Forging the River of The Flame in the Flood

Damián Isla, The Molasses Flood
The Flame in the Flood

- Third-person survival roguelike
- River journey through a post-societal American South

In other words:

- Don’t Starve meets Journey
- Huckleberry Finn meets The Road
- Oregon Trail meets Toobin’
THE GAME STRUCTURE - "GO WITH THE FLOW"

"TOOBIN' MEETS OREGON TRAIL"
https://youtu.be/64HnlW4HcD4
The River

The River is the central character

The shape and nature of the river defines gameplay

- Hour-to-hour
- Minute-to-minute
- Second-to-second
Our Art Director is super picky.

- Every individual piece of content is lovingly hand-crafted.
- Perlin landscapes etc. will not cut it.

Most of our techniques involve the arrangement of pre-made elements

- Very little generation at the geometric or textural level.
- (Except water rendering)
The 30,000’ View

- Layout
  - Composition
    - Optimization
      - Flow Generation
        - Decoration and Rendering

Disclaimer: Very little on this topic
I. Layout

Establish the high-level shape of the river

In our world, the river is:

- A Sequence of Biomes
- Within a Biome, a sequence of River Sections
Biome 1

A:
Width: ~50m
Length: ~100m
Length: 25km
Sequence: AAB

B:
Width: ~75m
Length: ~130m

Biome 2

A:
Width: ~50m
Length: ~100m
Length: 30km
Sequence: AAABCB

B:
Width: ~50m
Length: ~130m

C:
Width: ~150m
Length: ~160m
River Sections

Designer Parameters

- Dimensions
- Density
- Visitable location count
- Rapids/No rapids
- Flow speed scale
- Etc.
Corridors and Island Clusters

- Corridors
  - Narrow
  - No islands
  - Faster water flow

- Island Clusters
  - Wide
  - Lots of islands, lots of choices

- Creates the “ebb & flow” game pace
II. Composition

What goes into a River Section?

2 Answers:

- Landmass geometry
  - Islands and Shorelines
- Resource locations
  - Church, gas station, campground, etc.
Landmasses

Section density (landmass / km$^2$)
Resource Locations

Landmarks
- Recognizable
- Specific resources
- Game-balance sensitive
- 3-5 per section
- Max 1 per type per section

What is the best way for designers to set spawn frequency?
Resource Locations

Scheme 1: Weighted Sampling

- For each section chance of spawning a church is $p_{church}$
- Average interval: $\frac{1}{p_{church}}$
- Max interval: $\infty$
Resource Locations

Scheme 2: Scheduling

- Specify Min, Average and Max interval

- “The Church will appear about every 2-3 sections, but never more than 5.”
Resource Locations

Scheme 2: Scheduling

- Triangular Distribution
- Specify Min, Mode, Max
- Easily sampled
Resource Locations

Scheme 2: Scheduling

Same technique used for scheduling at the island level:

- Secondary locations
- Vegetation
- Loot
- Enemy AI
III. Optimization

Arrange elements chosen by Composition

- Including River Edges, Resource Locations

Islands are

- Hand-authored, with many variants
- Irregularly shaped
- Of varying scales

The Challenge

- Make them fit together pleasingly
- Make the result look organic
- Don’t make the boundaries obvious
- Create a sense of place
Optimization

1. Start with “Best guess” state
2. Evaluate using **Objective Function**
3. Iteratively modify state to
   
   Increase amount of goodness or
   Decrease the amount of badness

We use **Gradient Descent + Random Restarts**
“Best Guess” Start

- River Edges placed alongside River Section edges
- Resource locations distributed in pairs
  - Create as many “can’t visit both” situations as possible
- The rest are uniformly randomly distributed
For us: given a state vector

\[ X = \langle x_1, y_1, \theta_1, x_2, y_2, \theta_2 \ldots x_n, y_n, \theta_n \rangle \]

Generate an error value penalizing violations of constraints

- Match underlying River Section geometry
- Stitch together shorelines
- Don’t overlap islands

\[ E(X) = E_{Geom}(X) + E_{shore}(X) + E_{overlap}(X) \]
Overlap Error
Overlap Error
Overlap Error

Desired margin $M$
Sphere-to-sphere Error = $E_r(\text{overlap})$
Island-to-Island Error = sum of all Sphere-to-Sphere errors
Total overlap error = sum of all Island-to-Island errors
Overlap Error

\[ d = r_a + r_b + M - \text{Dist}(S_a, S_b) \]

How far we Want to be
How far we are
Potential Error Functions

\[ d = r_a + r_b + M - \text{Dist}(S_a, S_b) \]

\[ E_r(d) = \begin{cases} 
  d^2, & d > 0 \\
  0, & d \leq 0 
\end{cases} \]
Gradient Descent

So we have:

- State: \( X = \langle x_1, y_1, \theta_1, x_2, y_2, \theta_2 \ldots x_n, y_n, \theta_n \rangle \)
- Error function: \( E(X) \)

For Gradient Descent we also need:

- Error gradient: \( \nabla E(X) = \left( \frac{\partial E(X)}{\partial x_1}, \frac{\partial E(X)}{\partial y_1}, \frac{\partial E(X)}{\partial \theta_1}, \ldots, \frac{\partial E(X)}{\partial x_n}, \frac{\partial E(X)}{\partial y_n}, \frac{\partial E(X)}{\partial \theta_n} \right) \)

Algorithm:

\[
X_{t+1} = X_t - \alpha_t \nabla E(X_t)
\]
\[ \partial E_r / \partial \theta \]

\[ E(\theta) = U(\theta) + M \]

\[ E(\theta) = \frac{1}{2} D + M \]

\[ \theta = x_b - L \cos \theta + y_b - L \sin \theta \]

\[ E(\theta) = \frac{1}{2} D = \frac{1}{2} U \]

\[ \frac{\partial E}{\partial D} = \frac{\partial E}{\partial U} \]

\[ \frac{\partial E}{\partial \theta} = 2D - \frac{1}{2} U \]

\[ \frac{\partial E}{\partial y_b} = 2L \]

\[ \frac{\partial E}{\partial x_b} \]

\[ Avert Your Eyes \]

(See Appendix A)
Auto-Chad
Local Minima
Random Restarts

Calculate “Critical Error”

- After GD, randomize islands with Critical Errors
- Run GD again
- Keep track of our best state found so far
- Eventually, start deleting non-essential islands
Flow Generation

Decoration and Rendering
IV. Flow Generation

Interested in:
- Support the fiction
- Make navigation challenging
- Create the second-to-second arcade experience

Not interested in
- Hyper-realistic fluid simulation
- Cool to look at, but not predictable for player
- Besides severe performance limitations

What we’re looking for is really “Cartoon Fluid Dynamics”
Flow Volumes

- Self-contained 2d areas of water flow
- Flow is generated within a volume
- Blend flow at boundaries
Flow Map

Square grid of velocity vectors

- At runtime, use bilinear interp to generate smooth values
First step is to rasterize the interior of the flow volume.

- Turn it into a simple grid, where cells are either solid or empty.
- We cast a whole bunch of rays over several timesteps.
The Iso Map

- A grid of scalar “iso-values”
- A constant iso-value = a line of flow
- Choose iso-values to produce well-behaved, predictable flow

Note: Geometric, not simulation-based technique
  - Nothing particularly principled here
  - E.g. wrt pressure/speed/flow volume, no Navier-Stokes
Iso Maps w/o Islands

Generate 2 Distance Maps using Dijkstra

- Distance-to-left-shore, $D_{\text{left}}(x)$
- Distance-to-right-shore, $D_{\text{right}}(x)$

Iso-value:

$$I(x) = \frac{D_{\text{left}}(x)}{D_{\text{left}}(x) + D_{\text{right}}(x)}$$
Iso Maps With Islands

Iteratively blend in the influence of individual islands

Go from largest to smallest

- Where “large” means “widest with respect to flow”
I is unaffected
\[ I'(x) = \left(1 - \frac{D(x)}{D_{\text{max}}}ight) I^* + \frac{D(x)}{D_{\text{max}}} I(x) \]
Iso Map $\rightarrow$ Flow Map

- Calculate $\nabla I(x)$
  - Generate gradient vector by fitting a plane to 5x5 grid of iso-values using Least Squares
  - See Appendix B
Calculate $\nabla I(x)$
- Generate gradient vector by fitting a plane to 5x5 grid of iso-values using Least Squares
- See Appendix B

- Rotate 90 degrees

- Blur

- Scale velocity as you please
  - E.g. as a function of distance-to-shore
[Flow generation demo]
Decoration and Rendering

Flow Generation

Optimization

Composition

Layout
V. Decoration and Rendering

- Randomize island geometry
- Effects
  - Light shafts
  - Splashes, water streams
- Generate Water geometry
  - Highly-tessellated flat mesh following River layout
  - Undulation added in vertex shader
- Water Shader

*Water Flow in Portal 2*
Vlachos, Siggraph 2010
Putting it All Together
Putting it all together

Layout

Composition

Optimization

Flow Generation

Decoration and Rendering
Putting it all together

- Layout
- Composition
- Optimization
- Flow Generation
- Decoration and Rendering

- Hour-to-hour
- Minute-to-minute
- Second-to-second
2 Major Strategies

- Parallelization
  - Flow generation

- Timeslicing
  - Composition: spawn max N islands per tick
  - GD: one iteration per tick
  - Water surface: extend geometry
Staging

**Composition + Optimization**: desired at 5km, required at 3km

**Flow Gen**: desired at 3km, required at 1km

**Water Surface**: desired at 1km, required at 300m
Summary
Procedural Content Generation is a BIG DEAL these days

TFITF is a special case: linear, unidirectional structure makes it amenable to schedule- and sequence-based approaches.

Less this

More this
Takeaways - Technical

- Schedule- and histogram-based spawning

- Gradient Descent with Restarts:
  - Distribute irregular shapes according to a set of loose constraints
  - Useful for arrangement-of-handmade-parts situation

- “Cartoon Fluid Dynamics”
Takeaways - Architectural

Generalizable PCG Framework?

- Layout
- Composition
- Optimization
- Decoration

LCOD
Thank you!

Slides at naimadgames.com/publications.html

@damian_isla
@molassesflood
Appendix A
Computing the Error Gradient
\[ \frac{\partial E_r}{\partial \theta} \]

\[ E(\theta) = E_r(r_a + r_b + M - Dist(\theta)) \]

Sphere B location:
\((x_b, y_b)\)

Sphere A location:
\((L \cos \theta, L \sin \theta)\)

Distance-squared(A, B):
\[ U(\theta) = (x_b - L \cos \theta)^2 + (y_b - L \sin \theta)^2 \]
\[ \frac{\partial E_r}{\partial \theta} \]

\[ E(\theta) = E_r(r_a + r_b + M - Dist(\theta)) \]

\[ U(\theta) = (x_b - L \cos \theta)^2 + (y_b - L \sin \theta)^2 \]

\[ E(\theta) = E_r(R - U^{0.5}) \]

\[ E(\theta) = E_r(D) = D^2 \]

\[ R = r_a + r_b + M \]

\[ D = R - U^{0.5} \]

\[ \frac{\partial E}{\partial \theta} = \frac{\partial E}{\partial D} \frac{\partial D}{\partial \theta} \]

\[ \frac{\partial E}{\partial \theta} = 2D(-0.5U^{-0.5})(2Lx_b \sin \theta - 2Ly_b \cos \theta) \]
\[
\frac{\partial E_r}{\partial x} \\ E(\theta) = E_r(r_a + r_b + M - Dist(x, y))
\]

Sphere B location: 
\((x_b, y_b)\)

Sphere A location: 
\((x + x_a, y + y_a)\)

Distance-squared(A, B):
\[
U(x) = (x + x_a - x_b)^2 + (y + y_a - y_b)^2
\]
\[ \partial E_r / \partial x \]

\[ E(x) = E_r(r_a + r_b + M - Dist(x)) \]
\[ U(x) = (x + x_a - x_b)^2 + (y + y_a - y_b)^2 \]

\[ E(x) = E_r(R - U^{0.5}) \]
\[ E(x) = E_r(D) = D^2 \]

\[ R = r_a + r_b + M \]
\[ D = R - U^{0.5} \]

\[ \frac{\partial E}{\partial x} = \frac{\partial E}{\partial D} \frac{\partial D}{\partial U} \frac{\partial U}{\partial x} \]

\[ \frac{\partial E}{\partial x} = 2D(-0.5U^{-0.5})2(x + x_a - x_b) \]
Appendix B
Computing $\nabla I(x)$ using Least Squares
Consider a local 5x5 grid centered on \((x,y)\).
Offset all \(I()\) values s.t. \(I(x,y)=0\).
Omit all cells that are solid or are occluded by solid cells.

Solve for \(\nabla I:\)

\[
\begin{bmatrix}
-2 & -2 \\
\vdots & \vdots \\
0 & 0 \\
\vdots & \vdots \\
2 & 2 \\
\end{bmatrix}
\begin{bmatrix}
\nabla I_x \\
\nabla I_y \\
\end{bmatrix} =
\begin{bmatrix}
I(x - 2, y - 2) \\
\vdots \\
I(x, y) \\
\vdots \\
I(x + 2, x + 2) \\
\end{bmatrix}
\]
\[ \nabla I(x) \]

Use Least Squares

\[
\begin{bmatrix}
-2 & -2 \\
\vdots & \vdots \\
0 & 0 \\
2 & 2
\end{bmatrix}
\begin{bmatrix}
\nabla I_x \\
\nabla I_y
\end{bmatrix}
= \begin{bmatrix}
I(x - 2, y - 2) \\
\vdots \\
I(x, y) \\
\vdots \\
I(x + 2, y + 2)
\end{bmatrix}
\]

Write as:

\[ A \nabla I = B \]

Approximate solution as:

\[ \nabla I = (A^T A)^{-1} A^T B \]