Slippi Stats: Intermediate Planning

University of Wyoming
Senior Design 2023-2024

Beckham Carver bcarver4@uwyo.edu, Kyle Lofthus klofthus@uwyo.edu,
Ben Wilkin bwilkin@uwyo.edu, Zachary Crimmel zcrimmel@uwyo.edu,
Michael Stoll mstoll3@uwyo.edu

What has changed:

The biggest change/hurtle we have encountered is that the official Slippi team has taken great initiative on implementing their own local DB and accompanying improved stats features. Currently they've made at least 82 related commits in the two weeks since they notified us of their goals in pull request #400 and #397. This pull request essentially encompasses our original MVP with some of the simpler portions removed, but likely to be added by the official team in a short timescale. This has made our team take a pause and consider our options moving forward, we discussed what we consider our four primary options

1. No change, stick to the plan and make our own anyways
2. Refocus on adding new functionality to the official implementation
3. Pivot to making Slippi compatible AI players/analysis tools
4. Nuclear option, new project *(ex. Smart Home integrated TV/wallpaper app)*

Our Original MVP is included, this also applied to options 1 and 2 for responding to the official Slippi team's new direction. Fleshing out the latter half of the MVP, and utilizing ideas discussed in our brainstorm document would fit more into refocusing. By focusing more on the statistics we would also likely be able to focus more on UI/UX design, creating a query system and effective visualizations.

Our team's next favorite choice is to pivot towards AI and player analysis. These were originally discussed as stretch goals. After working with and being intimidated by the Slippi-Launcher codebase- we have begun to feel less optimistic about living up to its standards. For this reason, we discussed pivoting towards AI and utilizing libraries like libmelee to create a standalone application for creating/training and playing against AI. Combined with the database and statistics features now on the roadmap, we could also interface with the DB to inform parameters for training models, or modifying/weighting more traditional state machines. As this application would be standalone we would be able to take a more 'if it works ship it' mentality.

The nuclear option, our team has briefly discussed alternative projects. Such as creating a multiplatform smart TV application for displaying locally stored images, and integrating smart home/IOT features. We decided that it would be best to create a 'hello world' level implementation of one of the other choices before we set sites for an entirely new project.

Next Steps:

Our next steps are to eliminate our above options and truly settle on the best path. This could mean sticking with our original MVP or creating a new one. We will do this by testing the waters on each subject so we know what we're choosing or stepping away from ie. create a hello-world for each of the above 4 options. We have already done so for options 1 and 2, and found the codebase for Slippi-Launcher very daunting- this can be seen as a pro or a con. For option 3 we plan to create a basic script in libmelee to control a character and take a look at any existing attempts to apply .slp files to machine learning models.

Ultimately we need to get boots on the ground quickly so we can say 100% whether or not our existing MVP is the right fit for us. Kyle is currently taking the practical machine learning class, and so he will help us look at the existing AI's and to see if these are waters worth treading. We will also need to reach out again to the official Slippi team to see if they have a specified timeline on database features to know if we can plan option 2 around their schedule easily. If we decide to pivot or change our project entirely we will need to redo the project and MVP documentation accordingly.

Original MVP:

TLDR:

- Implement a local SQLite database into the Slippi-Launcher to track post-game statistics from .slp files. Utilize this database to create a statistics & replays page to show cumulative statistics, and persistently update them.

Database:

- Store the resultant data from the existing parsing tools and the replay browser output
    - Game start .slp structure
    - Game info .slp structure
    - Game end .slp structure
    - Metadata .slp structure
    - Output JSON from 'show stats' button
        - Offense: Kills, damage done, openings
        - Move count: rolls, air-dodge, spot-dodge
        - Hit count by type: neutral, counter, trades
        - Inputs: inputs/min, L-cancels
        - Kills: timestamps & moves
    - Opening+conversion timeline JSON
    - Computed winner
- Easily maintainable
- Support migrations
- Support indexing
- Type-safe
- Slippi Team Recommendation: Kysely (query builder)
    - Type-safe sql queries
    - Simplified SQL syntax in code
    - Supports migrations
- Technical aspects:
    - Avoid running queries on the main thread
    - Create good queries that scale well with DB size
- Previous DB attempts:
    - https://github.com/project-slippi/slippi-launcher/pull/384
        - **This is the closest thing to be mergeable, but:**
            - Tests not implemented properly
            - It's not behind a feature flag (only recently added to the codebase)
            - Runs on the main thread
    - https://github.com/project-slippi/slippi-launcher/pull/364/
        - Does use a worker so it's not on the main thread
        - But it uses raw sql

Cumulative Stats:

- Display basic multi-game statistics:
    - Query stats listed in database divided respectively
    - Ie. Win Rate by characters & stage, playtime, character game count etc.
- Display single game statistics when selected, fetching from DB

- Create in its own module, ie: src/stats
- Minimal logic for calculations stats on the renderer side (react/tsx files)
    - May have to migrate current calculations to backend
    - Renderer should fetch stats from the Backend/DB and render
- Only display games with the currently logged in user present in them
- Only display stats played through the Slippi Launcher, no uploaded console games
- Ignore .slp files that do not contain player metadata

Stretch goals:

- Improve performance of the replay browser
- Queries, develop an advanced search function, accessible by users
    - https://github.com/project-slippi/slippi-launcher/issues/222
    - Add custom filters for finding games based off character or stage, player names, etc.
    - Expose endpoints for 3rd party applications to query the database.