Untitled: A DirectX Game

Sam Drysdale

May 16, 2023

Contents

1 Summary

2 User Controls

3 Features

The following is a technical discussion of the key features of *Untitled*, with a focus on 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 Noise

3.2 Procedural Terrain

Given its grounding in nuclear semiotics, 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 (x, z)-coordinate can correspond to only one y-value prevents any features that ominously overhang *Untitled*'s barren earth (see Figure [reference]).

[Hand-drawn figure, demonstrating the shortcomings of thorns].

Marching cubes are therefore central to this modelling process. The concept here is best introduced in two dimensions.

In much the same way, [3D generalisation].

[These are just the broad strokes; nuance to, say, weighing vertex normals correctly]. Definitive... Paul Bourke's *Polygonising a Scalar Field* (1994)... [mention marching tetrahedra? While X, marching cubes have been perfectly adequate for the purposes set out below...].

3.2.1 Case Study: Hexes

3.2.2 Case Study: Landmarks

3.3 Procedural Screen Textures

The post-processing in *Untitled* is, in one sense, rather simple. The 'stress vignette,' for instance, calls only two renders-to-texture on every frame: the board itself, and an alpha map of blood vessels that sprout from the edges of the screen. As striking as the final effect is, `vignette_ps.hlsl` is surprisingly straightforward in blending the textures into a final, pulsing eye strain overlay; far more deserving 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 $A, B, \dots \in N$, and terminals $a, b, \dots \in \Sigma$. The production rules in P map nonterminals to strings $\alpha, \beta, \dots \in (N \cup \Sigma)^*$; applied recursively to the axiom $\omega_0 \in (N \cup \Sigma)^*$, these rules can produce increasingly complex *sentences* of terminals and/or nonterminals.[1]

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

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

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

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. In the above example, interpreting 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 1.

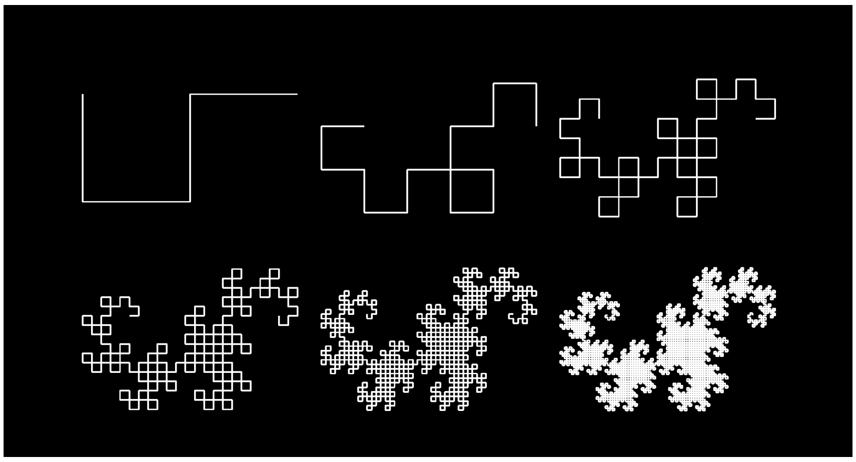
Figure 1: Dragon curves, generated by strings $\omega_2, \omega_4, \cdots, \omega_{12}$.

While *Untitled* only needs them to generate 2D alpha maps, note that L-systems are most common in the modelling of 3D plants and other branching structures (Prusinkiewicz & Lindenmayer 1996). Furthermore, this report will restrict its attention to L-systems paired with context-free grammars.

3.3.1 Case Study: Runes

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

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

[Example: various geometric runes!].

3.3.2 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

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

Furthermore, the branches diverge from their parent at angles

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

Figure 2: Zamir's model of arterial branching, with asymmetry ratios $\alpha = 1.0, 0.8, \cdots, 0.2$.

Untitled's framework is therefore capable of reproducing Zamir's results (see Figure 2), using an L-system with the single production rule:

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

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:

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

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]

3.4 Procedural Narrative

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

3.4.1 Grammars

Given their origin in linguistics, it is perhaps unsurprising that grammars (see Section 3.3) have found much use in the field of procedural narrative - *Tracery* (Compton et al. 2015) being the prime example. In mathematical terms, it is a stochastic, context-free grammar, with uniformly-distributed production rules for each non-terminal; the tool iterates a given axiom until all such non-terminals (demarcated by `#HASHTAG#` delimiters) have been replaced. This generation process is streamlined by design, Compton et al.'s intention

While *Tracery*'s lightweight, author-focused, the trade-off is [memoryless!]. [Short; *Improv*] In *Annals of the Parrigues*, sets out the need for a *consistent* generator, one that "defines any facts about the world that aren't already defined at the moment of generation" (, p. 83). *Voyageur* , meanwhile, integrates several weightings into distribution of production rules (Dias 2019). While [*Untitled*] - the game uses an implementation of *Tracery* with a couple of custom modifications...

Recency [Or 'dryness'].

3.4.2 Content Selection Architectures

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

[Though conceptually no different, ... , our 'narrative stack'...]

4 Code Organisation

4.1 Post-Processing

4.2 GUI

[Include HDRR/bloom here...]

5 Evaluation

5.1 Features

5.2 Code Organisation

6 Conclusions

[Coheres in a way that my CMP502 project absolutely didn't...]

Of course, there is a distinction to be drawn between *Untitled*'s functionality as an interactive experience and as a game. [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...]

References

Bourke, P. (1994), 'Polygonising a Scalar Field', 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., 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. (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.

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*, 2$^{\text{nd}}$ edn, Berlin, Germany: Springer-Verlag.

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.