes to the player's curiosity and the game's life-span.' (Johnson et al, *Cellular automata for real-time generation of infinite cave levels,* 2010*)*

3.4 Technologies/Approach

When creating topological features within game environments, a common requirement is a naturalistic environment. In purely random generation, large stalactite like formations may occur due to there being no relationship between any voxels (Figure 3), which is unlikely to create a realistic environment. To combat this, techniques such as applying various noise filters have been developed.

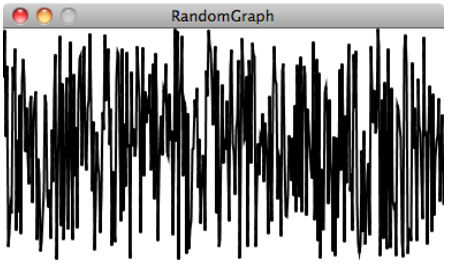

Figure 3. Random numbers over time, Khan academy

3.4.1 Perlin Noise

Perlin noise, developed by Ken Perlin in the early 1980s (Perlin K, *An Image Synthesizer,* 1985) and improved upon in 2002 (Perlin K, *Improving Noise,* 2002), is commonly used throughout the games industry. This is due to the ease of creating naturally smooth looking terrain. An example of this is shown in Figure 4, where a pseudo-random group of floats was generated on a 2d plane to create a height map, which was then applied to terrain. This is relatively simple to implement into games and can be layered with other perlin noise functions, known as a 'Fractional Brownian Motion'. As the simplest of all composite perlin noise algorithms, it uses separate 'octaves' made up of a weighted sum of perlin noise. This is then responsible for a certain set of features, for example, octave 1 may be responsible for landmass, octave 2 may be responsible for boulders, and octave 3 may be responsible for grass. This makes it suitable for developing large scale environments, whether this is during development, to create a world space which then has handcrafted elements added to it, or created at runtime with other elements mixed in. The latter of these can be seen to be successful within *Minecraft* (2009), where every different seed guarantees a different experience.

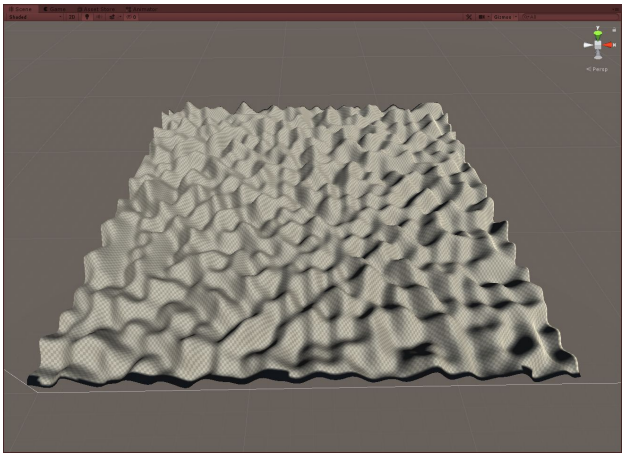

Figure 4.Perlin noise being used to generate a simple PCG landscape within the Unity engine.

3.4.2 *Fractal Landscapes*

A fractal is a complex pattern that at any level of zoom, gives the same pattern. When creating a landscape that is aiming to be realistic, fractals are ideal as they are self similar, and are commonly found within nature, for example, in snowflakes.

This is typically done through midpoint displacement, which finds the midpoint between two lines and mutates it vertically by a random amount. This process is then repeated for a predetermined number of iterations. This was found to create square shape artifacts, which was then fixed using the diamond-square algorithm (Fournier et al, *Computer rendering of stochastic models*, 1982). This adds a step before each line displacement, where it splits a rectangle into four diamonds, which have midpoint displacement applied before moving onto the full rectangle.

Gavin S.P Miller found that this variation was still flawed, 'Fractal subdivision methods are slow and generate defects due to what is known as the 'creasing' problem' (Miller G, *The Definition and Rendering of Terrain Maps,* 1986). Shown in figure 5 is an example of a diamond-square-algorithm within the Unity game engine, which clearly shows the defects and creases that can appear.

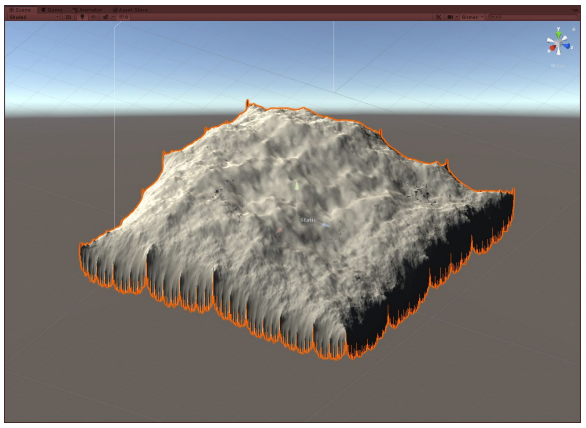

Figure 5: An example of DSA within the Unity engine, with visible creasing in the top left of the terrain.

3.4.3 *Cellular Automata*

Cellular automata (Von Neumon, *The General and Logical Theory of Automata*, 1951) are concepts in computer science similar to that of a vending machine. They take an input, such as coins and a decision, representing a mathematical equation, and output a drink based on the aforementioned inputs, corresponding to a specific set of states within the automaton.

This is represented within a cellular automaton as an n-dimensional grid, where each point within the grid represents a cell with a set of possible states for each cell and a transition rule. In a simple example, a cell could have an on or off state. Using the transition rule, an automaton evolves through a system of steps, where at each step a cells new state is governed by the states of the neighbouring cells, known as a neighbourhood. (Figure 6). In a one dimensional automaton known as a vector, neighbourhoods are defined by size n, where n is the number of squares on either side of the cell.

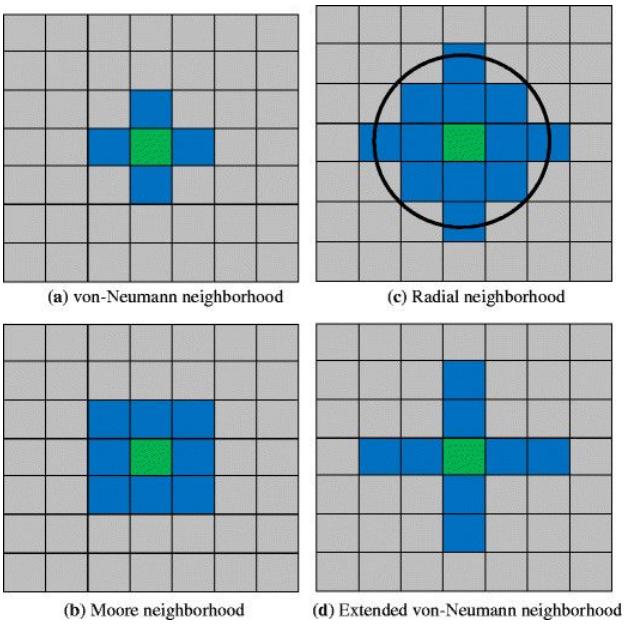

Figure 6. Variations of a CA neighbourhood, Dutta Abhishel et al, 2015)

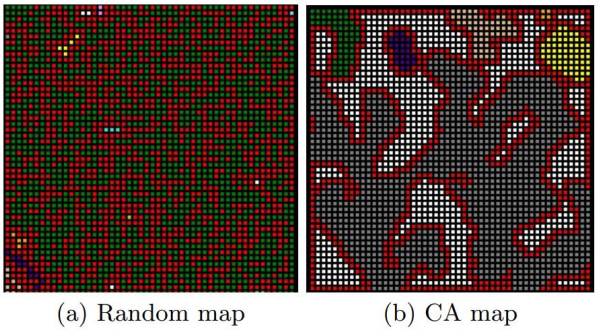

Figure 7. A comparison between a randomly generated cave map and a map generated using Cellular automata, (Johnson et al, 2010)

Typically, Cellular automata is used in game development to create dungeons, similar to those found in *Diablo* (Blizzard Entertainment, 1998). The ability to create natural looking cave systems such as those found in Figure 7 makes it ideal for this use case, and can even be applied to 3D dungeons such as in *Dark Cloud* (Sony Computer Entertainment, 2000). However, the use of cellular automata within PCG is typically limited to dungeon crawlers in practice, as a blend or perlin noise or fractal geometry is better for creating large landscapes.

4. ACKNOWLEDGMENTS

For more information regarding procedural content generation, Inigo Quilez has several informative blog posts on his website.

https://www.iquilezles.org/www/index.htm

Innes McKendrick's presentation on continuous world generation within *No Man's Sky* at Game Developers Conference 2017.
https://www.gdcvault.com/play/1024265/Continuous-World-Generation-in-No

Grant Duncan's presentation on the art pipeline within *No Man's Sky* (2016).

https://www.gdcvault.com/play/1021935/Do-Artists-Dream-of-Electric

5. CONCLUSIONS

In this paper several methodologies for creating game content were outlined and while none of them were definitively better than one another, each of them had their advantages and disadvantages. Due to this, it is probable that a developer will choose a specific algorithm based on stylistic choices. For example, the use of fractal landscapes to give a more eroded appearance to mountains. The primary concern when developing game maps is not strictly a PCG problem; but the uniqueness of terrain that makes a player believe they are exploring a new area. However, due to the ease of incorporating perlin noise into world generation, combined with its multitude of variations, it is suggested that perlin noise be used as the basis for overworld terrain for smaller projects.

6. REFERENCES

Compton, Kate (2016). *So you want to build a generator...* [online] Available: https://galaxykate0.tumblr.com/post/139774965871/so-you-want-to-build-a-generator. [Accessed 16 Oct. 2019].

DarkPanda (2013). *Civ1 Map Generation explained*, CivFanatics Forums [online] Available: https://forums.civfanatics.com/threads/civ1-map-generation-explained.498630/ [Accessed 16 Oct. 2019].

Downward Thrust (2016). *How Much Do "AAA" Games Cost To Make?*. [online] YouTube. Available: https://www.youtube.com/watch?v=vV_Nq9bjvXA [Accessed 16 Oct. 2019].

Dutta, Abhishek & Kar, Saurajyoti & Apte, Advait & Nopens, Ingmar & Constales, Denis. (2015). *A generalized cellular automata approach to modeling first order enzyme kinetics.* Sadhana. Available: https://core.ac.uk/download/pdf/55844926.pdf [Accessed 17 Oct. 2019].

Fournier, Alain; Fussell, Don; Carpenter, Loren (June 1982). *Computer rendering of stochastic models*, Communications of the ACM. 25 Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.85.3781&rep=rep1&type=pdf [Accessed 17 Oct. 2019].

Johnson, Lawrence., Yannakakis, Georgios N. and Togelius, Julian (2010) *Cellular automata for real-time generation of infinite cave levels*. 1st ed. New York: ACM, Available: https://core.ac.uk/download/pdf/132619480.pdf [Accessed 17 Oct. 2019].

legomasta99, (2018) *[Minecraft Computer Engineering] - Quad Core Redstone Computer v5.0* Available at: https://www.youtube.com/watch?v=SbO0tqH8f5I (Accessed: 17 Oct. 2019)

Miller, Gavin S. P. (August 1986). *The definition and rendering of terrain maps*. ACM SIGGRAPH 1986 Computer Graphics Dallas, ACM Available: https://metalbyexample.com/wp-content/uploads/miller-terrain.pdf [Accessed: 17 Oct. 2019]

Murray Sean (2017) *Building Worlds Using Math(s)* Game Developers Conference, Moscone Center San Francisco, CA Feb 27, Available: https://www.gdcvault.com/play/1024514/Building-Worlds-Using [Accessed 14 Oct. 2019].

No Man's Sky (2016). Press - No Man's Sky. [online] Available at: https://www.nomanssky.com/press/ [Accessed 17 Oct. 2019].

Perlin, Ken. (1985). *An image synthesizer*. In Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), ACM, New York, NY, Available:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.220.2248&rep=rep1&type=pdf [Accessed 17 Oct. 2019].

Shaker Noor, Liapis Antonios, Togelius Julian, De Vasconcelos Abreu Lopes Ricardo, & Bidarra Rafael (2016). *Constructive generation methods for dungeons and levels.* In Procedural Content Generation in Games (pp.44). Available: http://pcgbook.com/wp-content/uploads/chapter03.pdf [Accessed 17 Oct. 2019]

Von Neumann, John (1951), *The General and Logical Theory of Automata* in Cerebral Mechanisms in Behavior: The Hixon Symposium, New York: John Wiley & Sons. Available:https://pdfs.semanticscholar.org/e853/8f11920fa6e56b3d34771bb330bd3e07281d.pdf [Accessed 17 Oct. 2019]

Wong Luis (2015), *A comparison of maps in AAA games*, Available: https://elcomercio.pe/blog/geekgames/2015/05/the-witcher-3 [Accessed 17 Oct. 2019]

7. BIBLIOGRAPHY

AntVenom. (2017). *How do Minecraft Worlds Generate?*. 22 October 2017, online, Youtube. Available: https://www.youtube.com/watch?v=FE5S2NeD7uU [Accessed 16 Oct. 2019]

Duncan, Grant (2015) *Do Artists Dream of Electric Sheep?,* Game Developers Conference, Moscone Center San Francisco, CA March 2, Available: https://www.gdcvault.com/play/1021935/Do-Artists-Dream-of-Electric [Accessed] 14 Oct. 2019

Fingas, John (2015). *Here's how 'Minecraft' creates its gigantic worlds.* online, Available: https://www.engadget.com/2015/03/04/how-minecraft-worlds-are-made/?guccounter=1. [Accessed 16 Oct. 2019].

Jelle (2017). *Top 10 Biggest Open-Worlds Ever.* online, Available: https://gamehypermart.com/blog/top-10-biggest-open-worlds-ever. [Accessed 16 Oct. 2019].

Persson, Markuss (2011) *Terrain generation, Part 1*. online Available: https://notch.tumblr.com/post/3746989361/terrain-generation-part-1. [Accessed 16 Oct. 2019].

Weisstein, Eric W. (2019) *Cellular Automaton*. From *MathWorld* A Wolfram Web Resource. Available: http://mathworld.wolfram.com/CellularAutomaton.html [Accessed 14 Oct. 2019]

Whittaker, RH (2018), Whittaker Diagram [ONLINE]. Available at: https://web.archive.org/web/20130423230809if_/http://www.marietta.edu/~biol/biomes/whittaker.jpg [Accessed 26 Oct. 2019]