'Space Station 13': Transitioning to Open Source

Kai Richardson, FOSSY 2023

Slide 0 - Cover

Hi everyone!
I'm Kai Richardson, a Maintainer for Goonstation, a flavor of Space Station 13.
I guess, has anyone here actually heard of it?
Great, so let's dive in. This presentation is on a Windows laptop, by the way.

Slide 1 - Introduction

So, what is Goonstation?

I would broadly describe ourselves as a **game development community** supporting 4ish game servers, with around **120 players** online on average.

Our specific flavor of the game has been open-source for **three years**.

We have around **59 thousand** commits, have over **330 contributors**, and get around **10-20 community** contributions per day.

There's a few asterisks - which I'll get to in a minute.

Slide 3 (many) - What is SS13?

Space Station 13 as a whole can be broadly described as a **top-down** multiplayer round-based role playing game, set on a space station.

The game originally started development in **2003.**

However, not much really changed with the game until **2009**, when the source was first made available,

mainly in the form of **revision 4407** of our codebase.

Slide 4 - What is SS13?

So, here's how it looked in 2009 with revision 4407 - the reason for the name is that we still used SVN back then.

Slide 5 - What is SS13?

Here [*point to /tg/], the larger open-source contributing community forked off.

Slide 6 - What is SS13?

There were more forks…

Slide 7 - What is SS13?

…and even more. This is actually a pretty cut-down version of the family tree.

Slide 8 - What is SS13?

Over here in the yellow [*points to oss] brings us to 2020 - going open source.

Slide 9 - Open Source

After being closed source for 11 years, we open-sourced our code on April 1st, 2020 - we thought it'd be a bit funny.

Most people thought it was a joke, until the next day when the codebase was still there. But, it took a lot of work to get to the current contribution experience.

And here's the caveat of the talk: The Official Goonstation servers aren't 100% open source, depending on how you measure it.

We currently have maintained a small (1-4%) closed-source secret git submodule for mainly a few things: mysterious one-off storytelling events, *some* code for ban evasion detection, and gimmick features for gameserver admins.

You could consider it a **server-side extension** to the official game servers, **or not.**

Slide 10 - Open Source

However, the extreme majority of our code, and **just about everything we do** is open-source, so we generally brand ourselves as such.

For example, **our website**, API and **CI/CD**, and **bots** - all open source.

So, what are the sorts of things we had to do to get where we are, transitioning to open source without a real prior contribution community?

Slide 11 - Initial Decisions [head]

As with any organization going under a big change, there's major initial decisions you have to make. I'll be going over the main things that are necessary for every project.

Also, for the rest of this talk, I'll be this little **P symbol** to indicate if a software or service is proprietary.

Slide 12 - Licencing

The first of these is what license you're going to put the project under.

This can range from something very permissive like **MIT**, to something more restrictive but copyleft like **AGPL**.

For us, our licensing situation is slightly funny. Basically everything in the project is licensed under **CC–BY-NC-SA**. Or, Creative Commons Attribution Non-Commercial Share-Alike.

If you've worked with licenses before, this might sound a /bit funny/.

This family of licenses are mainly meant to license assets like sprites, **not code**. We had the disadvantage of being around for 11 years before open-sourcing, so we **inherited** this interesting license choice.

Slide 13 - Licensing 2

As for choosing a license for your own project - you might be in a similar inherited situation, or you might be in the much more 'fun' boat of having to choose.

This could be a whole talk by itself, and I'm no copyright lawyer, but the basics of the choice is to consider how you want the community to operate.

If you want all of the game (as well as eventual forks) to remain open-source, then maybe something like AGPL could be up your alley.

Or, if you want to be able to make secret content, you might need something more lax. It also depends on your method of actually distributing the game.

Slide 14 - Costs

For example, we run servers hosting our codebase. As we are a completely free game, we don't charge users to play.

However, running all of our server infrastructure and related back end **does cost money.**

Slide 15 - Costs

We pay this by **soliciting donations** from generous community members to help pay for costs - however in the past it was out of pocket.

We personally accept donations on platforms such as **Patreon** and **GitHub** sponsors.

There's also other great services like **ko-fi** which don't have fees. That's especially useful if you're a small project where every cent pays the bills.

But, all of these services are **proprietary** - there's some open-source solutions but you'll eventually have to go through some payment provider.

Slide 16 - Distribution

You also want to think about **where** you're going to be distributing the source for your game.

Ideally, it'd be in a format that people are already **familiar** with to lower the contribution barrier.

We personally use GitHub as that is where the majority of the larger space station 13 community resides,

however you could also replace this with a service like GitLab or host it yourself.

Slide 17 - Communication

You'll need to choose a platform for communication, so it's important to be aware of the different

types of community platforms.

For example, one tailored towards chatting like IRC, or one for long discussions like a forum.

In our case, right now we primarily use **Discord** for communicating and planning and we have a forum set up for ideas and suggestions.

However, Discord is not open source, but you could replace it with a similar program like **Revolt.**

However, plenty of open-source communities purely work on platforms like **GitHub -** it's really whatever you think is best for your community.

Slide 18 - Initial Steps Recap

So, when starting off with a new open-source project, you'll need to consider:
Licensing,
Costs,
Distribution
and Communication
before really building a community.

Slide 19 - Contributor Experience [head]

Now, *after* going open source, the next thing we tackled was the experience of each contributor.

Slide 20 - First Time User Experience

The most important thing we found is that the initial onboarding process for contributors is **crucial** -

if it gets to be too much at once, potential contributors can get discouraged from contributing.

Having it **streamlined and easy to follow** makes it easier for anyone,

even people who have never coded, made a sprite, or worked on a game **at all before.**

Slide 21 - In-Depth Onboarding Guide

One way we've tried to improve the first-time onboarding experience is having a **singular** point of entry: our Development Guide.
It covers the whole **basic workflow,** from downloading git, to editing code and sprites, and then making a pull request.
We aim to have both **simple and** advanced topics covered.
For example, we have **in-depth guides** on how to use a debugger or fix merge conflicts.

Slide 22 - In-Depth Onboarding Guide (Detail)

So, it's a big long markdown document that goes through everything.

Slide 23 - Clear Contributing Guidelines

Another thing we've found to be important for the quality of our game is to have easily-referenceable **guidelines** for each asset type.
This means we have **guidelines** on
basic code formatting,

what **sprites** should look like,
the qualities of **audio** we want, and **more.**

Slide 24 - Clear Contributing Guidelines

For example, here are some of the graphics from our spriting guide. Most of the guides are just a lot of text, though.

Slide 25 - Digestible Onboarding

Finally, we've tried to intentionally create areas where the bar to contributing is a**s low as possible.**
Every few months (we tried to do monthly but it's a lot of work) we hold a **small contest** to contribute **small** pieces of content.
For example, a very small 3x5 room used in **procedural generation** or the ingredients in a wacky soda from a vending machine.
I believe these small contributions let non-traditional contributors **make their mark** on the game, and demonstrate how easy it can be to contribute anything at all.

Slide 26 - Controversial Changes

Eventually, players/contributors will **disagree** with maintainer decisions.

These often will be things like **balance** changes, **removal** of features, or simply **not merging** something.

This is fairly **inevitable**, as not everyone agrees on everything!

However, it's important to handle these decisions transparently.

Slide 27 - Town Halls

For example, a popular way for development teams to get feedback from players has been through the use of **frequent town halls**, operating as q&a sessions.

But, honestly, it **hasn't really worked** out for us.

We've found that they put a **lot of burden** on our volunteer team, as a handful of people need to spend potentially hours answering **duplicate** questions.

We've also tried an **asynchronous** version where players could submit their questions ahead of time but that **still took hours** to craft answers to each and every question.

Slide 28 - Communication

One way to potentially head off players being disappointed about changes not making it in is to be clear about **expectations**, how maintainers will communicate how they feel about projects.

One way we've found to help with this is to give **continuous feedback** on proposed features.

Slide 29/30 - #Player-Projects

We use Discord **Forum**s for this, but this can be implemented using traditional forum software as well (we just wanted to consolidate).

Essentially, we have a **subforum** where contributors can **create** threads about their projects.

This also acts as an **official venue** for contributors to **advertise** their projects and perhaps attract co-contributors.

Slide 30 - Player Project Difficulties

However, this idea took some work to become especially useful to us.

At first, the venue was being used as a **dumping ground** for ideas - which while they might be interesting, there was just nobody working on them, which wasn't really the point of the space.

Our solution was to impose some small restrictions like having to have a somewhat fleshed-out design document, so people need to commit at least a little bit to an idea.

Slide 31 - Design Document

For the design document template, there's a few main sections.

The first, the **abstract,** is a few sentence pitch for people new to the idea.

Then we have **goals and non-goals:** things you want, and don't want.

The **content** section is all about the actual implementation, whereas the **alternatives** section concerns what you'll do if things aren't feasible.

The f**inal and most important section** is about what potential changes you could foresee happening if people disagree with parts of your idea.

Slide 32 - Success!

So, here are some examples of projects that have gone through this process.

We've had **a map** added that is mainly worked on and maintained by a single contributor.

We've had **multiple contributors** come together to a whole gamemode, reworking so much foundational code in the process.

Right now, we have a **fishing expansion** that is being sponsored and shepherded by a maintainer.

Slide 33 - Agreement

At the end of the day though, some people **just won't agree.** There might just be a foundational incompatibility in design philosophy.

This is part of the beauty of **open-source** - anyone can start a fork.

For example, people have used our codebase as a curated persistent storytelling platform, an experimental testbed, and more.

Slide 34 - Contributor Recap

xxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxx
yyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyy

zzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzzxxyyzz
xxyyzz

Slide 35 - Maintainer Experience [head]

Now, I want to talk a little bit about how we've lessened the work maintainers need to do, reducing overhead.

Slide 36 - Combating Maint Burnout

This being a **completely free** game, we don't really have funds to pay contributors with - while we do collect donations for server costs, we don't take out any money to pay ourselves.

So, we can't really expect nearly the amount of work someone would expect from an actual employee of a company - everything's on a **purely volunteer basis.**

If nobody wants to do anything it just **doesn't get done.** This has resulted in a saying, **'when you code it'.**

It's especially critical for contributors to not burn themselves out and overextend themselves.

Slide 37 - Feature Freezes

Feature freezes are opportunities for a lot of things.

For those unfamiliar with the term, essentially that means for some period of time (we normally do a month), **new features aren't** added to the game.

Traditionally this allows contributors to focus on **fixing backlogs** of bugs that might not otherwise get addressed.

However we also use them as opportunities to work on **large-scale refactors** that would disrupt the normal flow of PRs immensely.

And finally it's an opportunity to **take a break** and prevent burnout.

For example, we usually have one during December because everyone is off for holidays and spending time with family anyways.

Slide 38 - Backlog

And so as far as merging and reviewing PRs on a **timely basis**, it can definitely be **difficult** with an all-volunteer team.
People's lives just get busy.
One way to help speed up this process is to let the **larger community** help out in small but significant ways.

Slide 39 - Triage

One such way is by having an extremely **easy-to-join** 'triage team.'
On the GitHub platform, members with triage permissions can do basic tasks like **labeling** pull requests and issues,
requesting reviews, and more.
We set up a **simple form** that just asks for their game and GitHub username, and **that's it.**

By keeping the barrier low, we enable more people to help out.

Slide 41 - Organization

With a triage team, it's also important to have labels that matter.
In our case, we have 3 main types of labels for our issues and pull requests: Areas, Categories, and Status. Kudos to Bevy for this system.
The **Area** defines the part of the game it impacts, like the medical system vs an AI change.
Categories explain if it's a bug, a rework PR, a new feature, etc.
Status is used to mark PRs that have a merge conflict, open for adoption to another contributor, and more.

Slide 42 - Organization

One such tag that helps decrease the maintainer workload is the '**Ready for Final Review**' status tag.
A maintainer or triage member can put that label on a PR to indicate that it's been reviewed by a few people (such as community members) and just needs one more look over.
This is one of the ways we help **encourage people** of any skill level to help out and do community reviews.

Slide 43 - Maintainer Recap

So, to recap:
To help with maintainer **burnout,** it's important to not overextend yourselves and take **breaks**.
By enlisting the greater community to help, whether that be **labeling** issues or **engaging** in community review, there's less work for everyone involved.

Slide 44 - Closing Thoughts

It definitely was a **challenge** to get to our current contributor experience, but definitely **achievable** after lots of work from the community and the team.
By **trying out ideas** from the community, you can help make it a better place for everyone to collaborate.
We're **really glad** we moved to open-source - we've had so many great collaborations happen as a result.

Slide inf-2 - Acknowledgements

I would definitely like to thank my amazing team and the community for this wonderful journey into the world of open-source.
If you're looking for more thoughts on this subject, I highly recommend checking out Alice Cecile's talk from RustConf, or my fellow developer Pepper's talk given at GDC last year.

RustConf 2022 - Your Open Source Repo Needs A Project Manager by Alice Cecile
'Space Station 13': Behind One of the Largest Open Source Games

Slide inf-1 - Attribution

Here's some attributions for the images in this talk…

Slide inf - Closer - Q&A

Thanks! The slide deck and everything can be found on my GitHub.

I think we have time for a quick Q&A if there's any questions?