

Wizard Quest

Sprint 1 presentation



Team members:

- Emre Acarsoy
- Luca Croci
- Tom Croft
- Will Finney
- Kazybek Khairulla
- Luca Pacitti
- Harry Taylor

Overview

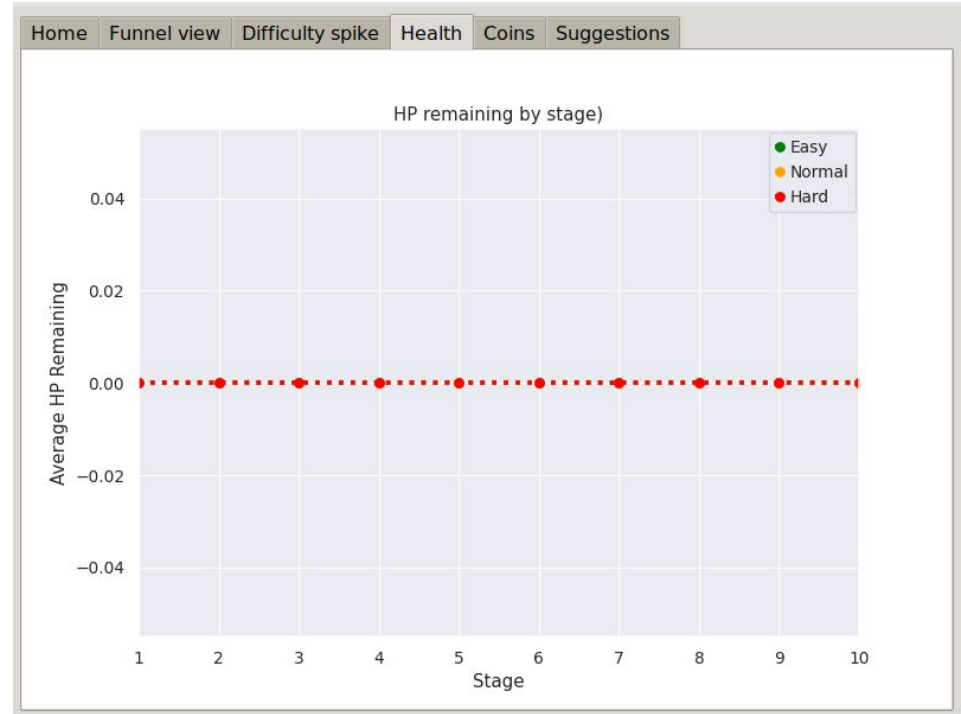
- 1) Problem statement
- 2) Approach and Design
- 3) The Game
- 4) The Telemetry App
- 5) Evaluation
- 6) Live Demonstration
- 7) Recap of Sprint 1
- 8) Look ahead to Sprint 2

Roles

- Emre Acarsoy Telemetry Lead | Java Developer
 - Luca Croci UI & UX Lead
 - Tom Croft Project Lead / Scrum Master | Lead Designer
 - Kazybek Khairulla Java Developer
 - Luca Pacitti Testing Lead | Java Developer
 - Harry Taylor Telemetry developer | Java Developer
 - Will Finney Java Developer | Gameplay Designer
-

Problem Statement

- Game balancing is difficult for designers.
- They often have to rely on their own intuition and small-scale playtests.
- This makes it difficult to identify problem areas where players lose engagement.
- What if we had live analytics generated whilst players engage?



Approach and design:

A tale of 2 applications

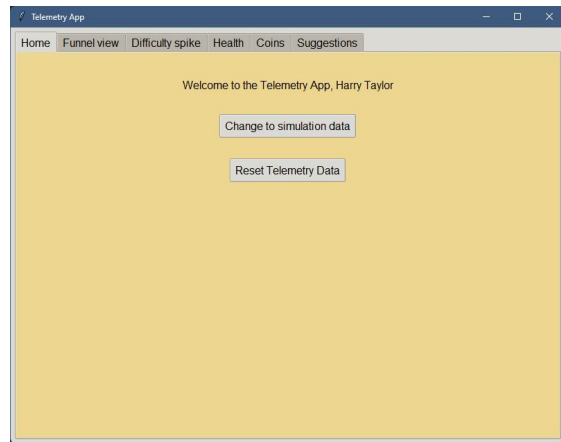
The game

- Wizard Quest, the game the users can play or simulate.



The telemetry visualiser

- Telemetry application, used by the game designers.



Implementation - The Game

Features

- Encounters
- Shop
- Design Parameters
- Damage Types

Components

- Upgrades and Abilities
 - Settings
 - Telemetry
 - Authentication
-

Implementation - Telemetry

Implementation

- Written in Python
- Data manipulation and visualisation libraries:
 - numpy
 - matplotlib and seaborn
- Dynamically generated game balance suggestions
- 5 views: Health per stage, averages per difficulty, etc...

Reading logs

- Events are read and parsed from JSON file.
- Users must sign in to use Telemetry App.
- Only Designers and Developers may view telemetry.
- Clear error messages for malformed JSON.

Measures of Evaluation

Is the game engaging, featuring interesting player decisions?

The player can choose from many upgrades which can improve their odds of success.

Does the system utilise less than 1GB of RAM?

Only 300mb of RAM is needed

Does the telemetry interface have valid spike detection?

Difficult areas are identifiable as spikes

Is the code product easily useable?

UI is intuitive albeit primitive

Are the telemetry app suggestions useful and easily actionable?

Suggestions are actionable, but lack precision as of limited changeable parameters—

—

Demo

Limitations of Sprint 1

The game

- Currently only a CLI, not GUI yet.
- Only 2 stages out of 10 (short gameplay loop)
- Limited variety of enemies, no bosses during encounters
- Only 1 design parameter (starting lives) is adjustable by designers

The telemetry app

- Python Test suite incomplete
- CSV export is not functional
- Not all dashboards implemented
 - Progression Curve
 - Comparison mode
 - Fairness indicators

Looking ahead at Sprint 2

The game

- GUI
- More encounters (and bosses!)
- More comprehensive test suite

The telemetry app

- Additional data views
- Additional dynamic balance suggestions
- Profile page
- More comprehensive test suite