For the Roam Team

📲 🔗 github.com

Matt and Prithvi currently have access. Feel free to fork the code to the internal repo under your org

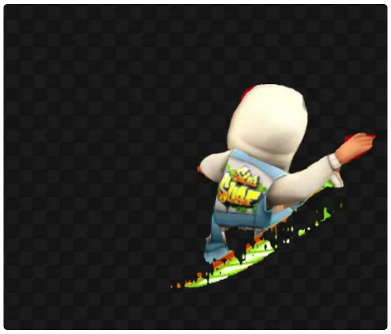
See README.md for description of code, data, and schemas. Make sure to include @README.md into your Cursor chat whenever you're asking general questions about the codebase (it's written in a way to tell AI where in the repo to look for to answer your question)

TODOs before production:

- in `.env`, change the `SIEVE_API_KEY` and pay for sievedata.com
- Make a data storage bucket for raw data (e.g. on cloudflare) and explain to crowdworkers how to upload the data there OR write the code to immediately start upload to the bucket e.g. when the user presses Ctrl+D in the terminal
- Change all the `PROD` variables to `True`
- Run `./build.sh` — that will build the app which you'll then give to crowdworkers, it'll appear in the `dist/` folder
    - Test that it actually works on windows+android. I only tested it on mac+android
- Make changes to the Instructions for Data Collectors page (marked with TODOs)

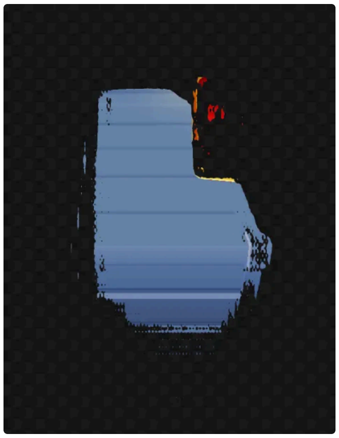
Future improvement directions

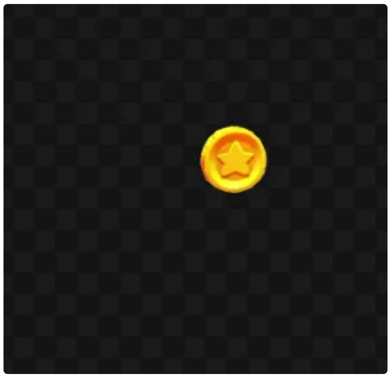
- **postprocess** `action_analysis.json` and `frame_analysis.json` into a format that's suitable for Roam's DB or Alec's tree generation model
  - probably just dump the entire json file as one prompt into gemini with 1M context and ask to summarize.
  - only do it once for every playthrough
- **make the game asset detection model better!**
  - segmentation model (SAM2 deployed here) is near perfect, i.e. it'll cut out objects perfectly

    
  - but Gemini which i use for finding objects often fucks up and i didn't have time to make it much better
    - (reason: it was trained on real-world images, not cartoonish images - so when the game looks like a cartoon, it can't come up with a good bounding box)
    - previously tried: yolo v8, florence-2 (both on Sieve). they're even worse lol
    - using this notebook as basis for my implementation

  → TODO: postprocess all images in the `analysis/segmented_assets` folder, e.g. with another LLM call
  → OR ask the hired crowdworkers to delete incorrect images

  examples of what it thinks is a "Gold Coin":





- **get useful data based on timestamps**
  - was outside of my 2-day scope but a lot of low hanging fruits
    - idea: use CV to see how fast the background or character moves based on user actions
    - idea: see by how much everything accelerates as the user progresses thru the game
    - idea: understand camera behavior, e.g. how much it tilts in response to user swipe
    - etc
  - use raw video data (`screen_recording.mp4`) and touch data (`touch_events.log`)
    - see `src.processing.visualize:parse_touch_log()` for log parsing
    - Caution: there are time mismatches between the android phone registering the touches and the video stream hitting the computer. VERY easy to fuck up! I killed like 4 hours reconciling these times. glhf
    - [illegible]