Mandelwalk

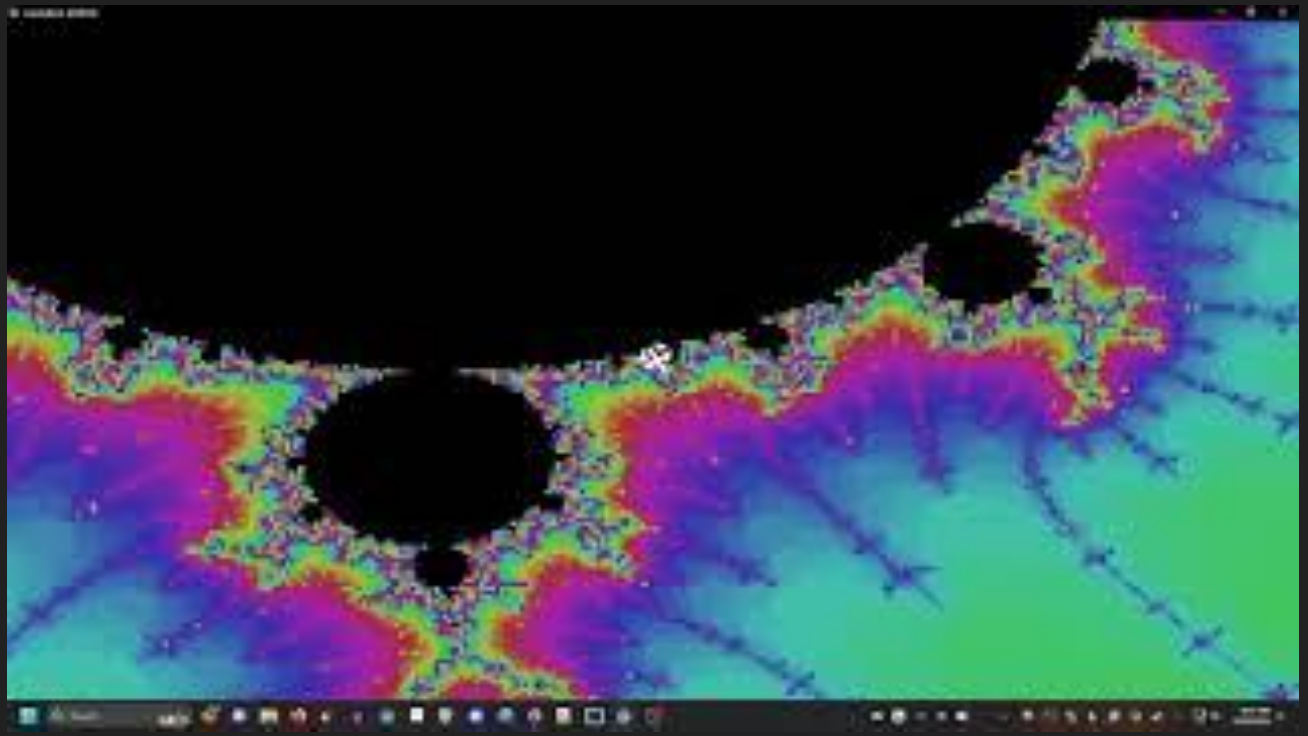
by Micah Nichols

Problem Description

- GOAL: Create a Mandelbrot platformer
  - Existing visualizations are neat, but aren’t interactive enough

Mandelwalk is a unique Mandelbrot platformer where you play as a cursor that can traverse the Mandelbrot boundary. All assets for the project (music, sound effects, shaders, code) were produced by me.

Demo



Languages/libraries/etc. used

- Godot 4.2 for engine (rendering, game scripting, scene building)
- godot-rust: A GDExtension to call Rust code from Godot
- ocl: An OpenCL crate for Rust

Most of the challenge in this project comes from the orchestration of each of these libraries in tandem (can I even call GPU code from Godot like this?), along with my inexperience with Rust.

Why use GPU

Games are as good as their responsiveness.

Calculating the Mandelbrot set quickly on-demand is ideal for ensuring that the exploration is as smooth as possible.

How GPU used

The OpenCL library used for the project exposes a similar interface for creating and executing GPU kernels (which are similar to CUDA kernels).

Rust OCL has a fantastic abstraction on top of the OCL API, allowing it to automatically manage the platform context, device management, command queue, and the majority of the kernel execution functions.

How to Evaluate

Benchmarking the following functions, and comparing their execution time between Godot and Rust+OCL:

1. Test function (populating a large array with 255)
2. Mandelbrot function (populating a large array with RGB image data of the Mandelbrot set, 256 iterations)
3. Terrain function (calculating the game terrain/normals for physics)

Evaluation Results

CPU: Intel i7-10700F CPU @ 2.90GHz

GPU: NVIDIA GeForce RTX 2070 SUPER

Evaluation #1: Test

Resolution	Godot time	Rust+OCL time
288x162	0.015 sec	0.117 sec
576x324	0.060 sec	0.167 sec
1152x648	0.242 sec	0.348 sec
2304x1296	0.961 sec	1.074 sec

- Rust+OCL methods take 0.1 seconds longer to populate arrays than Godot
- Makes sense, as Rust+OCL makes the same function calls as Godot in this logic, but with added overhead
- Therefore, this control test measures the Rust+OCL kernel overhead as ~0.1 seconds

Evaluation #2: Mandelbrot

Resolution	Godot time	Rust+OCL time
288x162	0.675 sec	0.128 sec
576x324	2.665 sec	0.169 sec
1152x648	10.761 sec	0.348 sec
2304x1296	43.848 sec	1.083 sec

- Performance gains are extremely present here
- Godot is very bad at Mandelbrot
- Note that 288x162 is the resolution used for the renders in-game

Evaluation #3: Terrain

Resolution	Godot time	Rust+OCL time
288x162	0.026 sec	0.140 sec
576x324	0.104 sec	0.211 sec
1152x648	0.421 sec	0.526 sec
2304x1296	1.71 sec	1.705 sec

- Godot really wins this one
- The actual terrain computation is simpler than anticipated, and Rust+OCL is heavily bottlenecked by the overhead measured

That’s all

You get no graphs