impl Game for App

http://www.piston.rs/

A presentation about Piston and Rust-Graphics
(written by Sven Nilsen)

About Sven Nilsen

- Bachelor in computer science with speciality in AI
- ... but worked with 2D software for 6+ years!
- Working with other people on:
  - Rust-Graphics (back-end agnostic 2D library)
  - Piston (back-end agnostic game engine)
  - Rust-Empty (a *user friendly* makefile for Rust projects)

What if...

- ... you could write a small game in a few days
- ... without knowing anything about OpenGL
- ... in a language known for its steep learning curve
- ... without sacrificing performance
- ... or scalability
- ... or portability?

Fill rectangle in Piston

```
use graphics::*;
use piston::{Game, RenderArgs};

pub struct App { ... } // your application data

impl Game for App {
    fn render(&mut self, args: &mut RenderArgs) {
        ... // Set up context and clear the screen
        // Draws a red rectangle in upper left corner.
        c.rect(0.0, 0.0, 50.0, 50.0).rgb(1.0, 0.0, 0.0).fill(args.gl);
    }
}
```

Context types – a new design tool

- Example: c.rect(x, y, w, h).rgb(1.0, 0.0, 0.0).fill(gl);
- Put data in – get new type out
- Not bound to device or back-end
- Immutable – always generate a “new context”
- Clone-able – allows returning value from a function
- No model/world separation – just use 2 contexts!
- Catches many bugs caused by source changes
- Makes it easier to pick special case algorithms

Context types - optimization

- All constructors are inlined
- enum Field<'a, T> { Value(T), Borrowed(&'a T) }
- Compiler see pointers – no stack allocation
- Emits same assembly as under hard coding
- Uses the stack built into the language
- Theoretically faster than traditional APIs
  - No book keeping of previous values
  - No undoing of relative transforms

Piston game loop

- GameIterator – state machine
- Settings for UPS and FPS
- Fixed update time step – allows determinism
- Extrapolated time for smooth rendered motion
- The `Game` trait wraps the iterator
  - More familiar for people using existing APIs

Piston goals

- Cover the same space as Processing or Löve but for Rust programming language
- Cross platform development
- Test performance of Rust-Graphics
- Platform for game design research (AI? Network?)
- Make more people use Rust for game development
- Work with 3D libraries in Rust community

Rust-Graphics goals

- Back-end agnostic 2D graphics
- Make reuse of code easier across projects
- GPU focused
- Use Rust's type system for expressiveness
- Research context types in library design

Links

- Piston: http://www.piston.rs/
- Rust-Graphics: https://github.com/pistondevelopers/rust-graphics
- A great way of learning Rust!
- Thanks for all the contributions so far!