Joshua Johnson

CS523

Bart Massey

11/03/2025

# Sonic The Hedgehog Ring Collector Tycoon

Topic Area: Instrumented Simulation

Email: jojohns2@pdx.edu

GitHub Link

## **Project Vision**

### Overview

The goal of this project is to develop a Sonic the Hedgehog inspired idle/tycoon simulation game implemented in Rust. The player's objective is to collect as many rings as possible through clicking and automated systems inspired by the Sonic universe. The project will include a basic GUI displaying the current ring count, available upgrades, and stats about ring production.

### Core Gameplay Mechanics

- Manual Ring Collection: The player can click a button to collect rings manually. Each click increases the ring count by one.
- Automated Collectors: Once a player has enough rings, they can purchase collectors that generate rings automatically over time. Examples include:
  - Knuckles' Help: Knuckles digs up hidden rings periodically
  - Chili Dog Cart: Open a chili dog stand that slowly earns rings

    - Chaos Emeralds: Amplify collection rates by multiplying production speed
- Upgrades and Progression: Each collector type can be upgraded to improve efficiency, speed, or multiplier bonuses. For example:
    - Add toppings to the chili dog cart for faster production
    - Empower Knuckles with special gloves to dig faster
    - Use Chaos Emeralds to unlock “Super Sonic”, doubling all collection rates temporarily
- Log and Display Production Rates: The simulation will log and display production rates, ring-per-second stats, and upgrade efficiency, giving users visibility into the inner workings of the simulation and how much they are earning.

## Technical Design

- Language: Rust
- Crates: Binary crate, with possibly one library crate to abstract the simulation logic
- LOC: 500-1000
- Dependencies:
    - egui for the GUI interface
    - tokio for simulation timing and async ring generation
    - serde for save/load functionality (may not be implemented)
- Architecture
    - Core Simulation Module: Defines collector structures, rates, and upgrade logic

    - Game State Module: Tracks total rings, unlocked collectors, upgrades, and time-based ring accumulation
    - UI Layer: Displays current stats, handles button clicks, and renders upgrade menus

<u>Concerns/Open Questions</u>

- Newness to Rust: The author is fresh to Rust, so it’s hard to gauge how long the project will take to implement and how many lines of code.
- Learning Dependencies: The project will require learning how to interface with new dependencies for the GUI and async timing, and may involve switching choice of which dependencies to use.
- Scope: Focusing on core gameplay mechanics before polishing to prevent scope creep. One goal at a time!