Industry Advance

Davids Paskevics, Phillip Wellner, TODO

FU Berlin

2020

The project

- Game for the GBA console
- Port/clone of the libre Java game *Mindustry*
- Mashup of logistics sim/tower defense
- Less complex than Factorio/Minecraft, so will hopefully fit on the HW

The hardware

- Handheld game console by Nintendo
- JP release in 2000
- Engineered for cost and battery life
  - 32 bit ARMv4t ISA processor
  - Clock @16MHz
  - No cache
  - 32KiB fast (on-die) RAM, 256Kib slow (off-die) RAM $\rightarrow$ We put stack here
  - "slow RAM" = narrow bus + wait cycles $\rightarrow$ We put heap here
  - Generous up to 32MiB cart ROM
  - Nintendo wanted to make the "ultimate 2D handheld" $\rightarrow$ no HW 3D support or FPU

Prior art

- ISA supported in *LLVM* (and therefore *rustc*)
- Cross-compiling *libcore* easy thanks to *build-std* cargo feature
- Amazing rust-console/gba crate provides basic HAL
- Snake game from around 2015, but no "complex" games written in Rust to our knowledge

Starting out

- "Hello world" is tricky on this hardware
- Easier to draw something
- Shouldn't be hard, right?
- Until you want to draw something useful for a game, that is

Graphics on the GBA

- Modern consoles/PCs are powerful enough to allow uploading pixels to the display freely
- GBA has such "bitmap" modes as well (slow!)
- Also supports HW accelerated "tile modes" (fast!) TODO: Insert graphic on how tiles/sprites work
- Problem: custom GFX format
- Solution: Use a converter (and an ugly python wrapper)

Rust ecosystem

- Use of easily integrated *no_std* crates
  - *serde* for map metadata loading
  - *tiny_ecs* for ECS (good pattern for structuring games in Rust)
  - *hashbrown* for hashmap
  - *fixed* for fixed-point maths (performant)
- Crates reduced dev time by a lot
- Downside: Some libs assume stuff (atomics support)
- Downside: Others are needlessly *std*-dependent (patches)

Rust APIs

- Tile-based custom text output w/ formatting in <50 LOC thanks to *Write* trait
- Custom test framework w/ minimal boilerplate

Rust safety

- Only hit memory bugs in our *unsafe* allocator
- Saves us a lot of time (which we need to squash logic bugs instead)
- *clippy* could be better at reporting alignment issues

Rust speed

- No CPU bottleneck yet, despite many abstractions
- Code isn't very optimized yet
- *const_fn* allowed for a low-cost FS implementation

We didn't finish

- Picked a project too ambitious in scope
- Underestimated engine dev effort
  - The HW is not designed for this kind of game
  - Still not sure how we'll implement saving, multiplayer, etc.
- This is not a 5h/week job

Conclusion

- I believe Rust is a good fit (but I'm biased)
- Will develop project further
- Interested? Fork us on github: https://github.com/industry-advance/industry-advance (take a look under "Projects"!)