



---

# The open source game engine Bevy

---

Niklas Eicker

05.07.2024



# About me

- senior consultant at TNG
- using Bevy since 2020
- maintaining several open source plugins
- occasional Bevy contributor

@NiklasEi on GitHub



My first game jam entry using Bevy. Oicana is a mixture of puzzle game and tower defence.

“A refreshingly simple data-driven  
game engine built in Rust Free and  
Open Source Forever!”

— [bevyengine.org](http://bevyengine.org)



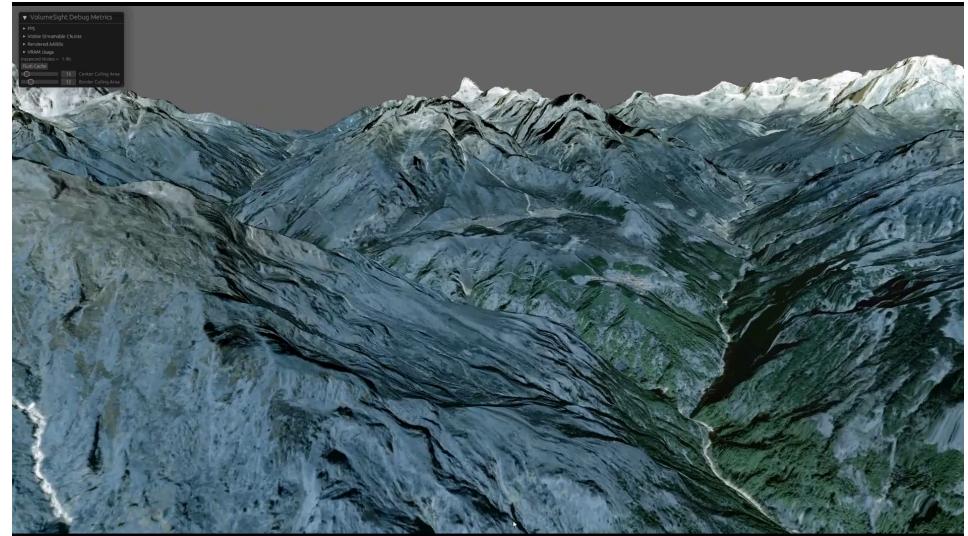
# Bevy Intro

- dual-licensed under Apache 2.0 / MIT
- cross-platform: Windows, macOS, Linux, Web, Android, iOS
- rather new; first public version in August 2020



# Projects using Bevy

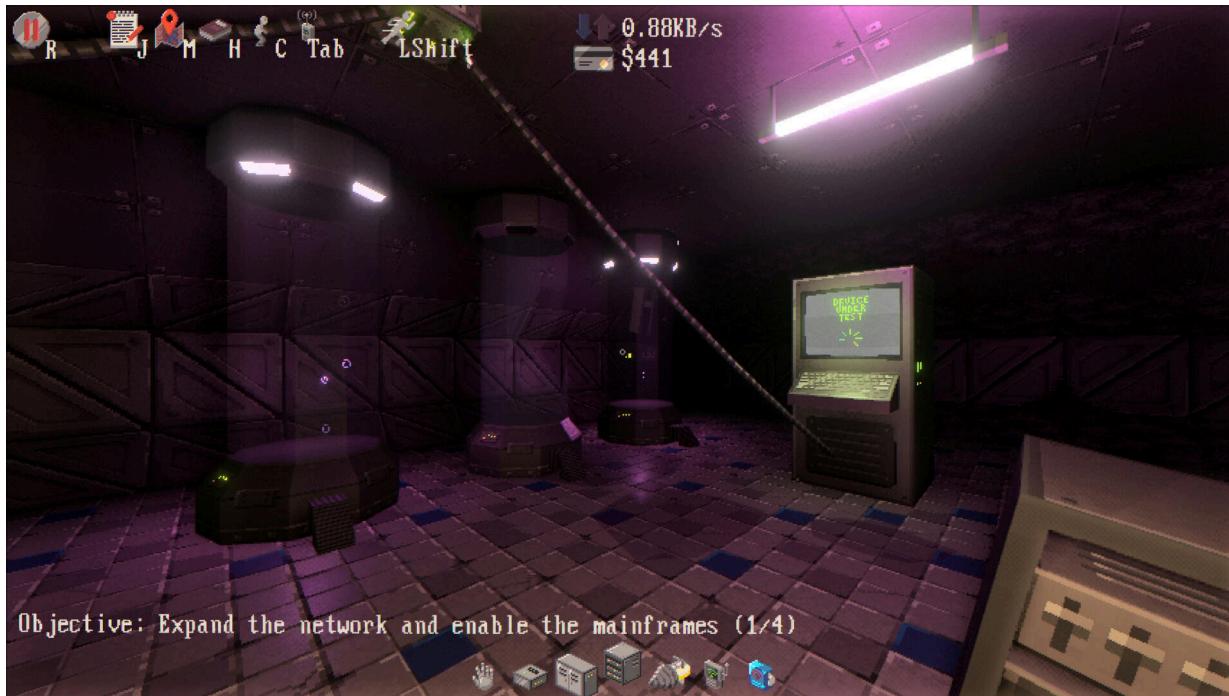
- hundreds of small games on [itch.io](#)
- some usage outside of game dev like CAD and other simulations
- a couple of projects in mobile stores and on steam



32km<sup>2</sup> of Mount Everest in 1m resolution and automatic LOD.  
Shared by @borgerking from Foresight on the Bevy Discord  
(thread).



# Bevy Games on Steam - Tunnet



Build, debug and optimize  
a computer network in  
an underground facility.



# Bevy Games soon on Steam - Times of progress



A City Builder set  
during the Industrial  
Revolution.



# A short organisational history

August 2020



## **First public version**

Version 0.1 is published with a blog post and a community starts to build up.

January 2022



## **Release of Bevy 0.6 - 9 months after 0.5**

From now on: start release process every 3 months

January 2023



## **Subject Matter Experts (SMEs)**

New role to improve bottleneck in reviews



March 2024



## **Bevy foundation**

A non-profit organisation; the board are the maintainers.  
Move Sponsorship from individual contributors to the foundation.

May 2024



## **Alice hired by the foundation**

Alice, a long-time maintainer, is hired by the foundation to work on Bevy full-time.

July 2024



## **Version 0.14.0**

The first version with release candidates



# What makes Bevy different?

- everything based on Entity Component System (ECS)
- written in Rust
- very modular ( 40 “internal plugins”)

```
> └── bevy_a11y
> └── bevy_animation
> └── bevy_app
> └── bevy_asset
> └── bevy_audio
> └── bevy_color
> └── bevy_core
> └── bevy_core_pipeline
> └── bevy_derive
> └── bevy_dev_tools
> └── bevy_diagnostic
```

Some of the internal Plugins that make up Bevy.



## Bevy Games soon on Steam - Tiny Glade



Tiny Glade is a small relaxing game about doodling castles.



A cosy home that I built with the Tiny Glade Demo.



# ECS

Separation of data and behaviour

- **Entity:** id (think 1, 2, 3...)
- **Component:** Data (e.g. position, health points, ...)
- **Systems:** Behaviour (e.g. “update hp based on position”)

Components get assigned to an entity. Systems can query and mutate components.

It's time for some code



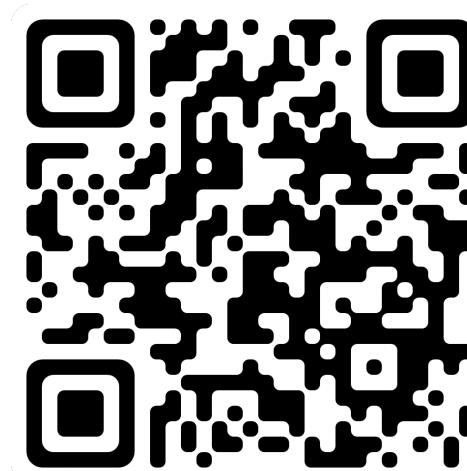
## Where to go from here

- Join the Discord: [discord.gg/bevy](https://discord.gg/bevy)
- Play around with Bevy examples: [github.com/bevyengine/bevy](https://github.com/bevyengine/bevy)
- Bevy Cheatbook: [bevy-cheatbook.github.io](https://bevy-cheatbook.github.io)

## Bevy game jam (probably 13th to 22nd)



# 0.14 release - blog post



Detailed release notes  
for Bevy 0.14

# Give Bevy a try!



Find the code and slides at  
[github.com/NiklasEi/btd24\\_bevy\\_talk](https://github.com/NiklasEi/btd24_bevy_talk)

# Appendix



# Community overview

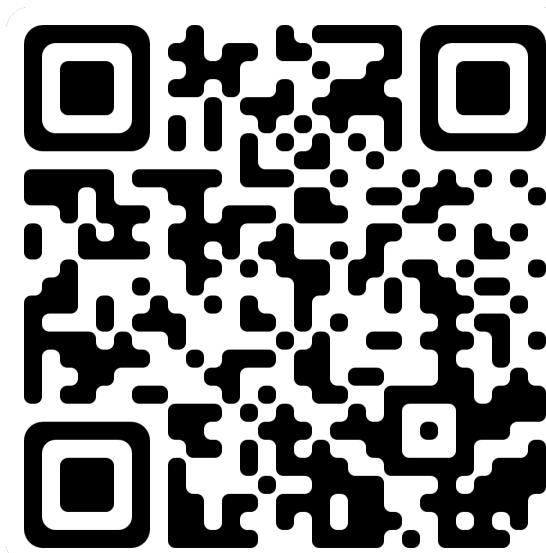
- official Discord server: [discord.gg/bevy](https://discord.gg/bevy)
- development on GitHub: [github.com/bevyengine](https://github.com/bevyengine)
- resources and assets: [bevyengine.org/assets](https://bevyengine.org/assets)



# Why ECS?

- in games, everything wants to access everything else
- imagine what your player class will look like

Watch Catherine West explain it best



“RustConf 2018 – Closing Keynote – Using Rust For Game Development by Catherine West”



# Bevy ECS

```
fn main() {
    let mut world = World::new();
    world.spawn(Counter { value: 0 });
    let mut schedule = Schedule::default();
    schedule.add_systems(increase_counter);
    schedule.run(&mut world);
}

#[derive(Component)]
struct Counter {
    pub value: i32,
}

fn increase_counter(mut counter: Query<&mut Counter>) {
    counter.single_mut().value += 1;
}
```