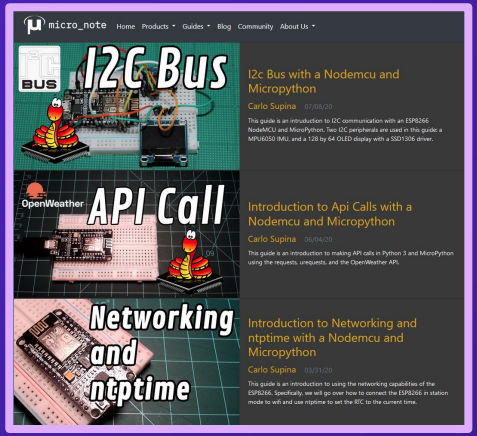
How to Revive a Dead Rust Project

Carlo Supina
Micah Tigley

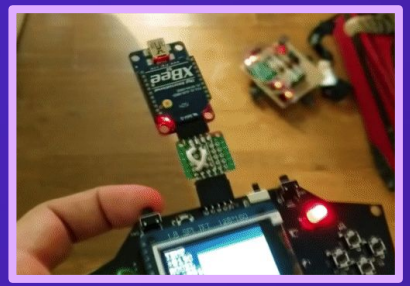
About Carlo

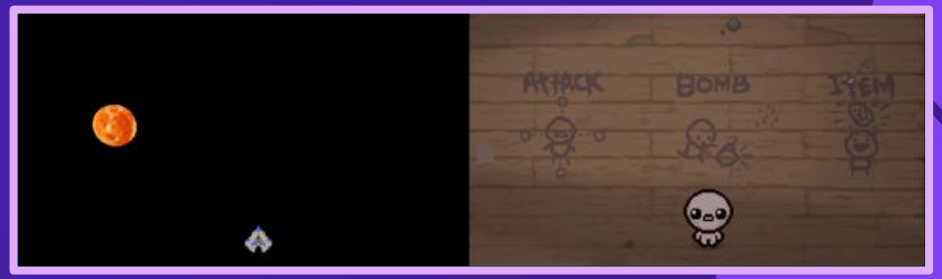
- Welder
  - Pursuing AWS certification
- Runs education company
  - micronote.tech
  - Micropython guides
  - Videos and articles
- Tinkerer and Designer
  - Electronics
  - PCB design
  - 3D part design
  - 3D printing



Carlo’s Path into Rust

- Embedded Rust with STM32F0DISCOVERY
  - RC rover with infrared camera
- Kibrarian: Kicad schematic and footprint manager
  - Project for learning “conventional” Rust
- space_shooter_rs: Space Shooter Game made with Amethyst Engine
  - Inspired by *The Binding of Isaac*





About Micah

- Software Engineer at Mozilla
- Background in front-end web development
- Interested in learning Rust and game development in my spare time

Micah’s Path into Rust

- Made contributions to existing open-source Rust projects
- Game development with Rust, Amethyst
  - Learned about ECS
- RustConf talk on game development

Origin of space_shooter_rs

- Making games is a fun way to learn a new language
- Discovered Amethyst through arewegameyet.rs
- Started working on a simple space shooter style game
- space_shooter_rs was chosen to be a showcase game

Encountering Roadblocks

- Initially a project for **learning** Rust
  - Contained mistakes that a person new to Rust would make
  - Unorganized due to being new to ECS architecture of Amethyst
- Game contained large components with redundant data
- Difficult for me to continue contributing to
- Difficult for new people to contribute to
- Aware that refactoring was needed

Teaming up! (Carlo’s Perspective)

- Watched Micah’s Rustconf talk about her experiences with Amethyst
- Thought she would be a great person to collaborate with on space_shooter_rs
  - Passion for game development
  - Knowledge of industry practices
  - Similar experience level with Amethyst
- Reached out after...
  - ...determining if I had time to commit to the project
  - ...I had a plan to catch her up with the project
  - ...I had a rough idea of what needed to be worked on first

Teaming up! (Micah’s Perspective)

- Learn more about working on a game made with Amethyst
- Expand on basic ECS concepts
- Learn game development in a collaborative environment

What We'll be Talking About

- Summary of "space_shooter_rs"
- Refactoring ECS code
- Strategy and planning a refactor

Summary of space_shooter_rs

- Inspired by *The Binding of Isaac*
  - Collectable items that synergize
  - Randomly generated levels
  - Satisfying controls
- Features currently in the game
  - 3 Enemies
  - 13 Items
  - 4 Consumable drops
  - A Store and currency
  - Animations
  - 3D backgrounds
  - WIP boss

Back to the Basics

Quick ECS summary:

- Entity-Component-System
- Favors easily composable objects
- Data-driven approach

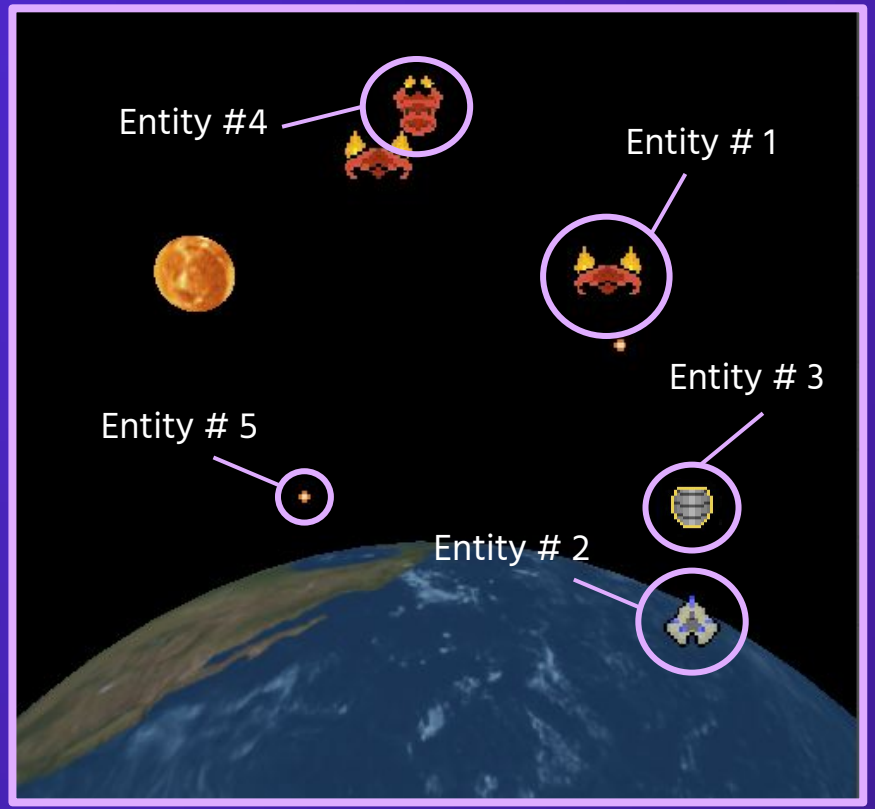
Entity-Component-System

Entity, often represented with a single ID, can be composed of a number of components

Component acts like a container for data that can describe an aspect of an object

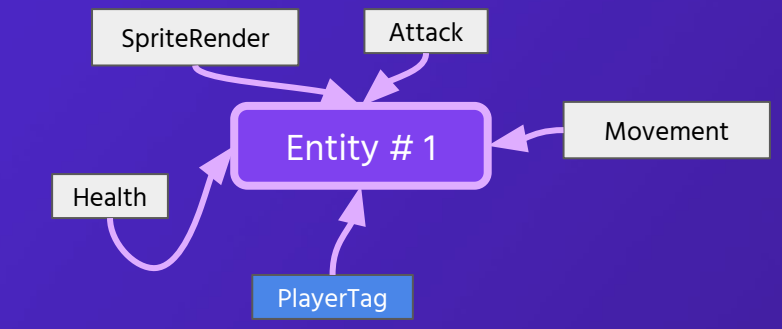
A **system** is a piece of logic that can operate on one or more entities in a game.

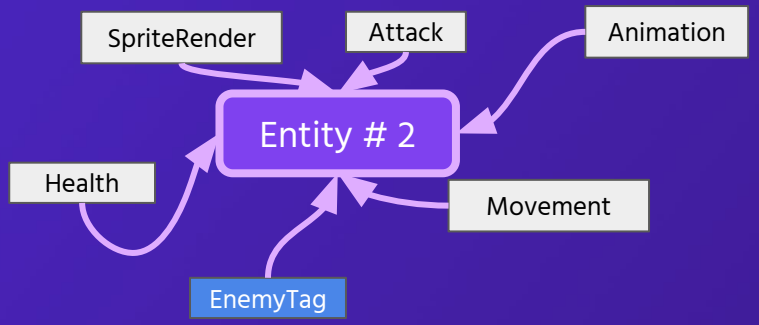
Entity Examples



Entity Examples (continued)

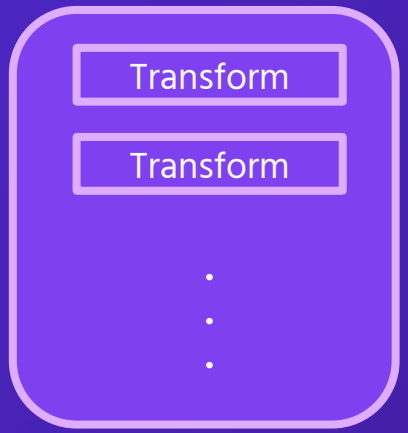
A collection of components make up an entity.



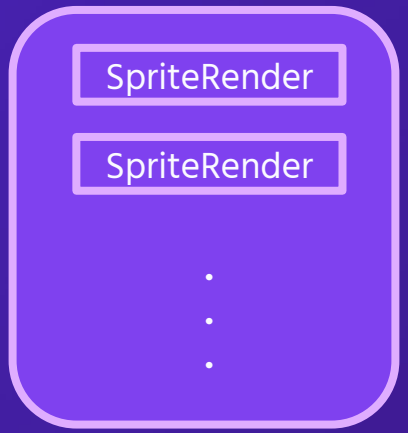


Component Storages

Transform Component Storage



SpriteRender Component Storage



Component storages are responsible for containing and managing components of one type.

Every component has an entity they're associated with.

Entity # 1

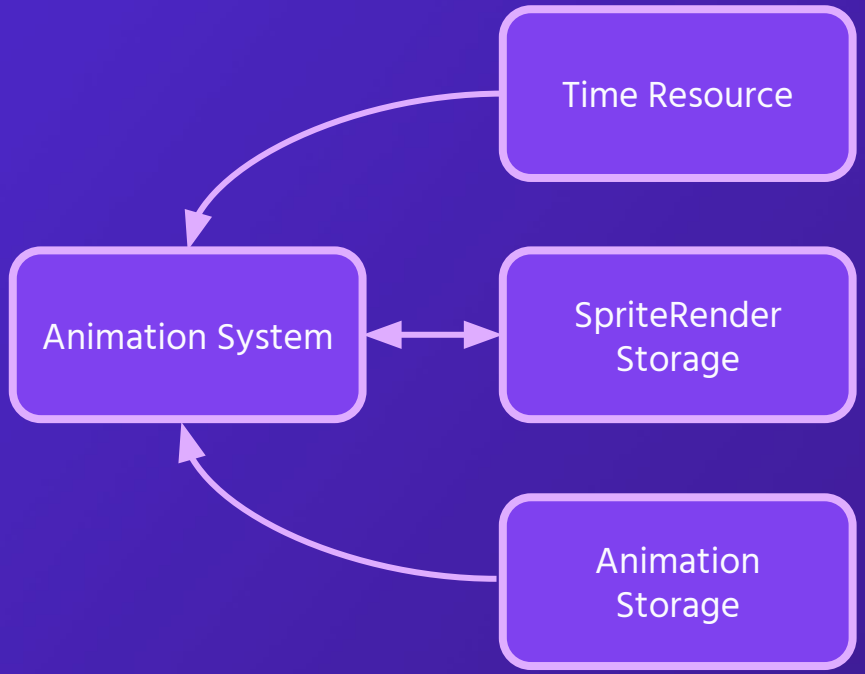
Entity # 2

SpriteRender

SpriteRender

.
.
.

Systems



Animation for “strafer” enemy

Breaking Down Components

Avoiding components that:

- Are bloated
- Had redundant data
- Were difficult to reuse

```rust
pub struct Spaceship {
    pub health: f32,
    pub current_velocity_x: f32,
    pub current_velocity_y: f32,
    pub max_speed: f32,
    pub acceleration_x: f32,
    pub deceleration_x: f32,
    pub acceleration_y: f32,
    pub deceleration_y: f32,
    pub money: usize,
    pub knockback_max_speed: f32,
    pub steel_barrel: bool,
    pub collision_damage: f32,
    // ...
}
```

Define a Set of Requirements

What is the expected behaviour of an entity?

- Helps conceptualize how the components of an entity work together
- Emphasize pieces of functionality of an entity instead of as a whole

Spaceship	Enemy	Item
Animation	Animation	Animation
Blaster	Blaster	Hitbox2D
Health	Health	Motion2D
Hitbox2D	Hitbox2D	SpriteRender
Motion2D	Motion2D	Transform
SpriteRender	SpriteRender	Item
Transform	Transform	
Spaceship	Enemy	

Breaking down Spaceship and Enemy

```rust
pub struct Spaceship {
    pub health: f32,
    pub current_velocity_x: f32,
    pub current_velocity_y: f32,
    pub max_speed: f32,
    pub acceleration_x: f32,
    pub deceleration_x: f32,
    pub acceleration_y: f32,
    pub deceleration_y: f32,
    pub money: usize,
    pub knockback_max_speed: f32,
    pub steel_barrel: bool,
    pub collision_damage: f32,
    // ...
}
```

```rust
pub struct Enemy {
    pub health: f32,
    pub current_velocity_x: f32,
    pub current_velocity_y: f32,
    pub max_speed: f32,
    pub acceleration_x: f32,
    pub deceleration_x: f32,
    pub acceleration_y: f32,
    pub deceleration_y: f32,
    pub allied: bool,
    pub enemy_type: EnemyType,
    // ...
}
```

Spaceship and Enemy component had redundant data for motion.

Motion2D Component

```
pub struct Motion2DComponent {
    pub velocity: Vector2<f32>,
    pub acceleration: Vector2<f32>,
    pub deceleration: Vector2<f32>,
    pub max_speed: Vector2<f32>,
}
```

Entities with motion (movement, speed, etc...) should have a Motion2D component.

Containable Systems Example: Items

- Core progression in the game is acquiring items
  - Items can be purchased from the store
  - Modify rules of the game to advantage the player
- Items provided a unique challenge to implement
  - Items meant to affect every part of the game
  - How do you keep the systems lean if they all need to know if an item was collected?

Containable Systems Example: Items

Sharply increases fire rate of blasts

Collision with player detected in collision system

SpaceshipItemCollisionSystem

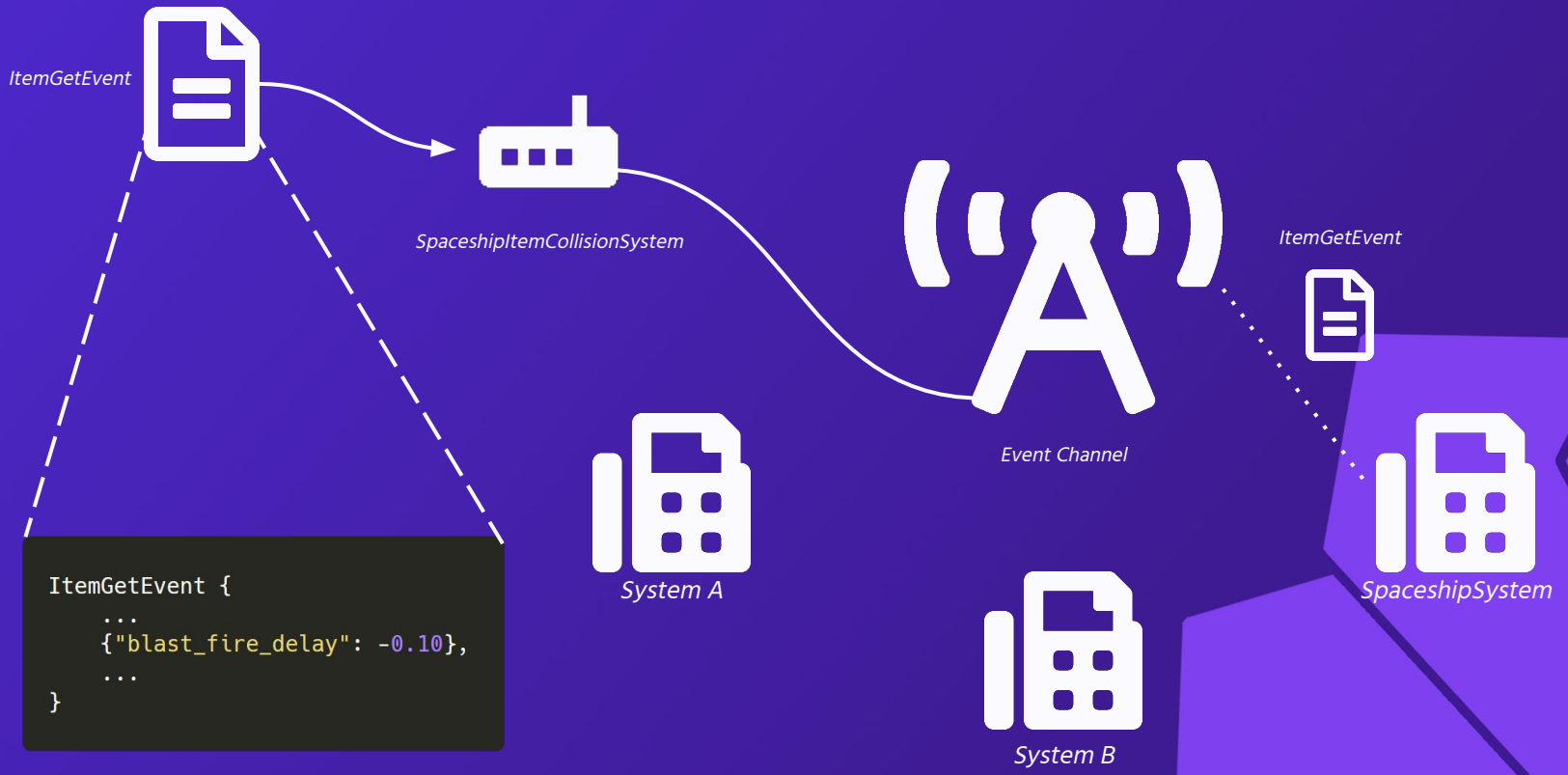
?

Item attributes added to player in spaceship system

SpaceshipSystem

```
"frequency_augmentor": (
    item_component: (
        stat_effects: {
            "blast_fire_delay": -0.10,
        },
        price: 8,
        sprite_index: 17,
        name: "frequency_augmentor",
    ),
),
```

Event Channel: Analogy




```
ItemGetEvent {
    ...
    {"blast_fire_delay": -0.10},
    ...
}
```

Containable Systems Example: Items

Sharply increases fire rate of blasts

Collision with player detected in collision system

SpaceshipItemCollisionSystem

"Item Get" event channel

ItemGetEvent

Item attributes added to player in spaceship system

SpaceshipSystem

```
ItemGetEvent {
    ...
    {"blast_fire_delay": -0.10},
    ...
}
```

Progress Beyond Writing Code

- Collaborative coding practices
- Writing documentation together
- Communication
- Weekly meetings

Collaborative Coding Practices

- Originally was not using Github to full extent
  - Used as a place to store and distribute the files
- Learned about collaborative tools from Micah by following her lead
  - Issues
  - Branches
  - Pull Requests
  - Code Reviews
- Kept code reviews public even though we have a direct communication line
  - Keeps decisions transparent

Documentation

- README.md as an entry point into Space Shooter
- mdBook
  - Preview book at: amethyst.github.io/space_shooter_rs
  - Contributing and Code of Conduct guidelines
  - Guides for adding new items

Append Sprite to Spritesheet

Once your item is designed and you have received feedback on your submitted item issue, you can begin putting it in the game!

Refer to the Contributing Guidelines section for details on how to fork, run, and edit the game on your machine.

If you do not have art yet, or you don't want to contribute art, that's okay! There is an item placeholder sprite already on the spritesheet that you can use for your item. It looks like this (but scaled down to 14 by 14 pixels on the actual spritesheet):

If you do have your own pixel art, you can add it to the items spritesheet located at `assets/texture/items_spritesheet.png`. Add your 14 by 14 pixel sprite for your item to the bottom row at the right end.

Save the spritesheet containing your new item, overwriting the existing items spritesheet.

The next thing you need to do, when adding your own pixel art, is edit `assets/texture/items_spritesheet.ron` to tell Amethyst where the new sprite is located. The vast

Preview the book at: amethyst.github.io/space_shooter_rs

Weekly Meetings: Discussion of Goals

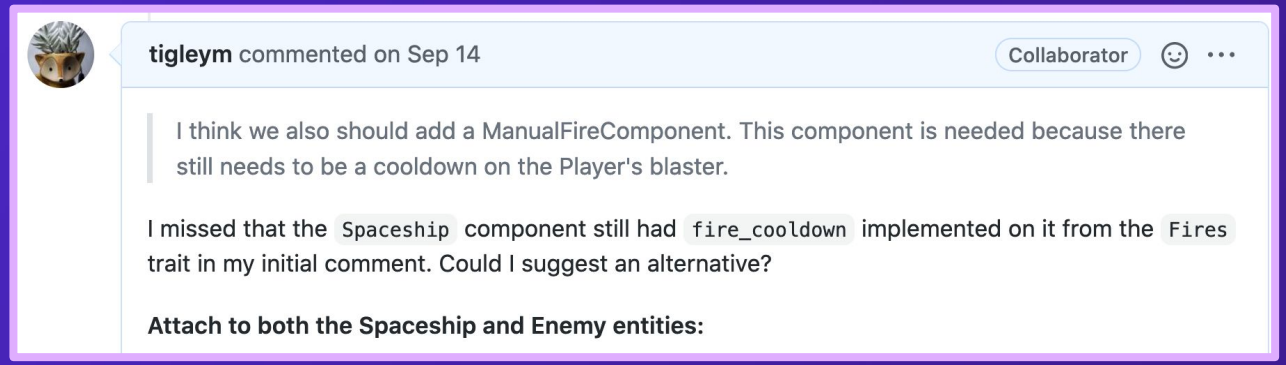
- Short term goals
  - Example: Refactoring a bloated component into multiple, smaller, more general components
- Long term goals
  - Example: Adding a “boss” enemy to the game, and what kind of components, entities, and systems it would require
- Project Goals
  - Discussion of the future for space_shooter_rs as a project
  - What level of documentation should we have?
  - Do we plan on selling a version of the game?

Weekly Meetings: Sharing Ideas

- Some ideas are bad, but **all ideas are worth sharing**
- Examples:
  - Character abilities
  - Items
  - Bosses
  - Game structure

Discussions on Github

- Issues
- Code reviews



tigleym commented on Sep 14 Collaborator

> I think we also should add a ManualFireComponent. This component is needed because there still needs to be a cooldown on the Player's blaster.

I missed that the `Spaceship` component still had `fire_cooldown` implemented on it from the `Fires` trait in my initial comment. Could I suggest an alternative?

Attach to both the Spaceship and Enemy entities:

Issue regarding weapon components:
https://github.com/amethyst/space_shooter_rs/issues/49

Direct Messaging

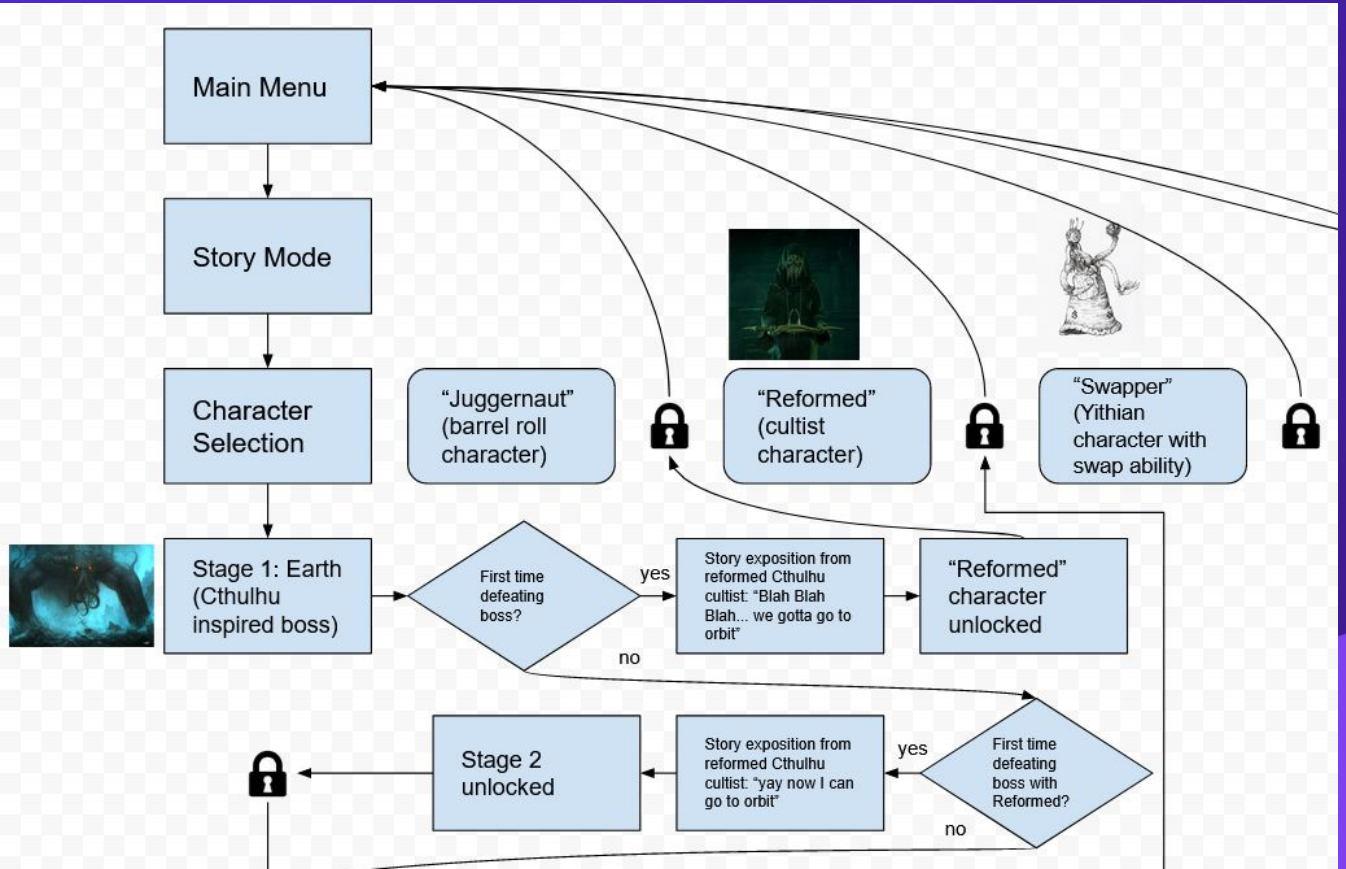
- Less relevant
- Ideas not ready for public
  - Feature ideas not relevant to main goal (refactoring)
  - Off-topic implementation ideas

Informal Documents

- “Current State” Document
  - Game as it currently was
- “Ideal State” Document
  - Ideal version of the game
  - Later transformed into casual place to share ideas

- “Missile” enemy
  - rotating hitbox
  - always tries to face player and moves toward them
  - if it makes contact with player does immense damage
  - technically an enemy, but could be absent from enemy pool, and instead spawned from other enemies instead of ordinary blasts
  - “Enemy” is a temporary classification for missiles
- Loading screens - currently “planets” load in a few seconds into the game depending the speed of your system
- Add the ability for the gamemaster to automatically generate levels
  - ron data file for game master to either explicitly specify a level or have parameters to use for automatic level generation
  - MOD support.
- When refactoring Spaceship (into PlayerComponent) keep in mind that in the future we may want to add local co-op.
  - maybe online co-op?
  - RON file for player inputs?
  - Game controller support?

Formal Documents: Flowchart



Formal Documents: Minimum Viable Product(s)

- space_shooter_rs as a showcase game
  - Just enough content to be an example for others
- space_shooter_rs as a fully released game
  - Maximize fun
  - Balance the game to be fair
  - Story

In Summary

- Engaging regularly in open discussion is beneficial to capturing project progress
- Documentation is important for solidifying knowledge about project architecture decisions
- Share ideas regularly to keep everyone on the same page

Call to Action

We're happy to help anyone get started with contributing to Space Shooter! If you interested in helping out with...

- Code
- Art
- Documentation

Feel free to reach out!

Carlo: cdsupina@micronote.tech

Micah: Discord: Micah#1331

Questions?