# Whack-a-Mole Design Document

**Thanh Nguyen <nguyentt@mail.gvsu.edu>**

## Approach

The "Mole Sim" Whack-a-Mole project uses a multithreaded approach with a main thread performing the setup and control work, and Mole threads being created by the main thread to perform Mole state changes.

## Structure

Data is managed using multidimensional arrays as a means to manage and store data. The view (as buttons) are organized in a square grid, and a similar two dimensional array is used to hold Mole Threads in a similar XY coordinate. Settings are loaded from a TOML file.

There is a single control thread that Mole threads are created from and are interrupted by.

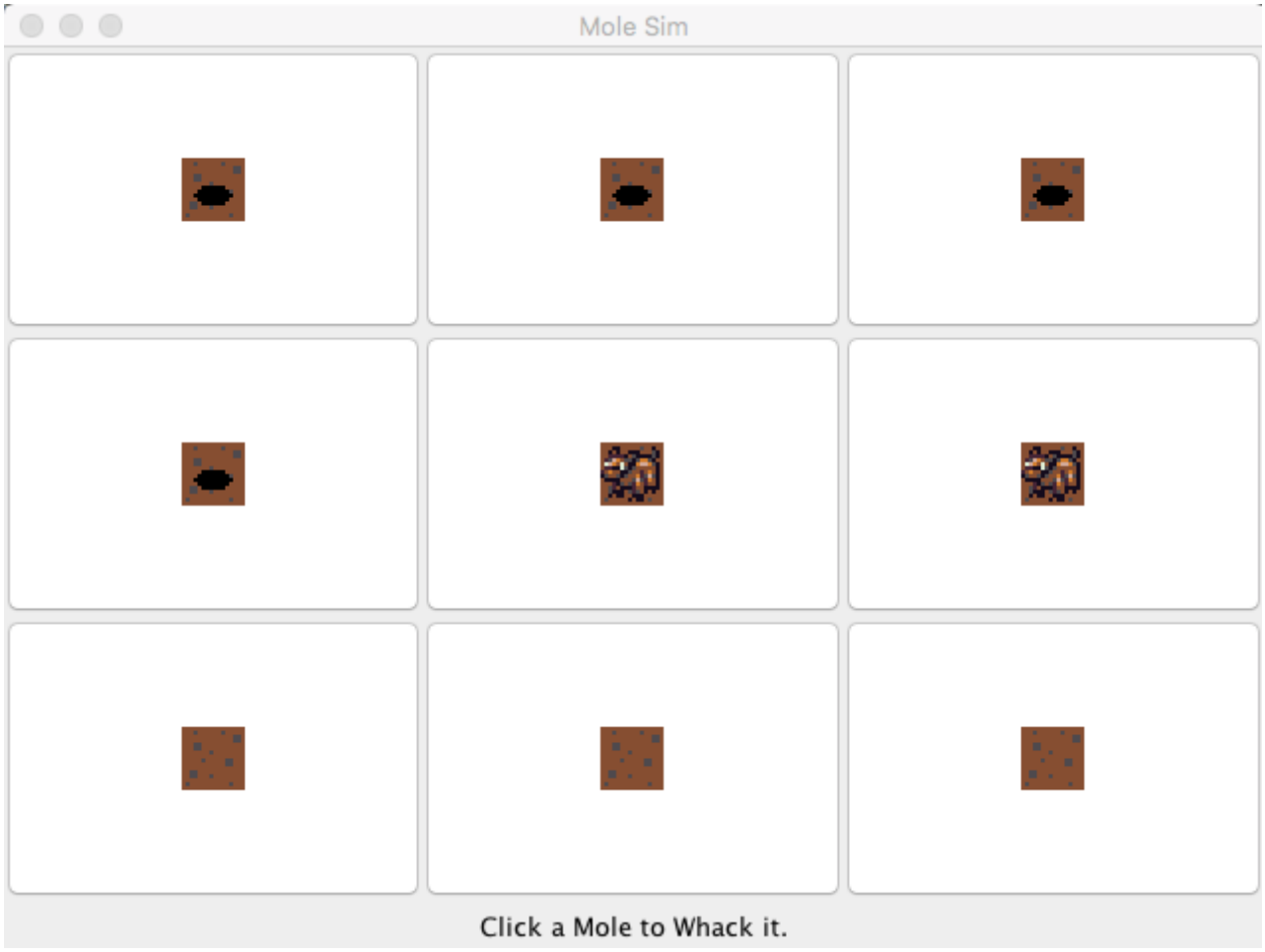
## Limitations

Setting up and mapping threads occurs by iterating through array dimensions, which will likely be significantly slower for larger grids. The simulation assumes that all grids are square. The game state is backed by and stored using the view, making the logic less portable than it should be.

There isn't a victory state, so the simulation can't be won, and no new moles are spawned when moles are whacked. The initial set of active 'popped' moles also appear closely synchronized as well, so they appear and hide at the same time. Another issue that occurs is that the moles that have become active tend to dominate it, and grab a lock on active status again before other moles can get in.

Finally, the TOML library is dependent on Java 8.

## Screenshot




## Credits

Uses icons/art from DawnLike by DragonDePlatino, under CC-BY-SA 3.0 (https://opengameart.org/content/dawnlike-16x16-universal-rogue-like-tileset-v10)

TOML-javalib (https://github.com/TheElectronWill/TOML-javalib), by Guillaume Raffin