Amethyst:

Past, Present, & Future

Eyal Kalderon @ebkalderon

Rust Toronto Meetup 2017

Preview

- What is Amethyst?
- Brief History
- Current State
- Future Plans

What is Amethyst?

An Intuitive Game Engine

- Data-oriented
- Data-driven
- Free and open source (MIT/Apache)
- Written in Rust with ♥

Data-oriented

- Programming paradigm
- Storing data compactly and exploiting modern hardware to process it efficiently
- Lends itself well to pipelining, modularity, and massive parallelism (task or data)

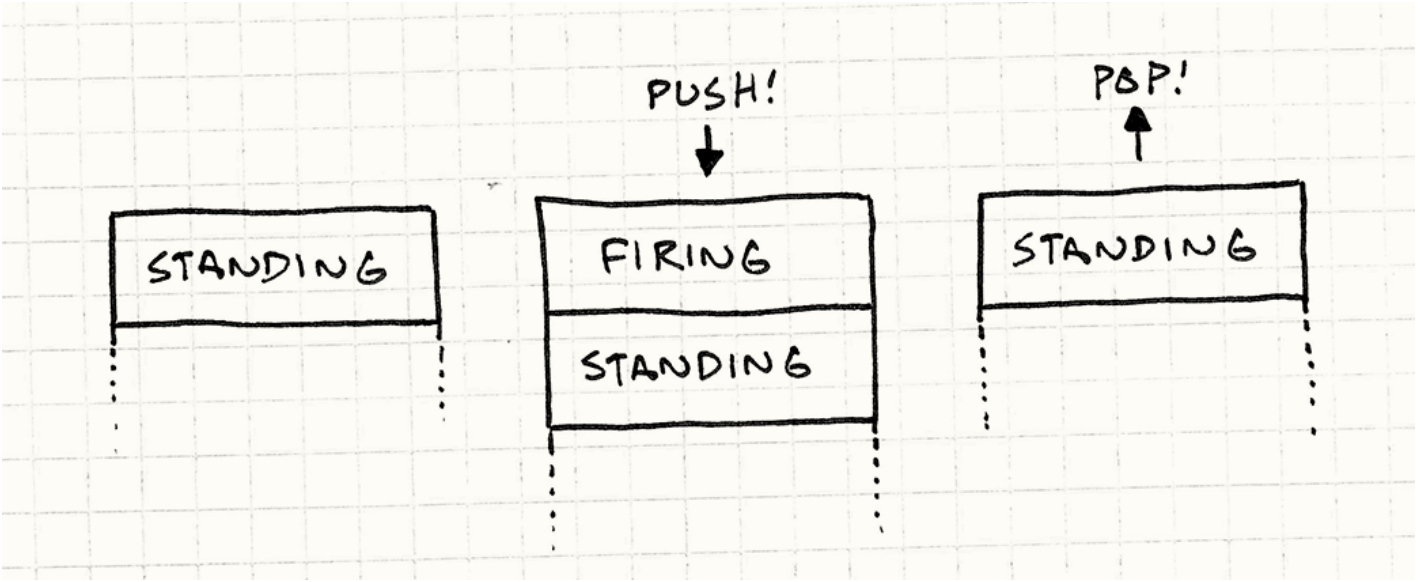
Data-driven

- Software design style
- Critical logic defined in data rather than in compiled code
- Hot-reloading, instant feedback
- Orthogonal concept to data-orientation

Features

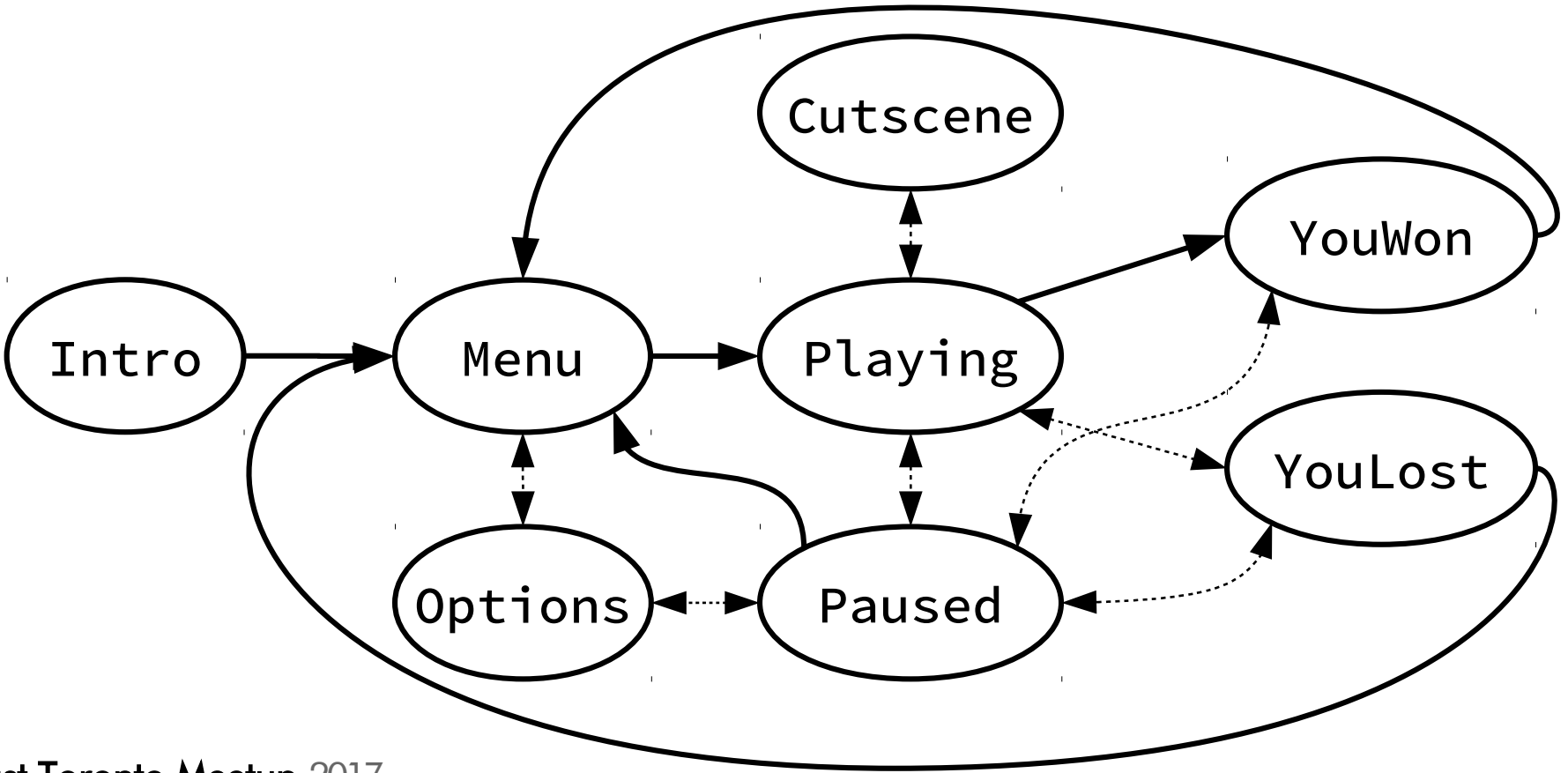
- Easy game state management
  - Pushdown automaton (FSM with memory)
  - Transition between screens or gameplay modes
- Entity-component-system (ECS) model
  - **Specs** provides a scalable parallel framework
- Extensible 3D rendering system
  - Supports custom graphical passes
  - Direct3D 11 and OpenGL backends via **gfx-rs**
- Presented as cohesive unit, subcrates are also usable

Pushdown Automata



Source: Game Programming Patterns

Pushdown Automata



Brief
History

History

- Born 13th Jan 2016
  - First commit 20th Dec 2015
- Personal toy project
  - Practice Rust 1.0
  - Used Dark GDK, Unity, Unreal
  - Inspired by Bitsquid (Stingray)
- Catching steam around Feb 2016
- GitHub organization Mar 2016

History

- Initially intense discussion and design
- Renderer prototyped 4th Apr 2016
- **Specs** parallel ECS
  - Born 5th Apr 2016
  - Separate project
  - Co-evolved alongside Amethyst
- `iter_mut()`, `iter_mut()`, `iter_mut()`

State of
the Engine

Current State

- Growing rapidly, high code churn
- Core features present:
  - State machine, ECS, rendering, configuration (YAML)
- Notably missing:
  - Audio, GUI, physics, tooling, entity definition (YAML)
- Proof of concept Pong clone
- Documentation available, but sorely out of date
- Greater test coverage and benchmarks would be nice

API Example


```
pub struct HelloWorld;

impl State for HelloWorld {
    fn on_start(&mut self, eng: &mut Engine) {
        println!("Starting up...");
    }

    fn update(&mut self, eng: &mut Engine) -> Trans {
        println!("Playing!");
        Trans::Quit
    }

    fn on_stop &mut self, eng: &mut Engine) {
        println!("Shutting down...");
    }
}
```

Future
Plans

More Parallelism

- Asset management is being reworked (#244)
  - Cleaner API
  - Asynchronous and parallel loading using futures
- Renderer is undergoing rewrite (#233)
  - More data-driven
  - Parallel rendering
  - Build pipeline state objects at runtime
  - Collaborating more closely with **gfx-rs**
- **Specs**, asset manager, and renderer all share threadpool

User Experience

- Embedded scripting language support
  - Possibilities: mruby, Lua, Dyon, Javascript, others
- Hot-reloading of Rust code, scripts, and assets
- Improve tooling situation
  - Traditional “mega-editor” split into several small command-line tools, scriptable
  - Editor is merely a frontend to these utilities
  - Compile shaders, generate mipmaps, and compress assets when building in release mode

Determinism

- Many cool improvements to be made here!
- Recording and playback of in-engine demos
  - Think Quake-style demos, compact text files
  - Store RNG seeds, initial world state, player inputs
  - Frame-by-frame rewind and fast-forward
- Multiplayer becomes a networked extension of demos
- Tool slaving, network transparency
  - Simulate on development PC, preview and profile game on mobile/consoles, etc. over USB or network

Join us!

amethyst.rs

Links

- Getting involved
  - https://www.amethyst.rs
  - https://github.com/amethyst/amethyst
  - https://gitter.com/amethyst/rooms
- Further reading
  - https://bitsquid.blogspot.com/
  - http://gamedevs.org/uploads/benefits-of-a-data-driven-renderer.pdf
  - http://gamedevs.org/uploads/flexible-rendering-multiple-platforms.pdf
  - https://torkleyy.github.io/blog/amethyst-assets/

Thank You!

Questions?

References

Design Patterns Revisited – State Pattern. (2009). [Drawing of pushdown automaton]. *Game Programming Patterns*. Retrieved from http://gameprogrammingpatterns.com/images/state-pushdown.png

Wikimedia Commons. (2015). [Wyoming Night Sky]. Retrieved from https://commons.wikimedia.org/wiki/File:February_-conservationlands 15_Social_Media_Takeover-_Top_15_Places_on_National_Conservation_Lands_for_Night_Sky_Viewing_(16358792937).jpg