Concrete Earth: A DirectX Game

Sam Drysdale

May 16, 2023

Contents

1 Summary **1**

2 User Controls **2**

3 Features **2**
3.1 Procedural Terrain . . . 2
3.1.1 Case Study: Hexes . . . 4
3.1.2 Case Study: Landmarks . . . 5
3.2 Procedural Screen Shaders . . . 6
3.2.1 Case Study: Ciphers . . . 6
3.2.2 Case Study: Blood Vessels . . . 7
3.3 Procedural Narrative . . . 8
3.3.1 Grammars . . . 8
3.3.2 Content Selection Architectures . . . 9

4 Code Organisation **9**
4.1 Rendering . . . 9
4.2 GUI . . . 9

5 Evaluation **10**
5.1 Features . . . 10
5.2 Code Organisation . . . 10

6 Conclusions **11**

References **12**

1 Summary

Concrete Earth is a game about nuclear semiotics. Ostensibly, the story is set in a medieval fiefdom, cursed by the occult ciphers long-etched into its landscape. The twist is this occurs millenia *after* a nuclear apocalypse - the 'curse' is radiation poisoning, the ciphers the dead language of the previous civilisation, warning of nearby waste repositories. Though it is still in development, this prototype demostrates how the DirectX 11 functionality might bring such a concept to life. Indeed, for a game so concerned with language, it is appropriate that many key features are grounded in linguistics; while the terrain itself is generated by marching cubes, *Concrete Earth* uses grammars to create both procedural screen shaders and a procedural narrative.

2 User Controls

The user plays as a carriage driver, journeying forwards over an endless map. Periodically, they will be stopped by a 'multiple-choice' event; an option must be selected before continuing. Such is the core game loop of *Concrete Earth*.

The application uses the following key mappings:[1]

W	Move north
A	Move northwest
D	Move northeast
1	Select option #1
2	Select option #2
3	Select option #3
Esc	Quit

Clicking **LMB** will also select an option. The application runs at a fixed 1920×1080 resolution.

3 Features

The following is a technical discussion of the key features of *Concrete Earth*, with a focus on its advanced procedural generation. Mathematical models and code snippets are kept to a minimum, edited for clarity rather than accuracy to the original application.

3.1 Procedural Terrain

Given the concept, this game's terrain is of course intended to evoke a sense of "shunned land" (Trauth et al. 1993). In generating the jutting thorns and blocks so essential to the post-nuclear setting, though, there comes a problem - concavity. Height maps are perfectly adequate for creating rolling hills and valleys, but the fact that each of their (x, z)-coordinates can correspond to only one y-value would prevent any overhangs in *Concrete Earth*'s barren landscape (see Figure 1).

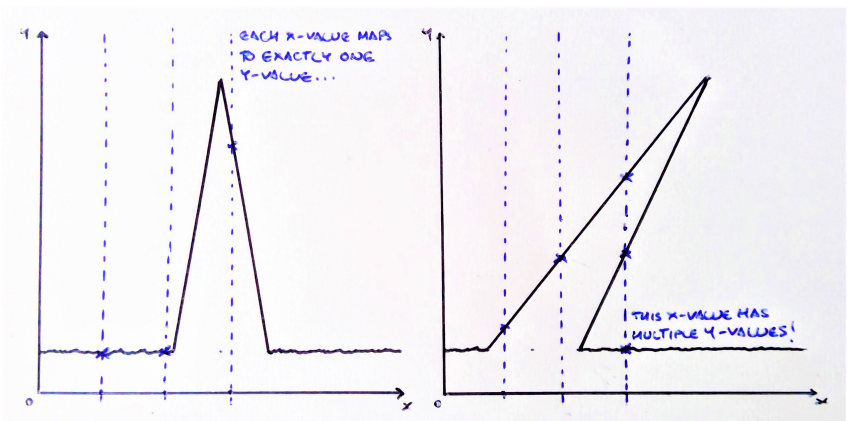

Figure 1: Two sketches of 'thorny' terrain; the former can be generated by a height map, whereas the latter is too concave.

[1] While the framework registers inputs on press, note that options are only selected *on release*.

Consider the two-dimensional analogue of this problem: how does one construct the bounding curve of a (potentially concave, or even disconnected) area? Taking an $(n+1) \times (n+1)$ grid, with $n \in \mathbb{N}$,[2] let the scalar field $f : [0,1]^2 \to \mathbb{R}$ assign each gridpoint (i,j) a corresponding value $f_{i,j} = f(i/n, j/n)$. Figure 2 shows how to partition a cell so as to enclose the gridpoints with $f_{i,j} < c$ constant. Drawing such a partition into every cell of the grid will therefore approximate the *isoline*[3] $f(x,y) = c$, with increasing accuracy as $n \to \infty$.

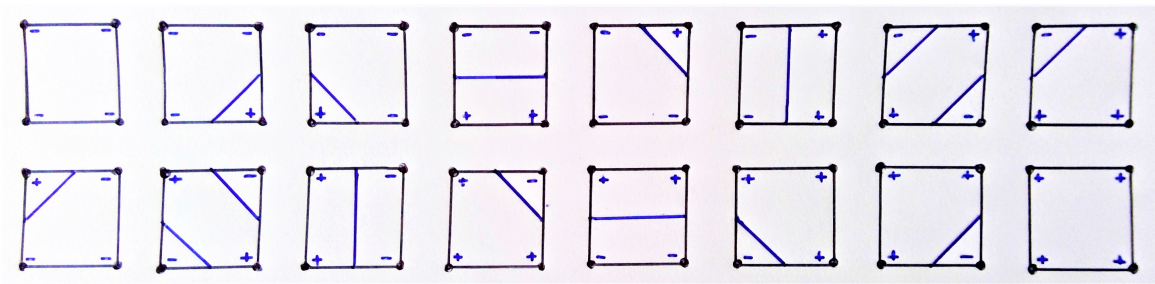

Figure 2: Partitions of a marching square. With each of the 4 gridpoints existing in one of 2 states ('+' for $f_{i,j} \geqslant c$, '−' for $f_{i,j} < c$), exactly $2^4 = 16$ partitions are possible.

This is the *marching squares* algorithm, so named for how it uses a `for` loop to 'march through' and fill in each of the $n \times n$ square cells. Though a challenge to conceptualise, intuition can be built by running the 2D algorithm with pen and paper; Figure 3 contains manual constructions for

$$f(x,y) = \min\left((x-0.5)^2 + (y-0.5)^2, 4\left((x-0.75)^2 + (y-0.75)^2\right)\right),$$

a scalar field describing two overlapping circles.

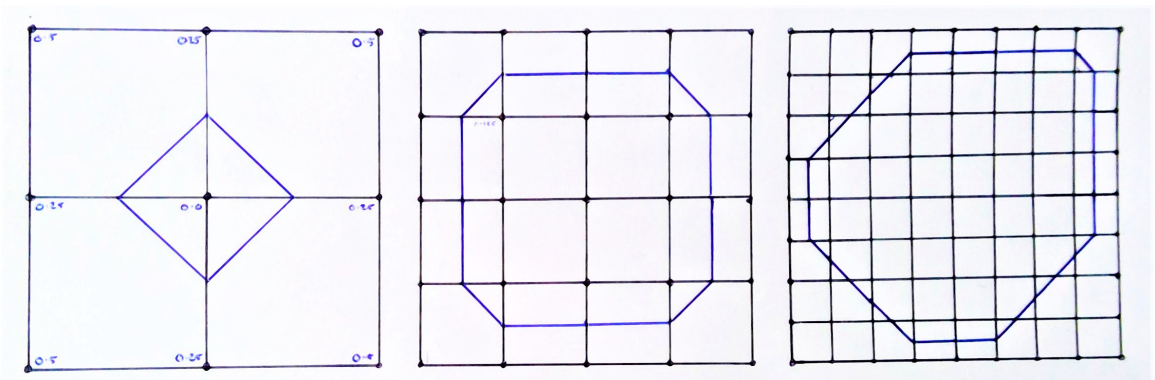

Figure 3: Bounding curve $f(x,y) = 0.16$, at increasing levels of granularity.

Consider gridpoints $\mathbf{a}$, $\mathbf{b}$, such that $f_{\mathbf{a}} < c < f_{\mathbf{b}}$. By Figure 2, their partitioning line will bisect edge AB exactly - if the partition is to represent an isoline, then it surely requires $f(0.5\mathbf{a} + 0.5\mathbf{b}) \approx c$, but

[2] This could equally be an $(n+1) \times (m+1)$ grid; keeping the dimensions the same is just 'neater'.
[3] Informally, a 'contour line' of the $2D$ field.

this is a rather arbirtrary assumption about the scalar field. Instead making the standard, first-order approximation that f is linear over short distances, it would follow that

$$f\left((1-t)\mathbf{a}+t\mathbf{b}\right) \approx c, \text{ where } t = \frac{c - f_{\mathbf{a}}}{f_{\mathbf{b}} - f_{\mathbf{a}}}.$$

Understanding that it is more accurate for the partition to intersect AB at $(1-t)\mathbf{a}+t\mathbf{b}$, such linear interpolation is integrated into the marching squares algorithm (see Figure 4).

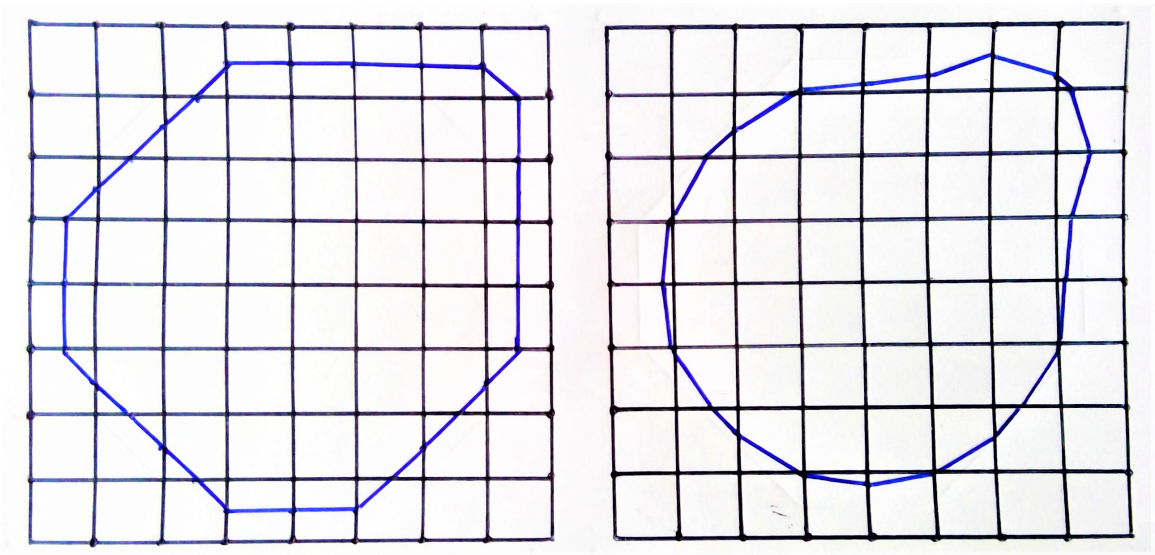

Figure 4: Bounding curve $f(x, y) = 0.16$, before and after linear intepolation.

Coming back to the task at hand, then, *Concrete Earth* uses *marching cubes* to generate *isosurfaces* bounding 3D volumes. Partitioning with surfaces... (which,). [Outside of the expected changes - scalar fields now operate on domain $[0, 1]^3$, vertices now have $2^8 = 256$ possible configurations along [the edges of] each marching cube - this is actually a rather straightforward generalisation.]

This report will not offer a full breakdown of the 3D algorithm. In particular, the finer points of the DirectX implementation - [breaking down into triangles], or identifying adjacent marching cubes to construct 'smooth' vertex normals[4] - are deliberately . [Intention of this section, focused on intuition, guiding principle...]. Paul Bourke's *Polygonising a Scalar Field* (1994) remains the definitive source on the technique, and one that this project's version relies on heavily.

3.1.1 Case Study: Hexes

Having established this framework, generating terrain becomes a matter of writing the right scalar field. [Ironically enough, ...is similar enough a height map].

Having established this framework, generating terrain becomes a case of

$$\text{Height}(\mathbf{x}) = y + 0.1 \cdot \text{FBM}(\mathbf{x}),$$

[...] with a (scaled down to avoid obvious 'floating islands'). This report treats the underlying simplex noise as a black box; [justification].

[4] These being averaged from the surface normals of the surrounding faces; if , then it will be weighted by $1/((\mathbf{b}-\mathbf{a}) \times (\mathbf{c}-\mathbf{a})) \propto$ the inverse of its area

To then [...].

$$\mathrm{Hex}(\mathbf{x}) = \frac{\sqrt{(2x-1)^2 + (2z-1)^2}\cos\left((\theta \bmod \pi/3) - \pi/6\right)}{\cos(\pi/6)}, \text{ where } \theta = \operatorname{arctan2}(2z-1, 2x-1).$$

[...] a vertical hexagonal prism

$$\mathrm{Terrain}_c(\mathbf{x}) = \max\left(\mathrm{Height}(\mathbf{x}),\, c \cdot \mathrm{Hex}(\mathbf{x})\right).$$

[Explain maximum; explain c.]

[Bounding hexagonal prism... As much as flat faces are antithetical to marching cubes (if they run parallel to the grid, then... can't interpolate? notice that interpolation doesn't account for the degenerate case where two adjacent gridpoints are equal)...]

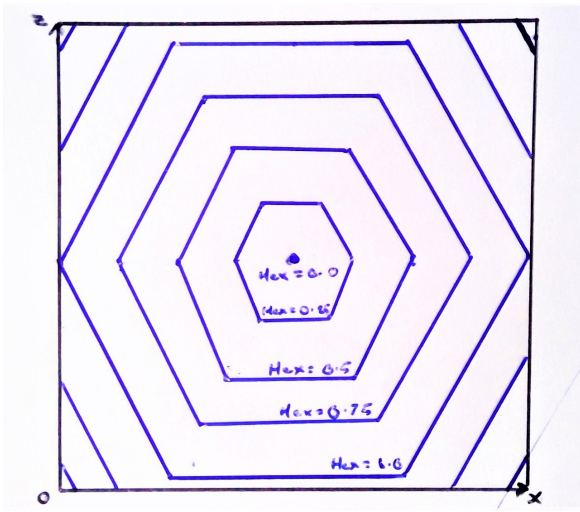

Figure 5: Isoclines of $\mathrm{Hex}(x, y_0, z)$, a horizontal slice of the hexagonal field.

3.1.2 Case Study: Landmarks

apex $\mathbf{a}$, base with centre $\mathbf{b}$, and half angle ϕ,

$$\mathrm{Thorn}_{\mathbf{a},\mathbf{b},\phi}(\mathbf{x}) = \frac{1}{\phi}\arccos\left(\frac{(\mathbf{x}-\mathbf{a})\cdot(\mathbf{b}-\mathbf{a})}{|\mathbf{x}-\mathbf{a}|\cdot|\mathbf{b}-\mathbf{a}|}\right).$$

Much like before,

$$\mathrm{Landmark}_{c;\,\mathbf{a},\mathbf{b},\phi}(\mathbf{x}) = \min\left(\mathrm{Terrain}_c(\mathbf{x}),\, c \cdot \mathrm{Thorn}_{\mathbf{a},\mathbf{b},\phi}(\mathbf{x})\right),$$

noting

[Explain how we add randomness to thorns]. That said, [smooth, geometric structure].

All landmarks follow this same pattern.

[Figure: High-quality render of 3x2 tiles (top down and on angle)]

3.2 Procedural Screen Shaders

The post-processing in *Concrete Earth* is, in many respects, trivial. `vignette_ps.hlsl`, for instance, calls only two renders-to-texture on every frame: the board itself, and an alpha map of blood vessels sprouting from the edges of the screen. As striking as the final effect is, the shader is surprisingly straightforward in blending the textures into a single, pulsing eye strain overlay; far more worthy of further discussion is how the blood vessels themselves are generated.

In formal languages, a grammar is a tuple $G = (N, \Sigma, P, \omega_0)$. This contains two disjoint sets of symbols: nonterminals $A, B, \cdots \in N$, and terminals $a, b, \cdots \in \Sigma$. The production rules in P map nonterminals to strings $\alpha, \beta, \cdots \in (N \cup \Sigma)^*$; applied recursively to the axiom $\omega_0 \in (N \cup \Sigma)^*$, these rules can produce increasingly complex *sentences* of terminals and/or nonterminals.[5]

The Chomsky hierarchy (Chomsky 1956) classifies grammars by their production rules:

Type-3. Regular grammars map $A \mapsto a$ or $A \mapsto aB$.

Type-2. Context-free grammars map $A \mapsto \alpha$.

Type-1. Context-sensitive grammars $\alpha A \beta \mapsto \alpha\gamma\beta$.

Type-0. Unrestricted grammars map $\alpha \mapsto \beta$, where α is non-empty.

Note that all Type-3 grammars are also Type-2, all Type-2 grammars also Type-1, and so on.

Suppose, for example, that $N = \{F, G\}$, $\Sigma = \{+, -\}$, $P = \{F \mapsto F + G, G \mapsto F - G\}$, $\omega_0 = F$. Letting ω_n denote the sentences generated by applying the production rules n times, it follows that

$$\begin{aligned}
\omega_1 &= F + G, \\
\omega_2 &= F + G + F - G, \\
\omega_3 &= F + G + F - G + F + G - F - G, \\
\omega_4 &= F + G + F - G + F + G - F - G + F + G + F - G - F + G - F - G, \quad \cdots .
\end{aligned}$$

While these defintions are rather abstract, Lindenmayer (1968) provides a remarkable application. Treating each symbol as an instruction like 'go forward' or 'turn right', *L-systems* visualise sentences via 'turtle graphics'; when those sentences have been generated recursively by a grammar, the line drawings inherit that same self-similar structure. Consider the above, where reading non-terminals F, G as 'draw a line while moving one unit forwards,' and terminals $\pm$ as 'turn $\pm\pi/2$ on the spot,' produces the fractal dragon curves in Figure 6.

While L-systems are most common in the modelling of 3D plants and other branching structures (Prusinkiewicz & Lindenmayer 1996), *Untitled* only uses them to generate 2D alpha maps. Moreover, it restricts its attention to L-systems paired with context-free grammars.

3.2.1 Case Study: Ciphers

Parametric L-systems (Hanan 1992) exist as a generalisation of the above. [theory].

The modules in *Concrete Earth*, then, track three . More than anything, this offers a certain clarity of code - at least from a

[Example: various geometric runes!].

[5] In mathematical literature, $\omega_0 \in N$ (Hopcroft, Motwani & Ullman 2000), but *Untitled* takes an informal approach.

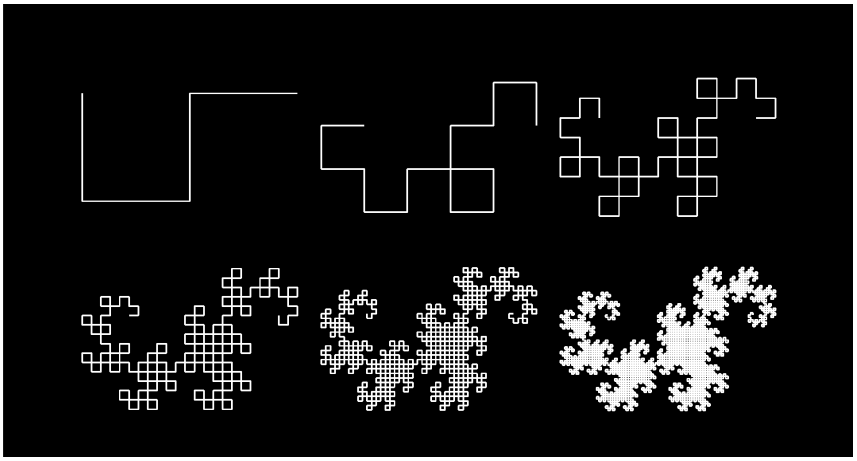

Figure 6: Dragon curves, generated by strings $\omega_2, \omega_4, \cdots, \omega_{12}$.

3.2.2 Case Study: Blood Vessels

Zamir (2001), meanwhile, uses parametric L-systems to visualise the bifurcation of blood vessels. Suppose a branch with length l, width w bifurcates into two branches M and m, such that $l_M \geq l_m$. Defining the *asymmetry ratio* $\alpha = l_m/l_M$, it follows that

$$l_M = \frac{l}{(1+\alpha^3)^{1/3}}, \quad l_m = \frac{\alpha \cdot l}{(1+\alpha^3)^{1/3}}, \quad w_M = \frac{w}{(1+\alpha^3)^{1/3}}, \quad w_m = \frac{\alpha \cdot w}{(1+\alpha^3)^{1/3}}.$$

Furthermore, the branches diverge from their parent at angles

$$\theta_M = \arccos\left(\frac{(1+\alpha^3)^{4/3} + 1 - \alpha^4}{2\,(1+\alpha^3)^{2/3}}\right), \quad \theta_m = \arccos\left(\frac{(1+\alpha^3)^{4/3} + \alpha^4 - 1}{2\alpha^2\,(1+\alpha^3)^{2/3}}\right).$$

Untitled's framework is therefore capable of reproducing Zamir's results (see Figure 7), using an L-system with the single production rule:

$$\mathbf{C}(l,w,\theta) \mapsto \mathbf{X}(l,w,\theta)[\mathbf{C}(l_M, w_M, \theta+\theta_M)]\mathbf{C}(l_m, w_m, \theta-\theta_m).$$

Figure 7: Zamir's model of arterial branching, with asymmetry ratios $\alpha = 1.0, 0.8, \cdots, 0.2$.

Liu et al. (2010) expand on this by introducing a stochastic component - that is to say, they allow [a more random structure]. *Untitled* incorporates such randomness into its own rules for blood vessels:

$$\begin{aligned}
\mathbf{C}(l,w,\theta) &\underset{0.4}{\longmapsto} \mathbf{X}(l,w,\theta)[\mathbf{L}(l_M, w_M, \theta+\theta_M)]\mathbf{R}(l_m, w_m, \theta-\theta_m) \\
\mathbf{C}(l,w,\theta) &\underset{0.4}{\longmapsto} \mathbf{X}(l,w,\theta)[\mathbf{L}(l_m, w_m, \theta+\theta_m)]\mathbf{R}(l_M, w_M, \theta-\theta_M) \\
\mathbf{C}(l,w,\theta) &\underset{0.2}{\longmapsto} \mathbf{X}(l,w,\theta)\mathbf{C}(l,w,\theta) \\
\\
\mathbf{L}(l,w,\theta) &\underset{1.0}{\longmapsto} \mathbf{X}(l,w,\theta)\mathbf{C}(l_M, w_M, \theta-\theta_M) \\
\\
\mathbf{R}(l,w,\theta) &\underset{1.0}{\longmapsto} \mathbf{X}(l,w,\theta)\mathbf{C}(l_M, w_M, \theta+\theta_M)
\end{aligned}$$

These describe a capillary with a 40% chance of bifurcating with branch M tacking clockwise, a 40% chance of bifurcating with M tacking anticlockwise, and a 20% chance of extending forwards without any branching. The determinstic production rules on **L**, **R** provide course correction, guaranteeing the [...]; further informal tweaks can be found in the `LBloodVessel` class, all intended to get the final look of the L-systems 'right' (see Figure [reference]).

[Figure of blood vessels in isolation]

[Discussion of animation (and the shortcomings thereof)...]

[Figure of final render]

3.3 Procedural Narrative

Untitled was originally conceived as a showcase of procedural text generation, an application of [authored X] towards interactive fiction.

3.3.1 Grammars

Given their origin in linguistics, it is perhaps unsurprising that grammars (see Section 3.2) find use in the field of procedural narrative - *Tracery* (Compton et al. 2015) being the prime example. From a mathematical perspective, this is a stochastic, context-free grammar, with uniformly-distributed production rules for each non-terminal; it iterates a given axiom until all non-terminals (demarcated by {`BRACE`} delimiters) have been replaced. To bemoan the generator as better suited to Twitter bots than immersive storytelling is to misunderstand its design. Compton et al. present a lightweight tool that is left *deliberately* narrow, intended for ease-of-use amongst even the most fledgling authors.

More substantial works, then, adapt *Tracery*'s grammar-based approach to their own specifications. In *The Annals of the Parrigues*, Short uses a consistent, context-sensitive generator that "defines any facts about the world that aren't already defined at the moment of generation" (2015, p. 83). *Voyageur* (Dias 2018) attaches weightings to the distributions of its production rules (Dias 2019). Produceral storytelling is an art form; as much as *Concrete Earth* wears the above influences on its sleeve, to try and consolidate these into an 'optimal', one-size-fits-all generator would be missing the point. This project therefore uses an implementation of *Tracery* with its own custom modifications.

Recency To reduce repetition of a word or phrase, one might consider when it has last been used. *Concrete Earth* treats this as a rank: given a non-terminal with N possible production rules, the most recently used rule will have a *recency*[6] of $N-1$, the second most recent $N-2$, and so on down to the rule that has been used longest ago (or indeed, never been used), with recency 0. Though it does not account for the intervals between uses, nor the second, third, etc, most recent use, this metric is already robust enough to achieve the desired effect (Kazemi 2019).

These ranks are straightforwardly integrating into the production rules' probability distribution. The likelihood of a rule with recency n is scaled by $p^{n/(N-1)}$, such that (all other weightings being equal) the most recently used rule will be p times as likely as the least recent.[7]

Consistency [Consistency of characterisation discussion...]

[Two cases...]

[6] Or 'dryness', as *Improv*, the generator for *Voyageur*, would call it.

[7] The same string `alpha` might replace two different non-terminals `#A#`, `#B#`. Recognising these as distinct production rules, the recency with which one was applied to `#A#` has no bearing on when the other is applied to `#B#`, or vice versa.

Nested Non-Terminals By their very nature, grammars struggle "to encode high level relationships between different things generated at different points" in a sentence (Compton 2019). [Explainer of *nested non-terminals*]. [Doesn't scale, but author-friendly enough when it only tracks a few key conditions].

Consider how *Concrete Earth*. Using a

```
{*{*GENDER} FORENAME} {*{*GENDER} PATRONYMIC} {*SURNAME}
```

Concrete Earth's grammar processes this as follows:

1. `{*{*GENDER} FORENAME}` ↦ `{*MASCULINE FORENAME}`*:*
2. `{*MASCULINE FORENAME}` ↦ `LEO`*:*
3. `{*{*GENDER} PATRONYMIC}` ↦ `{*MASCULINE PATRONYMIC}`*:*
4. `{*MASCULINE PATRONYMIC}` ↦ `NIKOLAYEVICH`*:* ("Nikolayevich" here meaning "Son of Nikolay") [8]
5. `{*SURNAME}` ↦ `TOLSTOY`*:* [No dependicies; saved]

Note that this

[EXAMPLE: Pronouns!]

This might feel too limited, from a dramatic perspective. For all this effort to [maintain consistency], personality is not so immutable; so much of good storytelling relies on character *development*. Indeed, how does the player feel like they have an impact, if [the grammar doesn't have permission to make changes]? *Concrete Earth* surely requires some sort of override...

3.3.2 Content Selection Architectures

Storylets (Kreminski & Wardrip-Fruin 2018) are [definition].

In *Concrete Earth*, these storylets are envisioned as the means for these [global changes]. [Discuss a beginning-middle-end structure].

The current implementation is limited. [Conditions: randomness]. [Effects: entering/exiting the party].

4 Code Organisation

Taking a step back, [...].

Section 3 has been structured [to reflect the two fundamental stages of graphics programming: first implementing frameworks (marching cubes, L-systems), then building tangible objects out of them (terrain hexes, blood vessels)]. *Concrete Earth*'s code is structured accordingly. [Using inheritance...]. [Crucially, little is left exposed in the main *Game.cpp* file...].

4.1 Rendering

4.2 GUI

[Include HDRR/bloom here...]

[8] [Non-binary...]

5 Evaluation

5.1 Features

If there is one element of this project that represents an unambiguous success, it is the L-systems. [...]. Not that there isn't more to do - the procedural animation of the blood vessels, for instance - but . [An *open-ended* tool, to be carried into future projects].

The role of marching cubes is much more utilitarian: height maps have a well-known limitation, to which this algorithm is the well-known solution (see Section 3.1). Implementing the code was a significant technical challenge, but largely successful. While , it's worth noting that [solutions/next steps exist]. Bourke also provides an introduction to *marching tetrahedrons* (1997), a variant of marching cubes that would hopefully improve efficiency.[9] Though a tetrahedral grid into [additional nuances], [having completed the basic algorithm].

The more interesting question here is how effectively the marching cubes enhance *Concrete Earth*'s setting. As much as they are used to procedurally generate terrain, that terrain feel somewhat lacking: while this report has already acknowledged [few landmarks], it's no less notable that [untextured]. That the game logic [generates at runtime and cycles through] only enhances (though this is a much quicker fix).

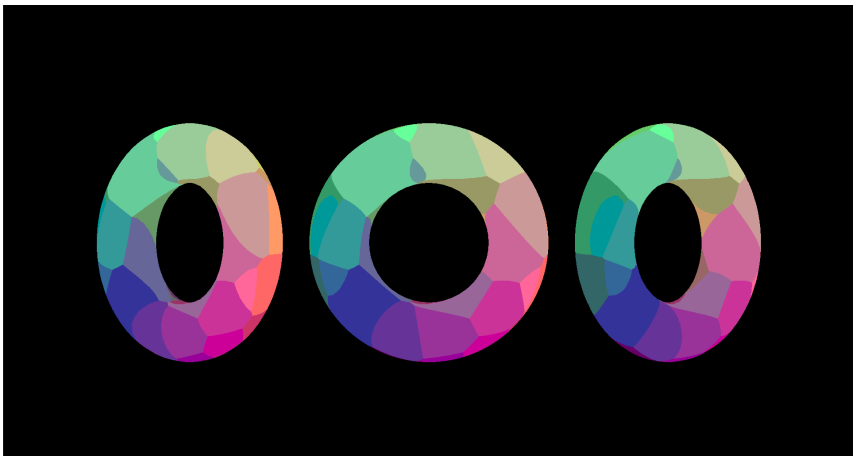

Figure 8: 3D Voronoi noise. While voxel textures [would have...]

To the extent that *Concrete Earth* has any single 'special feature', though, it would surely be its approach to procedural storytelling. [...]

5.2 Code Organisation

Built on previous work from CMP502, this project could be written to an already-established style. Functions and variables follow a consistent naming convention, with a strong focus on readability. Inheritance has helped reduce redundancies. Even the current, informal use of comments proves effective, these breaking down the logic behind key snippets, highlighting potential bugs, and overall acting as valuable memos-to-self (a collaborative project would obviously require a higher standard of documentation). In this very granular sense, *Concrete Earth* is a perfectly acceptable piece of code.

[9] [$2^4 = 16$...]

[Stripping back shaders... an effective strategy!]

[Position weakness of the storytelling as an *organisational* matter...]

[Discuss the broader merits of Section 4, ultimately landing on the value of *internal consistency*? Or - contrast narrative with models?]

6 Conclusions

From a more personal perspective, I can't help but feel that *Concrete Earth* shows promise.

[Coheres in a way that my CMP502 project absolutely didn't...]. [Furthermore, a better sense of performance] - I might not be happy with the loading times *per se*, but all things considered these advanced graphics techniques will still run on my laptop.

Of course, there is a distinction to be drawn between *Concrete Earth*'s functionality as a graphics showcase and as a game. As much as there's a certain level of interactivity on display,[10] [Perils of interactive narrative].

[What/how would I go about cannibalising this? Screen shader first/hexes... narrative much more an early experiment in structuring content selection architectures/context-sensitive grammars...]

[3rd last: iterating marching cubes, to build intuition... justify own rundown...]

[2nd last: iterating everything...]

[Last: framework does not exist in isolation - tie back to CMP502 to show potential! Shaders being stripped back is a great example...]

References

Bourke, P. (1994), 'Polygonising a Scalar Field', Available at: `http://paulbourke.net/geometry/polygonise/`. (Accessed: 9 February 2023).

Bourke, P. (1997), 'Polygonising a Scalar Field Using Tetrahedrons', Available at: `http://paulbourke.net/geometry/polygonise/`. (Accessed: 9 February 2023).

Chomsky, N. (1956), 'Three Models for the Description of Language', *IRE Transactions on Information Theory* **2**(3), 113–124.

Compton, K. (2019), 'Getting Started with Generators', *in* T. Short & T. Adams, eds, '*Procedural Storytelling in Game Design*', 2nd edn, Boca Raton, FL, USA: Taylor & Francis, pp. 3–17.

Compton, K., Kybartas, B. & Mateas, M. (2015), Tracery: An Author-Focused Generative Text Tool, *in* '*8th International Conference on Interactive Digital Storytelling*', Copenhagen, Denmark: 30 November-4 December, pp. 154–161.

Dias, B. (2018), *Voyageur*, self-published.

Dias, B. (2019), 'Procedural Descriptions in *Voyageur*', *in* T. Short & T. Adams, eds, '*Procedural Storytelling in Game Design*', 2nd edn, Boca Raton, FL, USA: Taylor & Francis, pp. 193–207.

Hanan, J. S. (1992), Parametric L-systems and Their Application to the Modelling and Visualization of Plants, PhD thesis, University of Regina, Regina.

[10] And likewise, non-interactive features like the fixed, orthographic camera carry some level of intentionality...

Hopcroft, J., Motwani, R. & Ullman, J. D. (2000), *Introduction to Automata Theory, Languages, and Computation*, 2nd edn, Boston, MA, USA: Addison-Wesley.

Kazemi, D. (2019), 'Keeping Procedural Generation Simple', *in* T. Short & T. Adams, eds, '*Procedural Storytelling in Game Design*', 2nd edn, Boca Raton, FL, USA: Taylor & Francis, pp. 17–22.

Kreminski, M. & Wardrip-Fruin, N. (2018), Sketching a Map of the Storylets Design Space, *in* '*11th International Conference on Interactive Digital Storytelling*', Dublin, Ireland: 5-8 December, pp. 160–164.

Lindenmayer, A. (1968), 'Mathematical Models for Cellular Interactions in Development II. Simple and Branching Filaments With Two-Sided Inputs', *Journal of Theoretical Biology* **18**(3), 300–315.

Liu, X., Liu, H., Hao, A. & Zhao, Q. (2010), Simulation of Blood Vessels for Surgery Simulators, *in* '*2010 International Conference on Machine Vision and Human-machine Interface*', pp. 377–380.

Prusinkiewicz, P. & Lindenmayer, A. (1996), *The Algorithmic Beauty of Plants*, 2nd edn, Berlin, Germany: Springer-Verlag.

Short, E. (2015), *The Annals of the Parrigues*, self-published.

Trauth, K., Hora, S. & Guzowski, R. (1993), Expert Judgment on Markers to Deter Inadvertent Human Intrusion Into the Waste Isolation Pilot Plant, Technical report, SANDIA National Laboratories, Albuquerque.

Zamir, M. (2001), 'Arterial Branching Within the Confines of Fractal L-System Formalism', *The Journal of General Physiology* **118**, 267–276.