Concrete Earth: A DirectX Game

Sam Drysdale

May 16, 2023

Contents

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).

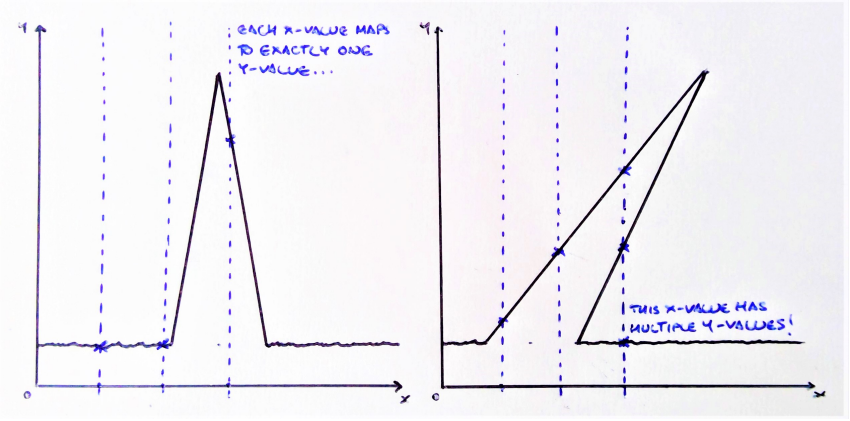

Figure 1: Two sketches of 'thorny' terrain; the former can be generated by a height map, whereas the latter is too concave.

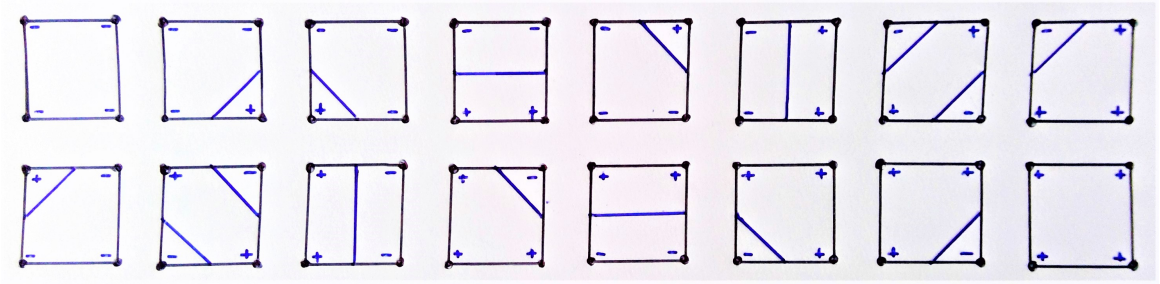

Figure 2: Linear partitions of a marching square. With 4 gridpoints, each existing in one of 2 states ('+' for $f_{i,j} \geqslant c$, '−' for $f_{i,j} < c$), exactly $2^4 = 16$ partitions are possible.

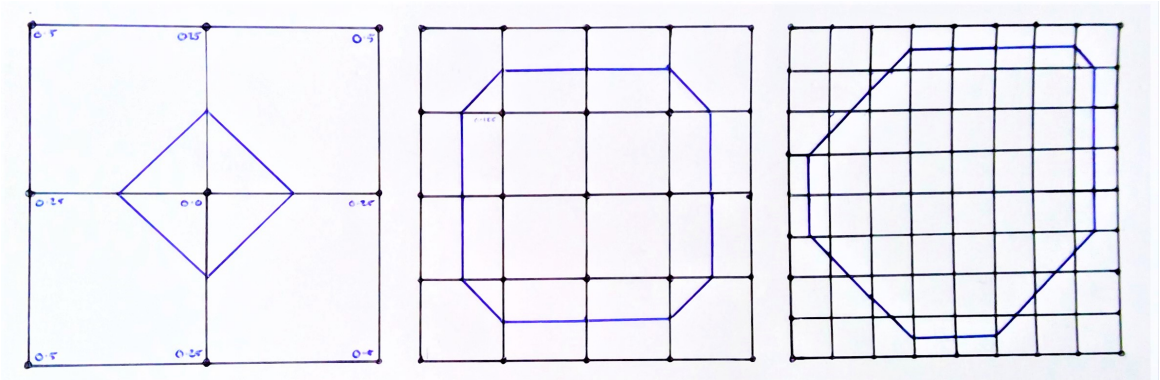

Figure 3: Bounding curve $f(x, y) = 0.16$, at increasing levels of granularity.

Take the two-dimensional analogue of this problem: how does one construct the bounding curve of a (potentially concave, or even disconnected) area? Consider an $n \times n$ square grid, with $n \in \mathbb{N}$,[2] and let a 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 $n \to \infty$.

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. Figure 3, for example, illustrates how it might apply to

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

a scalar field describing two overlapping circles.

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 - but if the partition is to represent an isoline, then it surely requires $f(0.5\mathbf{a} + 0.5\mathbf{b}) \approx c$, 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

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

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 5).

[1] While the framework registers inputs on press, note that options are only selected *on release*.
[2] This could equally be an $n \times m$ grid; keeping the dimensions the same is just neater.
[3] Informally, a 'contour line' of the $2D$ field.

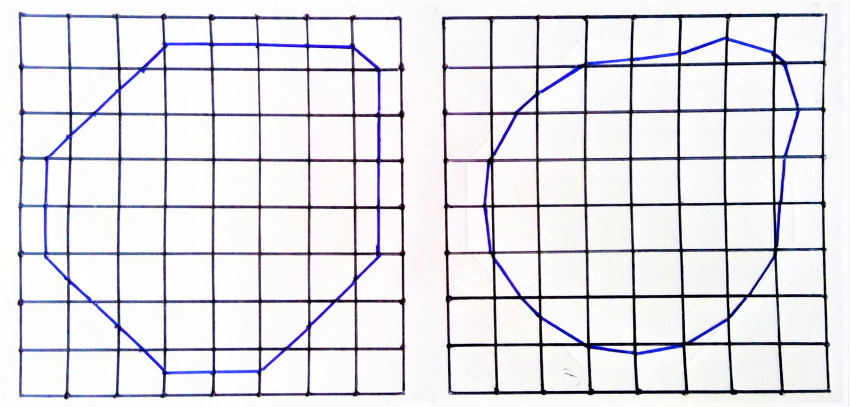

Figure 4: Bounding curve $f(x, y) = 0.16$, before and after linear intepolation.

Returning to the problem at hand, *Concrete Earth* uses *marching cubes* to generate the *isosurfaces* bounding 3D volumes. This requires an $n \times n \times n$ grid, and a new field $f : [0, 1]^3 \to \mathbb{R}$, the algorithm now filling cubic cells with planar surfaces to partition the gridpoints (i, j, k) satisfying $f_{i,j,k} < c$ (Again using linear interpolation to best approximate the 'true' isosurface). The underlying principle remains. Since 3D fields are hard to visualise in and of themself, marching cubes are best understood in relation to marching squares; despite any superficial differences, the intuition that has been built up above is readily generalised to higher dimensions.

This section will not cover the finer points of the DirectX implementation - not how it deconstructs partitioning surfaces into triangles for the vertex/index buffers, nor how it identifies adjacent marching cubes so as to calculate 'smooth' vertex normals[4]. The rest of the report will only require a high-level understanding of *Concrete Earth*'s terrain. Paul Bourke's *Polygonising a Scalar Field* (1994) remains the definitive source on the marching cubes.

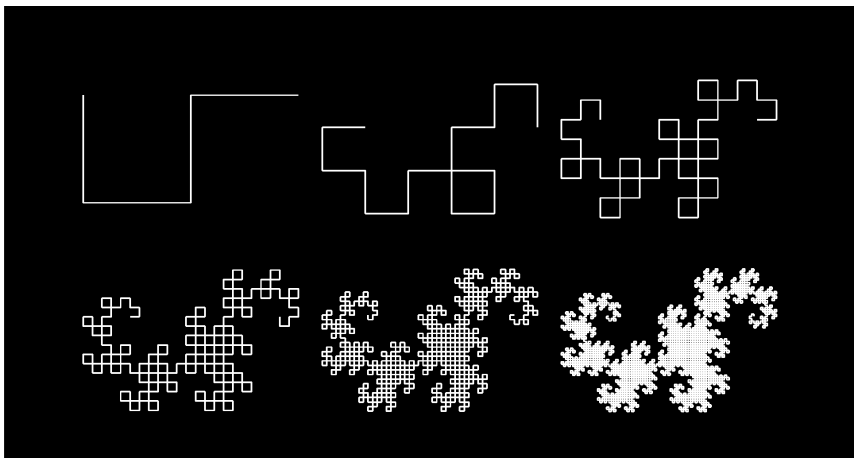

Figure 5: Planar partitions of a marching cube, before linear interpolation. There are 15 fundamental cases, ignoring symmetries; the code itself considers all $2^8 = 256$ possible configurations.

[4] A weighted average over the surface normals of the surrounding triangular faces; if a face has vertices $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, it will be weighted by $1/|(\mathbf{b} - \mathbf{a}) \times (\mathbf{c} - \mathbf{a})| \propto$ the inverse of its area

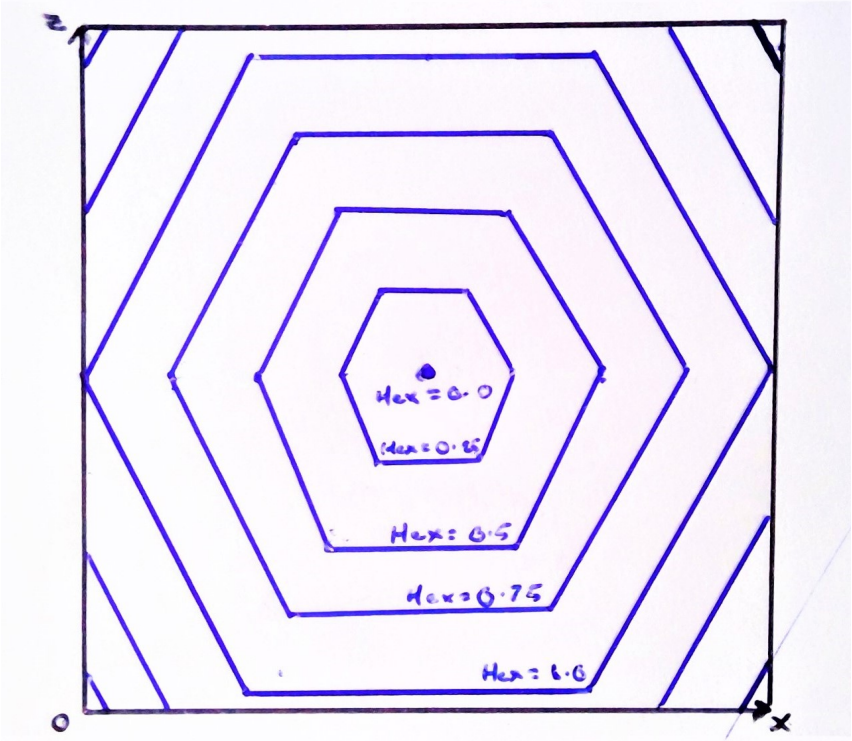

Figure 6: Isolines of $\mathbf{Hex}(x, y_0, z)$, a horizontal slice of the hexagonal field.

3.1.1 Case Study: Hexes

Indeed, having established this framework, generating a landscape becomes a matter of writing the right scalar field. [Barren - stark, even! - land, so start on a flat foundation...].

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

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

The first significant [...].

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

[...] a vertical hexagonal prism

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

[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)...]

3.1.2 Case Study: Landmarks

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

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

Much like before,

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

noting

Crucially, this . Moreover, any number of thorns can be added; taking the minimum over more than two fields will only expanded the isosurface further. The only surprising choice *Concrete Earth* makes in procedurally generating these landmarks is keeping their surfaces completely smooth. While it would absolutely be , the entire purpose of these [X] is that these strange, idealised geometries should feel distinctly unnatural. The physical markers of shunned land are man-made designs that *should* tap into that sense of the uncanny, in the hopes of communicating to future civilisations that 'this is not a place of honour'.

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

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

While these definitions 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 the sentences have been generated recursively by a grammar, the line drawings will inherit their self-similar structure. Consider the above example - reading non-terminals F, G as 'draw a line while moving 1 unit forwards,' and terminals $\pm$ as 'turn by $\pm\pi/2$ on the spot,' produces fractal 'dragon curves' (see Figure 7).

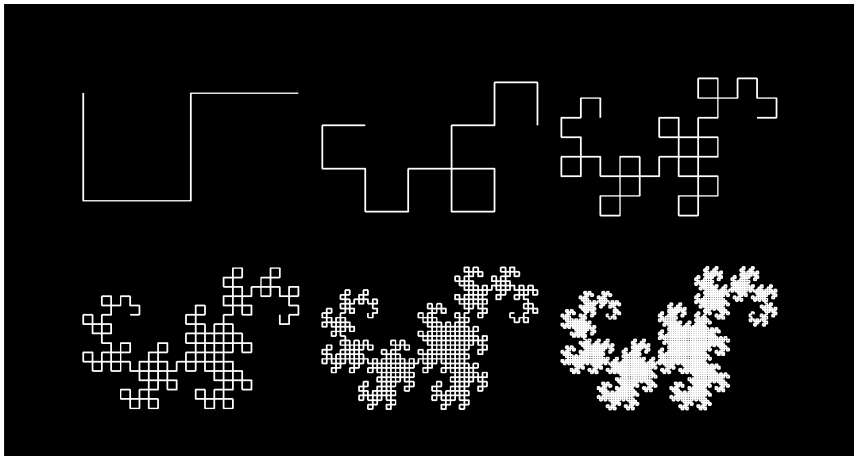

Figure 7: Dragon curves, generated by strings $\omega_2, \omega_4, \cdots, \omega_{12}$.

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

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

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!].

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

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

Furthermore, the branches diverge from their parent at angles

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

The established framework is therefore capable of reproducing Zamir's results (see Figure 8), using an L-system with the single production rule:

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

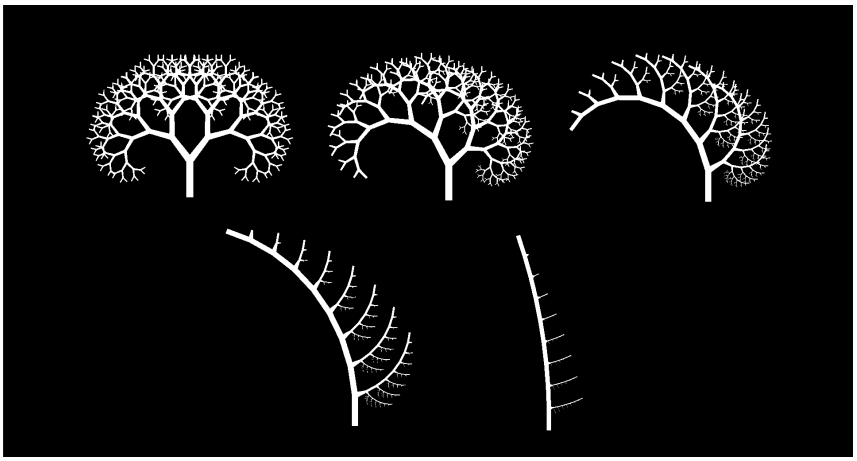

Figure 8: Zamir's model of arterial branching, as generated by asymmetry ratios $\alpha = 1.0, 0.8, \cdots, 0.2$. This is an explicit copy of Zamir (2001, Figure 8), recreated within the DirectX framework.

Liu et al. (2010) expand on this research with a stochastic component - they assign *multiple* production rules to a single non-terminal, choosing which to apply using a weighted probability

distribution. *Concrete Earth* incorporates such randomness into its own rules for blood vessels:

$$\begin{aligned}
\mathbf{C}(l,w,\theta) &\xrightarrow[0.45]{} \mathbf{X}(l,w,\theta)[\mathbf{L}(l_M,w_M,\theta+\theta_M)]\mathbf{R}(l_m,w_m,\theta-\theta_m)\\
\mathbf{C}(l,w,\theta) &\xrightarrow[0.45]{} \mathbf{X}(l,w,\theta)[\mathbf{L}(l_m,w_m,\theta+\theta_m)]\mathbf{R}(l_M,w_M,\theta-\theta_M)\\
\mathbf{C}(l,w,\theta) &\xrightarrow[0.10]{} \mathbf{X}(l,w,\theta)\mathbf{C}(l,w,\theta)\\
\\
\mathbf{L}(l,w,\theta) &\xrightarrow[1.00]{} \mathbf{X}(l,w,\theta)\mathbf{C}(l_M,w_M,\theta-\theta_M)\\
\\
\mathbf{R}(l,w,\theta) &\xrightarrow[1.00]{} \mathbf{X}(l,w,\theta)\mathbf{C}(l_M,w_M,\theta+\theta_M)
\end{aligned}$$

These describe a capillary with a 45% chance of bifurcating with branch M tacking clockwise, a 45% chance of bifurcating with M tacking anticlockwise, and a 10% chance of extending forwards without any branching. The deterministic production rules on **L**, **R** provide course correction, ensuring the models can be pointed towards the centre of the screen without veering off in either direction. Further informal tweaks can be found in the `LBloodVessel` class, all intended to get the final look of the blood vessels exactly right (see Figure 9).

Figure 9: *Concrete Earth*'s stochastic model of arterial branching, as generated by different seeds.

[Discussion of animation (and the shortcomings thereof)...] Finally, note that all L-systems have been implemented with a normalised `intensity` factor, which guides their growth according to

```
[(SIMPLIFIED) CODE SNIPPET]
```

As `intensity` increases from `0.0f` to `1.0f`, , the [X] component delaying the minor branches to .[6] Rendering these to texture results in a visceral, procedurally-animated post-process effect, more responsive than passing in a static alpha map (see Figure 10).

3.3 Procedural Narrative

Concrete Earth was originally conceived as a showcase of procedural text generation, an application of [authored X] towards interactive fiction. Since [not in place], this section will focus on the

[6] This was completely unintentional, emerging from a typo in the code -

Figure 10: (High-contrast render of) *Concrete Earth*'s eye strain effect.

intended narrative, and the infrastructure *already in place* to facilitate it.

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 a prime example. This is, formally, 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 interactive 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. The custom grammar treats this as a rank: given a non-terminal with N possible production rules, the most recently used rule will have a *recency*[7] of $N-1$, the second most recent $N-2$, and so on down to the rule that was used longest ago (or indeed, never used), with recency 0. Though it does not account for the intervals between uses, nor the second, third, etc, most recent use, there isn't any reason to overcomplicate a metric that already achieves 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.[8]

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

[8] 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.

Consistency Intended to create a feeling of transience, landmarks in *Concrete Earth* can be visited once and once only, and as such do not need tracked by an overall 'world model'. The game's approach to characterisation is not so simple. With a narrative largely centred around picking up passengers, journeying with them, then parting ways, it should follow that if a detail about a party member has already been established, the text generator must be able to consistently recover it.

The grammar's `GenerateSentence` function, therefore, accepts (up to) two pointers to `StoryCharacter` structs. These are handled by three *consistency markers*:

- `*`*:* The term `{*A}` tells the grammar to check the first character argument for key `A`. If a value exist, this will be substituted [If yes; if else], and stored in the character struct for future use.
- `**`*:* The term `{**A}` similarly applies to the second character argument.
- `?`*:* .[9]

Conditionality 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}`*:* The grammar starts with the nested `{*GENDER}` non-terminal. The asterisk tells it to use the character's established gender, but - finding nothing - it randomly decides whether they are a man, woman, or non-binary, and passes this same value into the sentence and the `StoryCharacter` struct.
2. `{*MASCULINE FORENAME}` ↦ `LEO`*:* ; even if [never made explicit], the internal consistency
3. `{*{*GENDER} PATRONYMIC}` ↦ `{*MASCULINE PATRONYMIC}`*:* Now that their gender has been established, the grammar now directly substitutes in the value from the `StoryCharacter` struct.
4. `{*MASCULINE PATRONYMIC}` ↦ `NIKOLAYEVICH`*:* The character's forename is matched to a patronymic of the same gender ("Nikolayevich" here meaning "Son of Nikolay") [10]
5. `{*SURNAME}` ↦ `TOLSTOY`*:* The final non-terminal comes with no conditions, and ; this, too, is saved within the

In this case, the grammar terminates after a single iteration. Should `{*BARREL #1}-{*BARREL #2}` (signifying a double-barrelled surname) have replaced `{*SURNAME}`, however, the new non-terminals would needed handled on a second parse.

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...

[9] Note that `?` can also inherit an empty string, if the preceding non-terminal is not being remembered.

[10] As far as this report can tell, there are no gender-neutral patronymics in Russian - without actually speaking the language, it seems most appropriate to drop these from non-binary characters' names entirely. [Ethical generation - what a generator might accidentally say about the worlds it creates (citation); as much as (...medieval society would be engaged in non-binary erasure on a systemic level?)]

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, [...]. [`Game.cpp` manages three core components: gameplay, UI, and rendering]

Gameplay `Board` [...explain the loop].

[`StoryEngine`].

[Corpora are stored in `.JSON` files. Accessed by `nlohmann::json` API].

UI [All implemented with ImGui. Simple, stored in `Game.cpp` itself. This feels appropriate... because of inputs?]

Rendering Indeed, 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...].

[Furthermore - shaders!]

[Conclude with existing 5.2 paragraph? Cleaner, more self-contained!]

4.1 Rendering

4.2 GUI

[Include HDRR/bloom here...]

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.[11] Though a tetrahedral grid into [additional nuances], [having completed the basic algorithm].

[11] [$2^4 = 16$...]

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 the relatively few landmarks available, it's no less notable that they are left untextured. With more time, , but having already experimented with procedural texturing in CMP502 it felt more fruitful to explore new techniques. That the game logic [generates at runtime and cycles through] only enhances (though this is a much quicker fix).

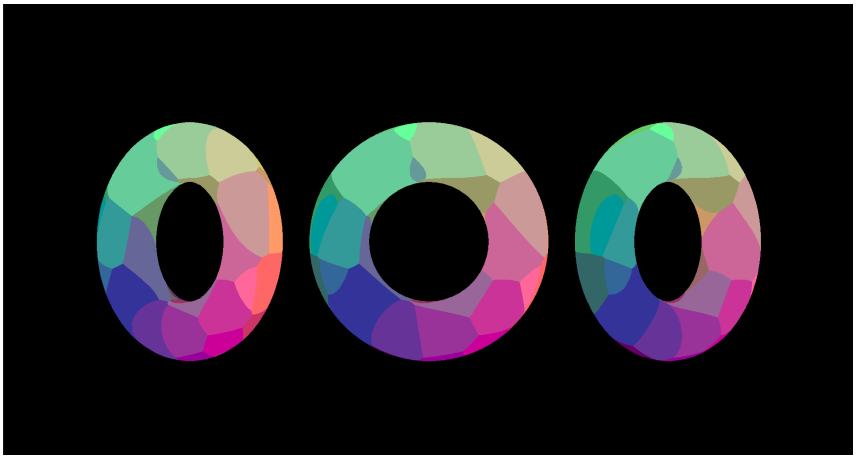

Figure 11: 3D Voronoi noise. While voxel textures [would have...]

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.

[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?]

[Holistically, though, the code *must* be a success.]

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,[12] [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.

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.

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

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.