Modern game dev. in Rust,

a comparison to C++ and C#

Thomas H. Sieben
Murphy Scholar Class of 2019

Adviser: *Jesse Tov*

Game scripting languages

Rust

- Relatively new (2015)
- Robust type system
- Controlled memory management
- Concurrency
- Not many game engines/support

C++

- Stable & supported
- Similar performance to Rust
- Many game engines
- Medium barrier of entry

C# (Unity)

- De facto game dev. tool
- Interfaces well with Unity game engine
- Highly supported
- Relatively low barrier of entry

Project goals

Research

Research existing game engines

- rust-sdl2
- piston
- amethyst
- ge211
- unity

Implement

Design a game between all 3 languages

Something easily testable, standardized, and 2D

→ Snake game

In the future, a more graphic intensive game

Assess

Analyze performance and approachability

Design microtests

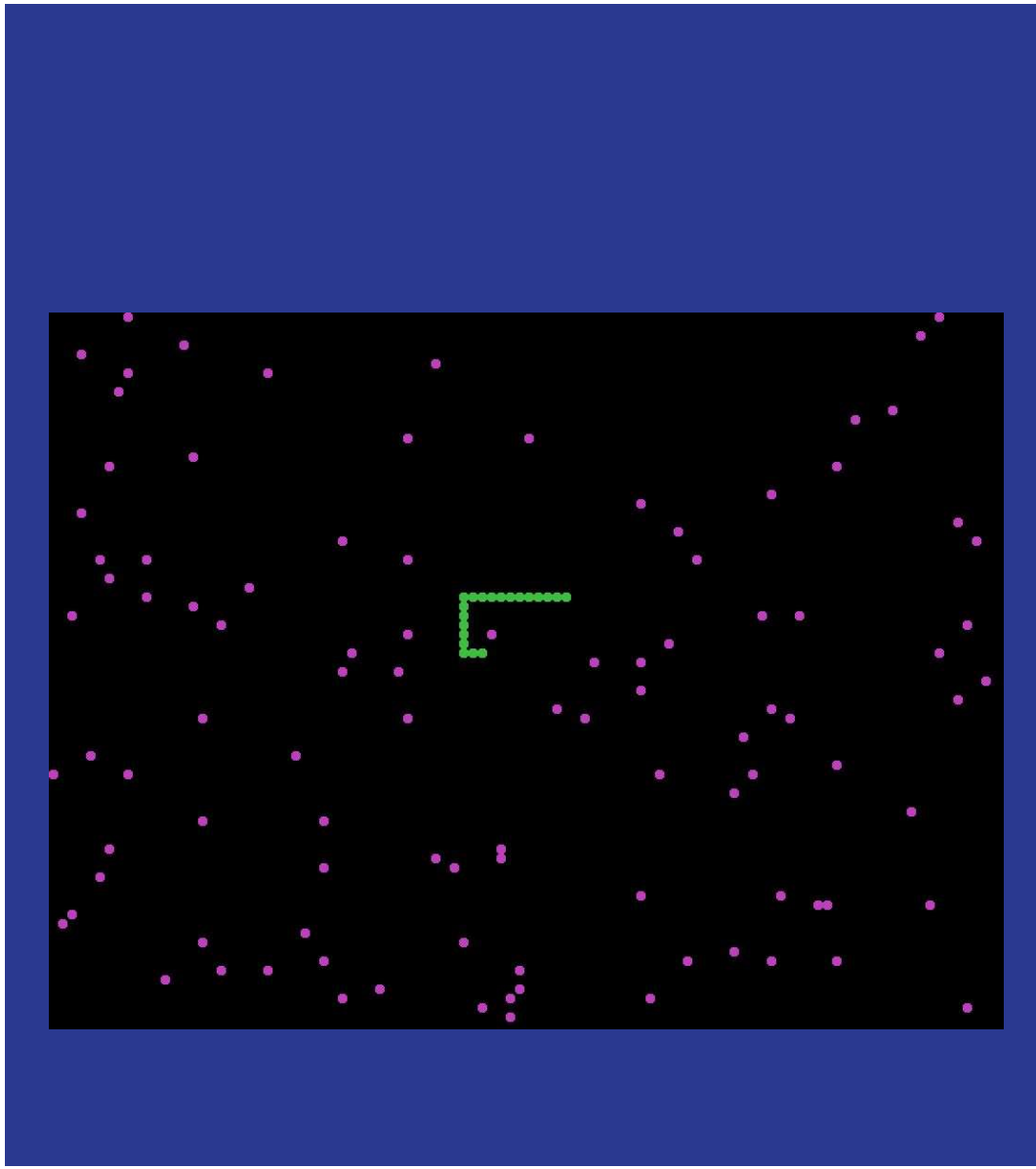
Assess time to complete

General feasibility

Snake implementation

ge211 snake

- ge211 engine designed by adviser, Jesse Tov
- uses sdl2 libraries
- no loss in framerate
- clear documentation, easy to complete



rust-sdl2 snake

- rust-sdl2 engine uses same sdl2 libraries as ge211
- easy to compare
- uses OOP, structures, and inheritance
- piston and amethyst engines were not used
    - little support for more than hobbyist

unity/c# snake

- unity is one of the easiest engines to pick up and master
- uses component-based system over OOP
- game objects can have mixed and matched components

Future considerations

- microtesting
  - add in performance tests between Rust and C++
- switch to a more graphically intensive game
- compare advantages of individual game engines within Rust
  - e.g., amethyst, piston

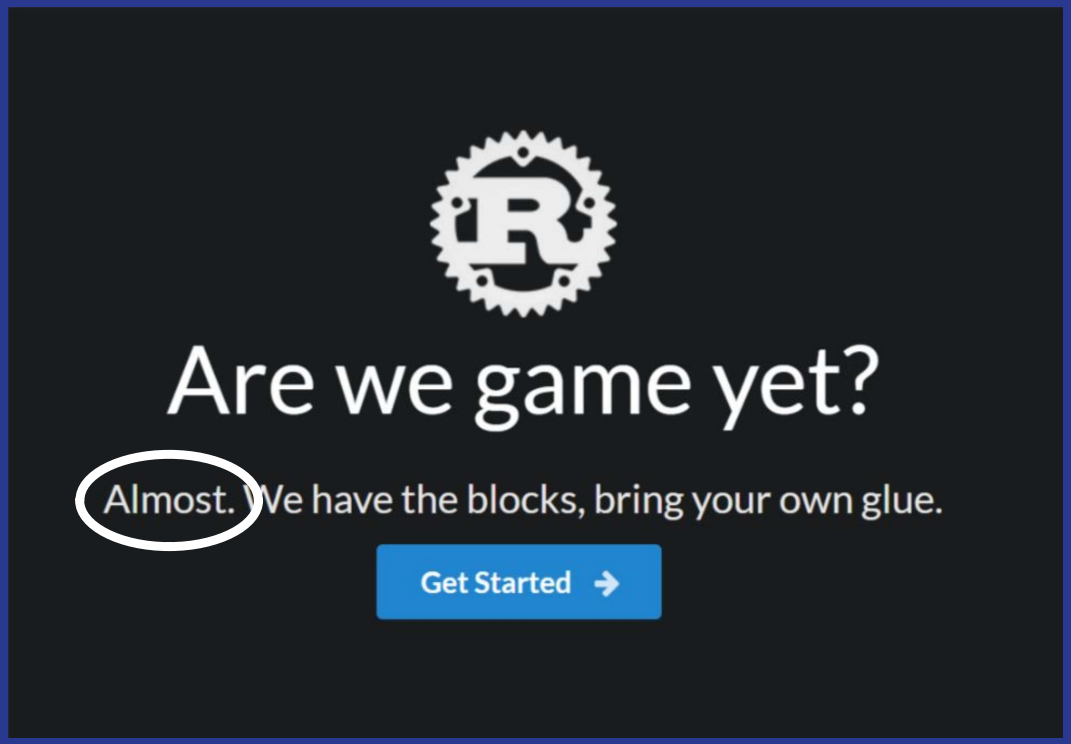
Fireworks

Testing the graphical capabilities and framerate

framerate →

created with ge211, credit to adviser Jesse Tov

Conclusions



Final thoughts

- Rust game dev. is most applicable to hobbyists
- “... bring your own glue.”
- C++ is just as performant, and with more resources
- Real-world dev. occurs in Unity or other industry-based game engines