



# ECE 555 Group Presentation

---

Igor Semyonov   Jordan Carnes   Robert Laverne Griffin

George Mason University, Department of Electrical and Computer Engineering

April 15, 2025

# Why not Just Use C?

It should not be used for production.

See <https://veresov.pro/cmustdie/>

# Why Rust

## Type System

**Listing 1:** Cats

```
1 struct FakeCat {  
2     alive: bool,  
3     hungry: bool,  
4 }  
5  
6 enum RealCat {  
7     Alive {  
8         hungry: bool,  
9     },
```

Impossible states can be encoded in the type system, which leads to compile time errors, instead of runtime checks.

**Listing 2:** Unphysical (zombie) Cat

```
1 let fake_cat = FakeCat {  
2     alive: false,  
3     hungry: true,  
4 };
```

# Why Rust Type System

```
1 # main.rs | cats.rs | main
5
4 let real_cat1 = RealCat::Dead;
3 let real_cat2 = RealCat::Alive {
2     hungry: true,
1 };
23 let real_cat3 = RealCat::Dead{hungry: true};
```

rust-analyzer: no such field [E0559]  
rustc: variant `RealCat::Dead` has no field named `hungry`  
all struct fields are already assigned [E0559]

NOR | cats.rs | 5 1 | utf-8 < rust | 88% | 23:1

Figure 1: Error when declaring an unrealistic cat

## Why Rust Safety

- Result: Errors are values, i.e., no hidden control flow
- Functions that can fail return Result and this must be explicitly handled.
- No manual memory management
- Values are dropped once they go out of scope

# Why Rust

## Ergonomics

High level abstractions that enable multithreading without the error-prone approach of pthreads in C.

# Why Rust

## Single threaded bsort

### Listing 3: Single threaded bsort

```
1 a.chunks_mut(split_length)
2   .for_each(bsort);
```

- `a` is a vector of numbers
- `chunks_mut` slices the vector into non-overlapping slices of the given length
- `for_each` iterates over the chunks, running the provided function with each chunk as the input.

# Why Rust

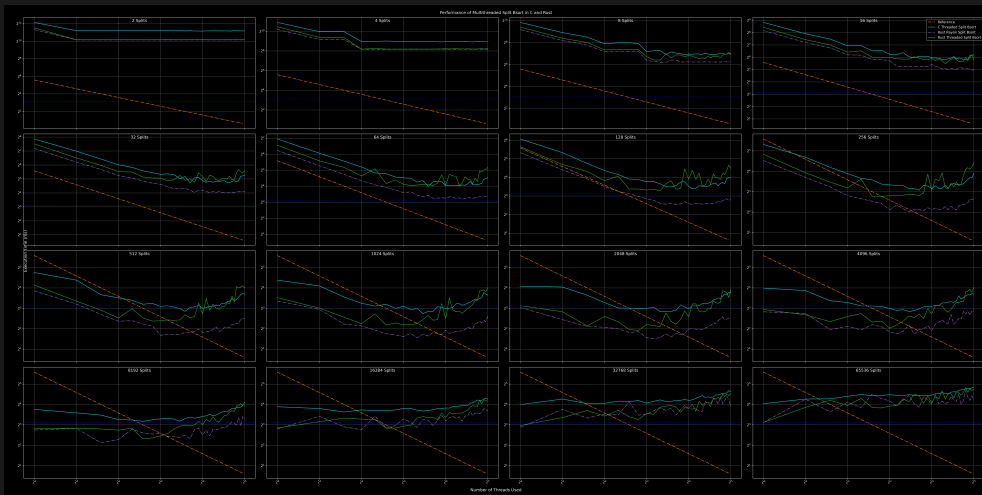
Concurrency is easy while avoiding race conditions

**Listing 4:** Example of easy parallelization

```
1  a.chunks_mut(split_length)
2      .par_bridge()
3      .for_each(bsort);
```

- Chunks is known to split a vector into distinct slices
- Hence we can safely send each chunk to a different thread





**Figure 2:** Preformacne results from bsort project. Note that the Rust reference sort is not plotted as it was under 1ms, compared to 8ms for the C qsort.

# Why Rust

## Zero Cost Abstraction

All these features are provided at compile time and are optimized away when compiling so that they do not affect performance at run time.

# Rust Ownership and Borrowing

## Problems with shared memory access in Rust

- Ownership Rules[1, Ch04-01]
  - Each value has an owner
  - There can only be one owner at a time
  - When the owner goes out of scope, the value will be dropped.
- Borrow Rules[1, Ch04-02]
  - At any given time, you can have either one mutable reference or any number of immutable references.
  - References must always be valid.

C, and especially CUDA, style shared mutable memory access violates the ownership and borrow rules.

At this time, we therefore need to use unsafe Rust when writing and calling GPU kernels.

# Rust and CUDA

## Current options

- Rust GPU
- Rust CUDA

Maybe mention other options for GPU programming in rust, like any rust support for ROCm, HIP, intel gaudi, or other.

# Rust and CUDA

Description of how the Rust-CUDA crate works

# Rust and CUDA

## Focusing on Rust CUDA

Currently being rebooted and is in active development.  
Uses nvidia's nvvm tool which is built on LLVM 7.

# What is a compiler?

Description of problems when going straight from source code to machine code.

## The problem LLVM solves

Description of LLVM and its intermediate representation and how this has enabled much easier language development.



How NVVM works

# Rust CUDA and NVVM

How NVVM is used in Rust CUDA

# Fractals

## Mandelbrot and Burning Ship

$c \in \mathbb{C}$  is in the mandelbrot set if the sequence  $\{z_n\}$  converges.

$$z_{n+1} := z_n^2 + c \quad z_0 = 0$$

The Burning Ship fractal is defined similarly but the sequence is

$$z_{n+1} := (|\operatorname{Re}(z_n)| + |\operatorname{Im}(z_n)|i)^2 + c \quad z_0 = 0$$

# Mandelbrot Kernel

```
1 pub unsafe fn mandelbrot(  
2     n_re: usize,  
3     n_im: usize,  
4     re_min: f32,  
5     re_max: f32,  
6     re_range: f32,  
7     im_min: f32,  
8     im_max: f32,  
9     im_range: f32,  
10    zn_limit: u32,  
11    out: *mut u8,  
12 ) {  
13     let idx_linear = thread::index() as usize;  
14     let idx = thread::index_2d();  
15     let idx_re = idx[1];  
16     let idx_im = idx[0];  
17  
18     let c_re = re_range  
19         * (idx_re as f32 / (n_re - 1) as f32)  
20         + re_min;  
21     let c_im = im_range  
22         * (idx_im as f32 / (n_im - 1) as f32)  
23         + im_min;
```

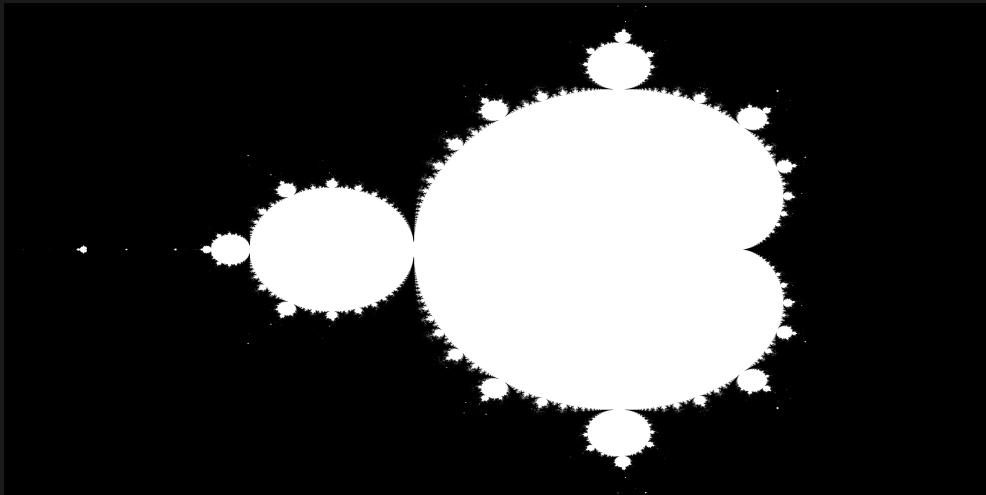
```
24     let mut z_re = c_re;  
25     let mut z_im = c_im;  
26     for _ in 0..zn_limit / ZN_SKIP {  
27         for _ in 0..ZN_SKIP {  
28             (  
29                 z_re, z_im,  
30             ) = (  
31                 z_re * z_re - z_im * z_im +  
32                     c_re,  
33                 2.0 * z_re * z_im + c_im,  
34             );  
35             if z_re * z_re + z_im * z_im > 4.0 {  
36                 let elem = &mut *out.add(  
37                     idx_linear);  
38                 *elem = 255;  
39                 break;  
40             }  
41         }  
}
```

# Launching the Kernel

```
1      unsafe {  
2          launch!(  
3              module.mandelbrot<<<<grid_size, block_size,  
4                  N_RE,  
5                  N_IM,  
6                  re_min,  
7                  re_max,  
8                  re_range,  
9                  im_min,  
10                 im_max,  
11                 im_range,  
12                 zn_limit,  
13                 out_gpu.as_device_ptr(),  
14             )  
15         )?;  
16     }
```

- Fairly similar to launching a kernel in C.
- Must be used inside an unsafe block since the kernel itself is an unsafe function.

## Output Image



# Fractals

## Timing Results for 1 frame on CPU and GPU

I will add the single, frame timing results here

# Fractals

## Live Demo

I plan to be sharing my screen for the presentation and will switch to a live demo here.



# This is a slide

## With a subtitle

This is some text in a column. Could be a figure instead.

- This is a list
- It is an itemized one
- Hence the bullets



S. Klabnik, C. Nichols, K. Chris, and Rust Community, *The Rust Programming Language*.  
No Starch Press, 2024.