CATCHING RENDERING REGRESSIONS ON ALL PLATFORMS

François Mockers

Bevy Game Development Meetup #4 - 2024-05-24

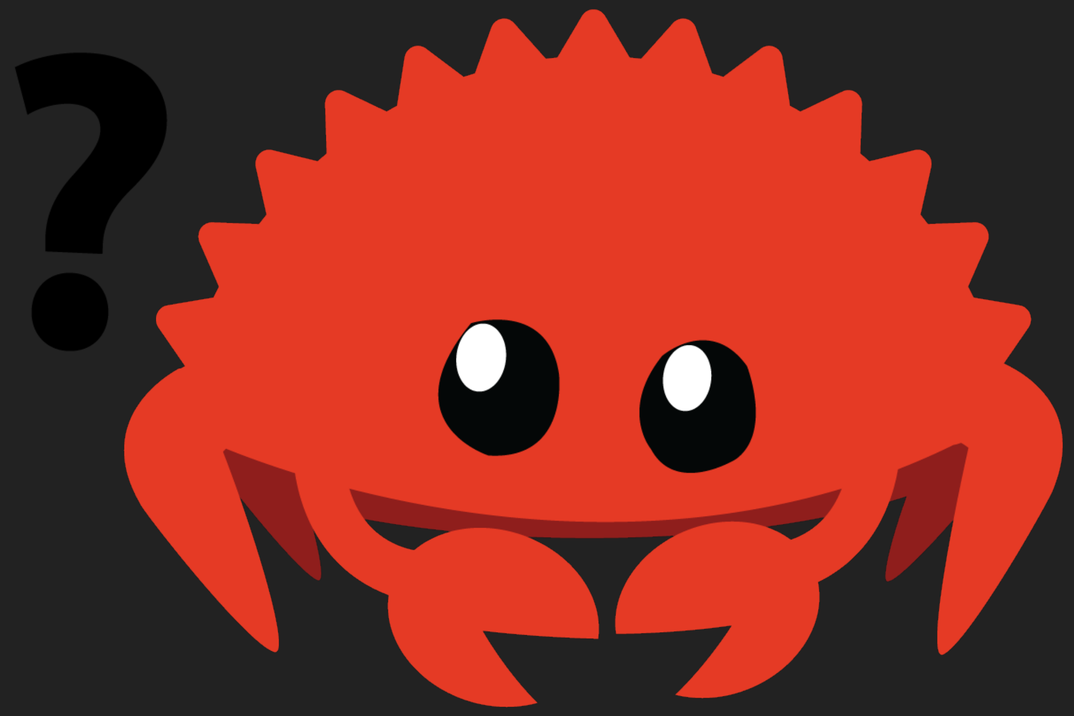
HOW TO TEST A GAME ENGINE

ON MACOS, LINUX, WINDOWS, IOS, ANDROID, WEBGPU, WEBGL2, . . .

IN CONTINUOUS INTEGRATION

WHO AM I

- François Mockers, mockersf on GitHub
- Using (and contributing to) Bevy since the 0.2
- Maintainer and SME on several topics
- Come talk to me if you have ideas to improve testing the engine!



HOW TO TEST A GAME ENGINE

UNIT TEST

- Test a specific function
- Coverage in Bevy is very low
  - We’re trying to increase it
  - Aiming for 100% coverage is not a good thing, it would mean useless tests and harder to maintain code

DOC TEST

- Show how to use a function
- UX test, typically doesn’t have assertion

INTEGRATION TEST

- Test the public API, and how different modules work together
- There are a few in Bevy!
- They’re actually documentation on how to test a game
  - tests/how_to_test_systems.rs
- Bevy doesn’t have integration tests

BENCH TEST

- Check performances of a function, compared to the last run
- Mostly focused on the ECS
- A few on reflection and on async tasks

EXAMPLES

- Big focus of Bevy
  - How to use features
  - Demo of the engine
- Run the examples in CI, catch crashes
- Encourage contributors to run examples related to changes

WHAT ABOUT RENDERING?

TESTS

- Check that the rendering world has the expected entities / components
- Check the rendering commands generated
- Hard to write, hard to maintain

VISUAL TESTS

- Snapshot testing
  - Take a screenshot before the change
  - Take a screenshot after
  - Compare them
- Comparison?
  - Pixel by pixel
  - FLIP from NVIDIA
  - By structural similarity
  - By histogram

REUSING THE EXAMPLES

- Run every examples
- Take a screenshot
- Compare the screenshot with last execution on `main`

DRIVING EXAMPLE EXECUTION

- Feature `bevy_ci_testing`

```
(
    setup: (
        fixed_frame_time: Some(0.03),
    ),
    events: [
        (200, Screenshot),
        (900, AppExit),
    ]
)
```

MAKING THE EXAMPLES DETERMINISTS

- Examples need to render exactly the same at a given frame
  - Fixed timestep
  - Only seeded random

INTRODUCING PIXEL EAGLE

- https://pixel-eagle.vleue.com/
- Image comparison as a service
- Made with Bevy in mind
  - Hash comparison of screenshots first
  - Pixel by pixel if needed
- Still basic for now, still evolving

BENCH TEST FOR THE RENDERER

- Coming soon…
- Blocked on having actual hardware
- Start by documenting and standardising how to bench test the renderer

BUT... ON ALL PLATFORMS?

CONSTRAINTS

- In the cloud
- Headless
- Reliable

LINUX

- Use llvmpipe software renderer
- Use xvfb to get a virtual X11 desktop
- `SystemInfo { os: "Linux 22.04 Ubuntu", kernel: "6.5.0-1021-azure", cpu: "AMD EPYC 7763 64-Core Processor", core_count: "2", memory: "15.6 GiB" }`
- AdapterInfo { name: "llvmpipe (LLVM 15.0.7, 256 bits)", vendor: 65541, device: 0, device_type: Cpu, driver: "llvmpipe", driver_info: "Mesa 23.3.6 - kisak-mesa PPA (LLVM 15.0.7)", backend: Vulkan }
- TODO for Wayland

MACOS

- Just works on the m1 GitHub runners
- `SystemInfo { os: "MacOS 14.4.1 ", kernel: "23.4.0", cpu: "Apple M1 (Virtual)", core_count: "3", memory: "7.0 GiB" }`
- `AdapterInfo { name: "Apple Paravirtual device", vendor: 0, device: 0, device_type: IntegratedGpu, driver: "", driver_info: "", backend: Metal }`

WINDOWS

- “Just” works on the windows GitHub runners
- `SystemInfo { os: "Windows Server 2022 Datacenter", kernel: "20348", cpu: "AMD EPYC 7763 64-Core Processor", core_count: "2", memory: "16.0 GiB" }`
- `AdapterInfo { name: "Microsoft Basic Render Driver", vendor: 5140, device: 140, device_type: Cpu, driver: "", driver_info: "", backend: Dx12 }`
- Actually very slow and very unstable
  - Most 3d examples don’t work
  - Disabled since 2024/05/15

WASM / WEBGL2

- Ubuntu / Firefox combination works (with xvfb and llvmpipe)
- Limited scope for now, only 6 examples
- Driving a browser with Playwright
- Uses the screenshot feature of Playwright, not the built in from Bevy

WASM / WEBGPU

- macOS / Chromium combination works
- Same status as WebGL2

IOS

- Tested on iPhone 13, 14 and 15, with iOS 15, 16, 17
- Thanks to BrowserStack Open Source support
- Run the mobile example
- Test done with Appium
- Uses the screenshot feature of Appium, not the built in from Bevy

ANDROID

- Tested on Xiaomi Redmi Note 11, Google Pixel 6, Samsung Galaxy S23, Google Pixel 8, with Android 11, 12, 13 and 14
- Thanks to BrowserStack Open Source support
- Same as iOS
- Android has more limitations
  - Some GPUs fail with MSAA
  - Some GPUs fail with shadows
  - Some GPUs fail with GPU preprocessing

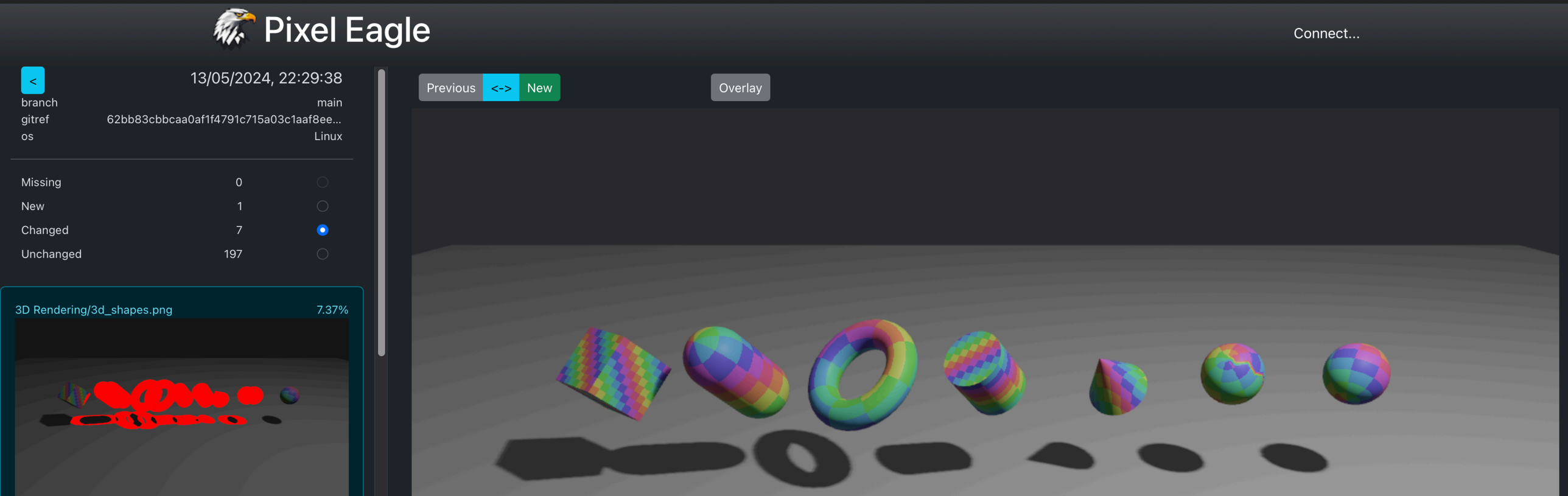
HOW DOES IT LOOK

GITHUB ACTIONS

- https://github.com/TheBevyFlock/bevy-example-runner/actions/workflows/report-main.yml

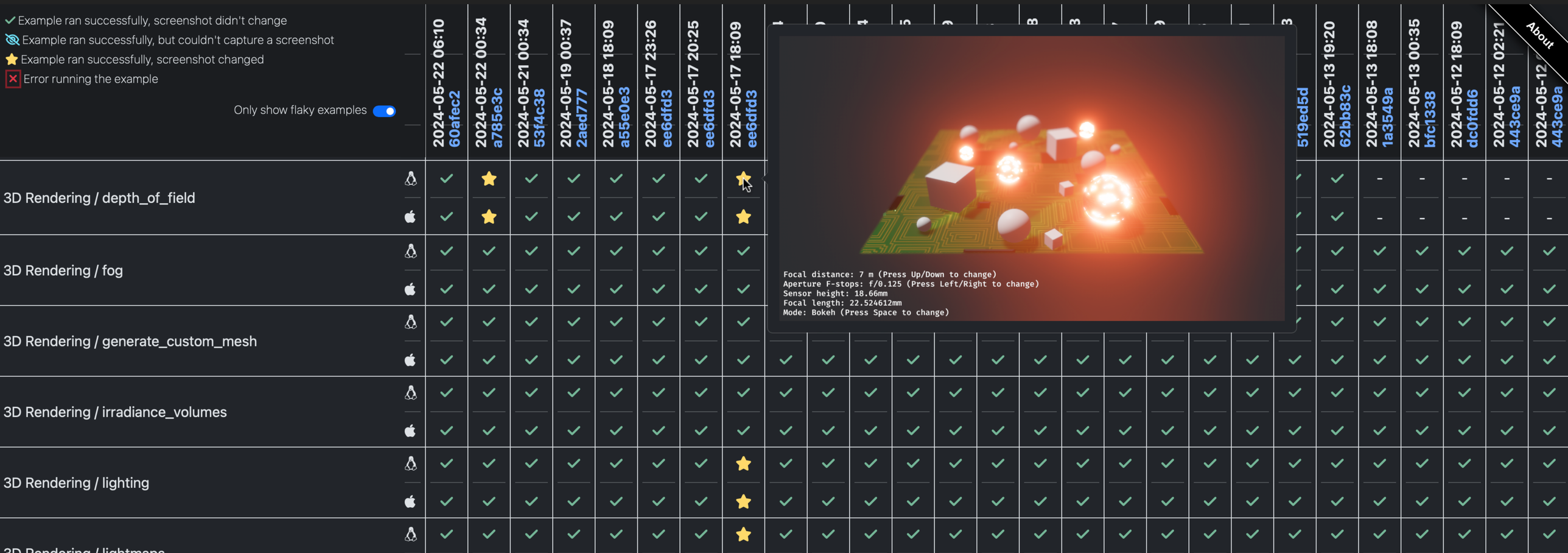
PIXEL EAGLE

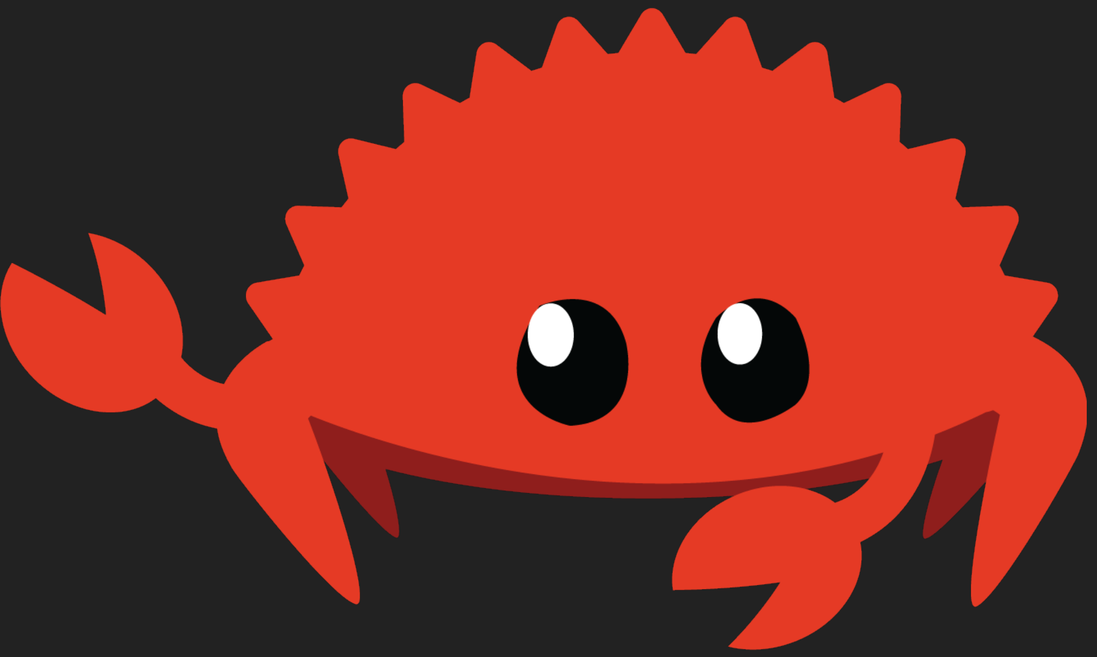
- https://pixel-eagle.vleue.com/project/B25A040A-A980-4602-B90C-D480AB84076D



REPORT PAGE

- https://thebevyflock.github.io/bevy-example-runner/index.html





NEXT STEPS

MORE MOBILE DEVICES

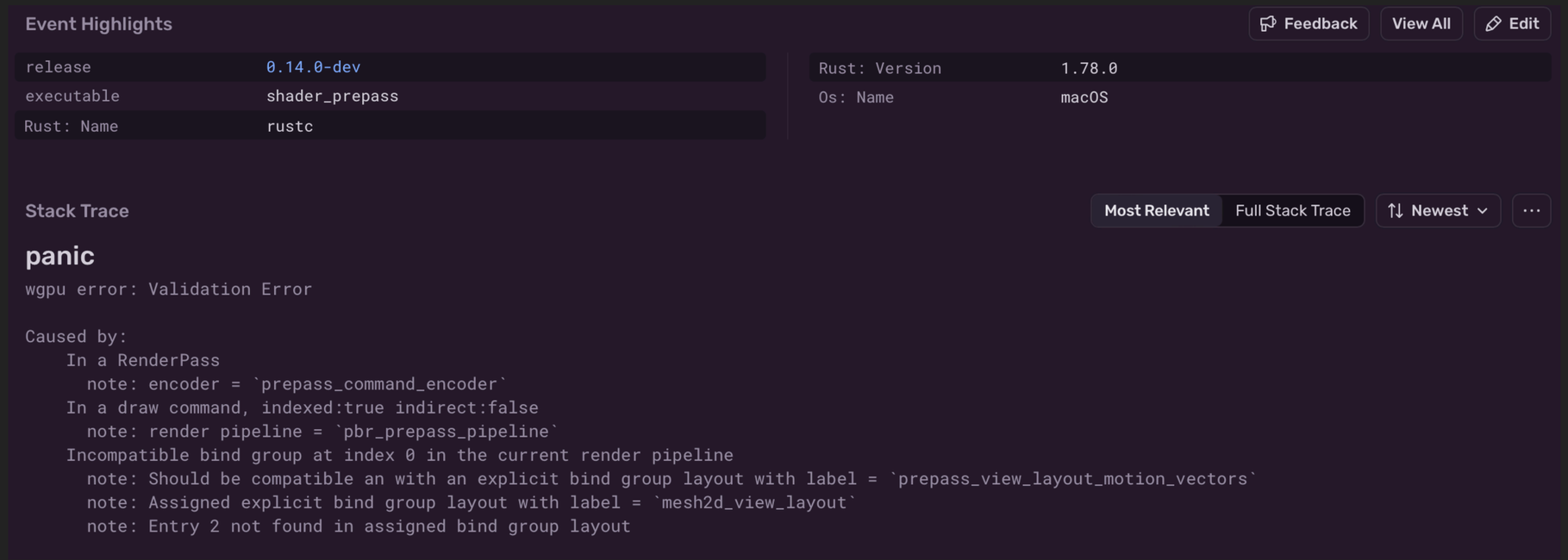
- Waiting for some bugs to be fixed on `main` (in milestone 0.14)
- https://www.browserstack.com/list-of-browsers-and-platforms/app_automate
- Need good coverage of
  - GPU
  - OS version
  - Device manufacturer

IMPROVE WASM TESTS

- Include them in the report page
- Run more than 6 tests
- Take screenshots through Bevy

ERROR REPORTING

- Getting error log from native build is OK, but hard from wasm or mobile
- Integration with Sentry https://github.com/vleue/vleue_sentry
- https://vleue.sentry.io/share/issue/e2578283652d4c449218ccfbdba37c3b/



INTERACTIONS

- New `bevy_ci_testing` command to send keyboard / mouse input to an example
- Change rendering settings in some examples, take a screenshot for each
- Render different scenes on mobile to test different pipelines

DEBUG BY PRINTLN / RUN ONE SHOT SYSTEM

- New `bevy_ci_testing` command to print diagnostics in logs
  - Can be retrieved from outside execution, to help debug from CI

- New `bevy_ci_testing` command to run a one shot system
  - Can run complex checks on entities / components, crash if not valid
  - Can trigger complex changes in the game to switch state or scene

PR VALIDATION

- Currently, only run after merge, every 6 hours
- Run a subset of examples on PR validation before merging
- Comment on PR with result
  - Do not block merging on difference, sometimes they are expected
- PR opened: https://github.com/bevyengine/bevy/pull/13248

IMPROVE PIXEL EAGLE

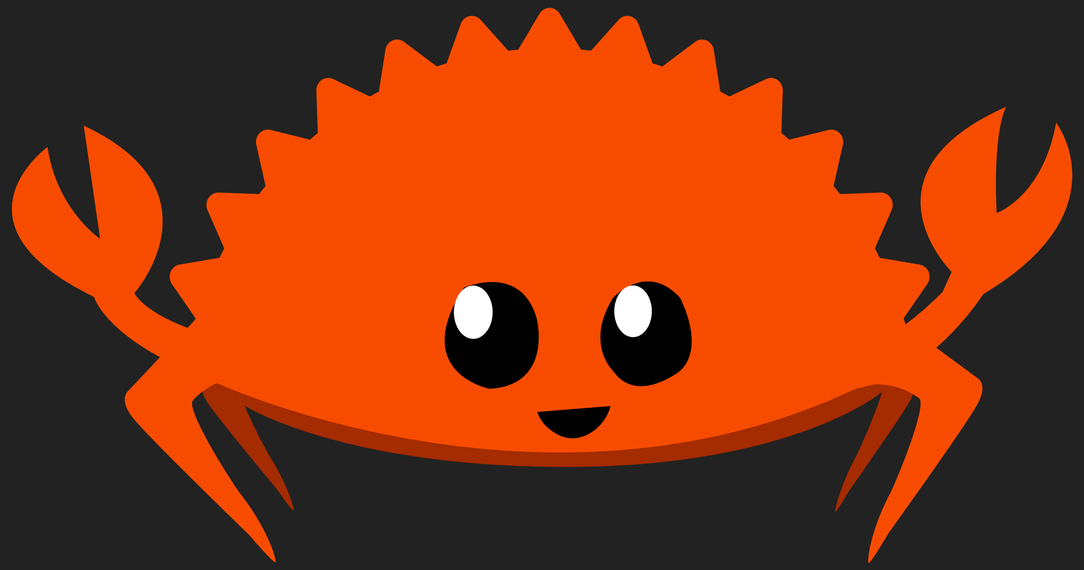
- Better listing page
- More comparison modes than pixel by pixel
- Ignored zones

- Find more users

HOW CAN YOU HELP

HOW CAN YOU HELP

- Some of the next steps just need contributors
  - Open PRs!
- Some of these things would cost money to go further
  - Sponsor the Bevy foundation!
- Would visual tests be useful for your project / organisation / company?
  - Come talk to me!



CLOSING THOUGHTS

NEAR THE END OF WHAT'S POSSIBLE FOR FREE

- Running every example is slow… when there are 241 examples, and regularly increasing
- Running faster would need more / bigger runners
- Enabling Windows again probably need bigger Windows runners
- Testing against realistic user configurations would need actual hardware

BACK OF NAPKIN COST ESTIMATIONS

- 20 examples by matrix job, split into 13 jobs per OS
  - Linux: 250 minutes
  - macOS: 137 minutes
  - Windows: 387 minutes
- Would cost $13 per run ($19 with Windows) on GitHub standard runners
- Around 40 PRs merged per week, $2080 / month of CI
  - Larger runners start at 2x price, wouldn’t be 2x performances

- Hardware colocated in a datacenter
  - Part time admin (on-call, on-site for hardware work): $2000 / month
  - Datacenter colocation: $150 / month

NOT JUST FOR THE GAME ENGINE

- Most of the setup used to test Bevy is built in and free
- You can use it in your game!
  - Test all targets
  - Test your shaders
  - Test UI positioning visually
  - Test transitions