AGAMEMNON

Dillon Chen, Olin Gao, Alexander Cox

BROUGHT TO YOU BY



Olin Gao

Globally admired visionary, works 170 hours a week to ensure all points are fairly distributed, and can really thread together an impressive party



Dillon Chen

A leading expert in AI and hacking, makes tiles snap in place with coding and can write Monte Carlo simulations with his bare hands



Alexander Cox

Has landed several internships at Gowogle, rivalled only by Prof.
Blackburn in game design, and can crush any foe by summoning rose trees

SUMMARY OF WHAT WE DID

DESIGN APPROACH

- worked in order of given tasks 1-7
- upon finishing task 7, team members given roles for different parts of game construction
- mix of white and black box testing,
 - -most black box tests already written by Josh Milthorpe
 - -white box tests used near end of project, helped debug
- ai testing done internally and externally

PROJECT STRUCTURE

- src
 - components
 - gui
 - players
 - greedy
 - minimax
 - monteCarlo
 - * Game (abstract)
 - * Agamemnon
 - * Loom

GAME AND FUN FEATURES

USER FRIENDLY GUI

- tried to limit number of menu changes, and ease of changing menus
- easy to understand buttons and interaction
- added pause and mute feature during game play
- able to randomise or choose seed for Loom variant

GAME

- have draggable tiles similar to Assignment 1: return to home if not able to snap to node
- sound effects for placing tiles and warping edges to confirm action
- displays scores of both players which update after each move
- displays picture for when AI is thinking, using multi-threading

FUN FEATURES

- captivating sound track
- intriguing story line
- great animation
- advanced personalisation
- crafty computer opponent
- mute, pause, rematch

AI

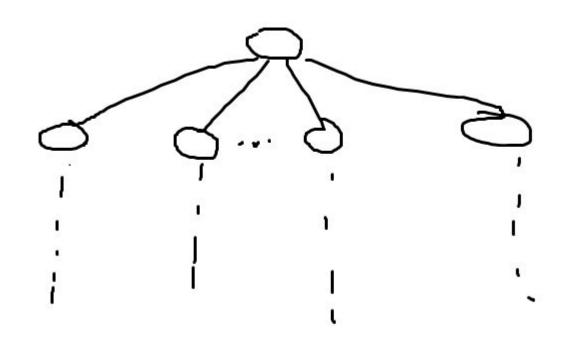
WHAT WE DID

- learning to find an opening strategy (for first four turns where there are no Weaver tiles)
- naive Monte Carlo simulation for remaining turns
- pray to gacha gods we do well in tournament
- luck is a skill

LEARNING

- ran a method for several hours which plays randomly from a starting tile
 - assigned score to each tile and node by average difference in score at end of game
 - ordered best to worst nodes for each tile

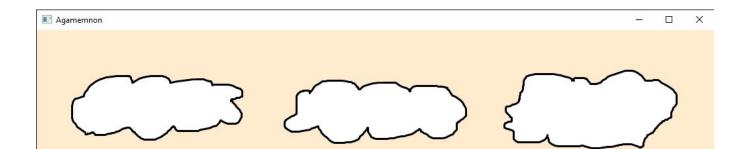
MONTE CARLO SIMULATION



MONTE CARLO SIMULATION

- find a list of possible moves from given state and tiles
- for each move play randomly until end of game for set amount of time (4800ms/#possible moves) and get average score difference
- play move which has highest average score

SCREENSHOTS

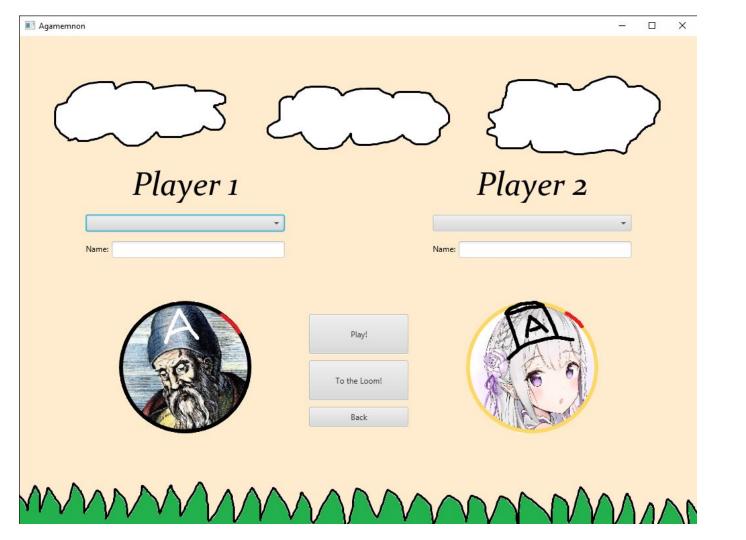


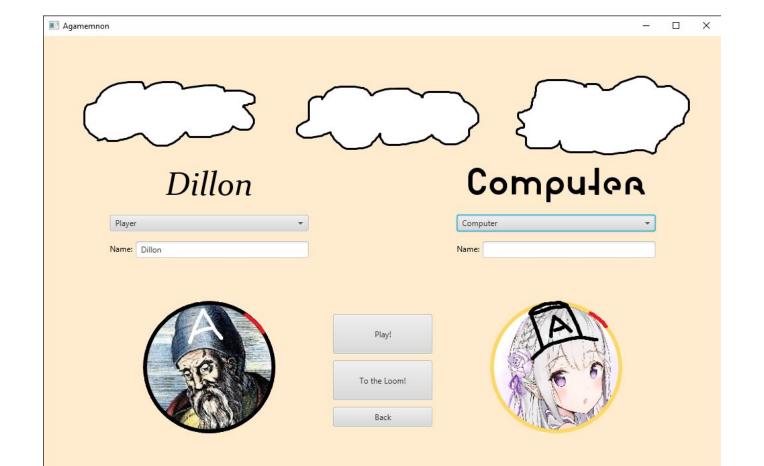
AGAMEMNON

Start

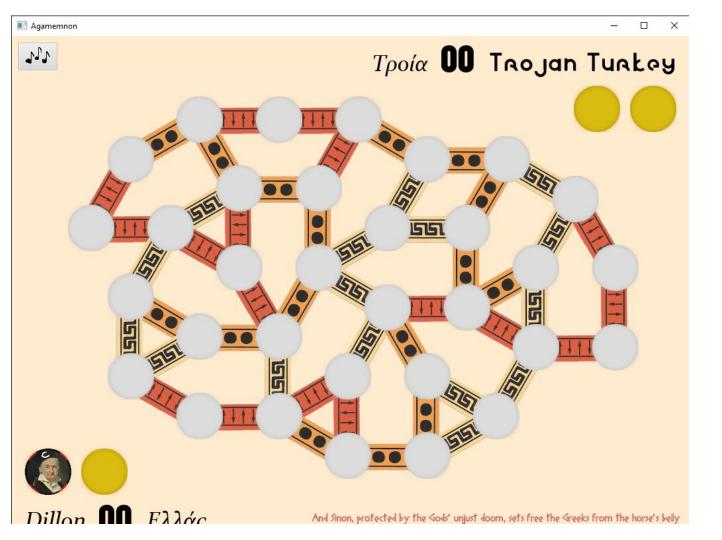
Exit

ummmmmm

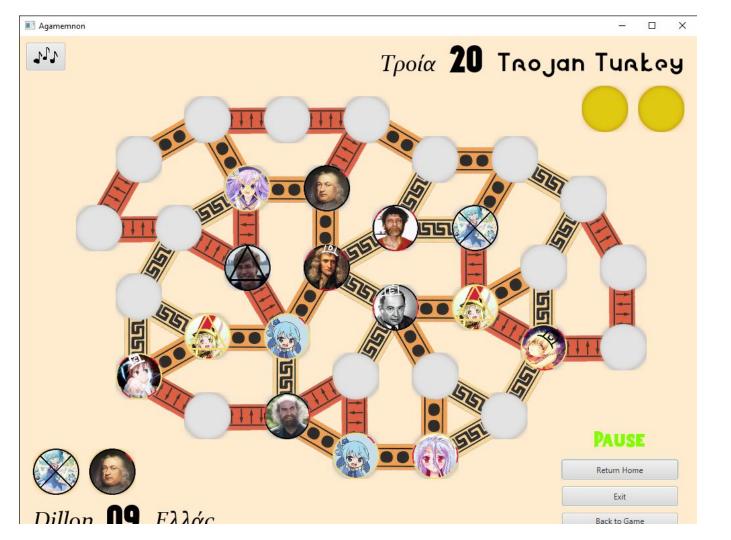




ummmmmmm



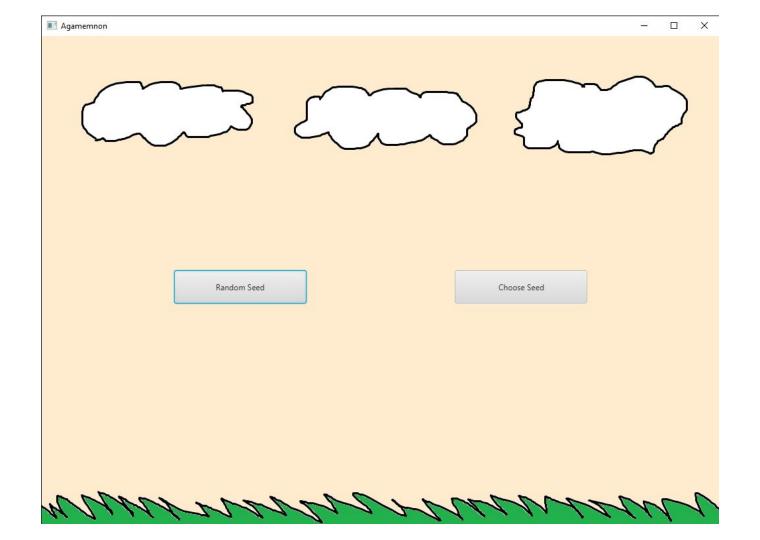


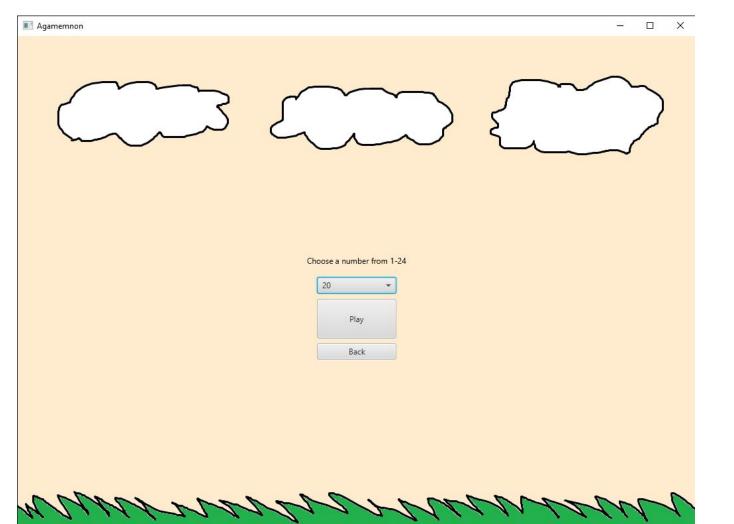


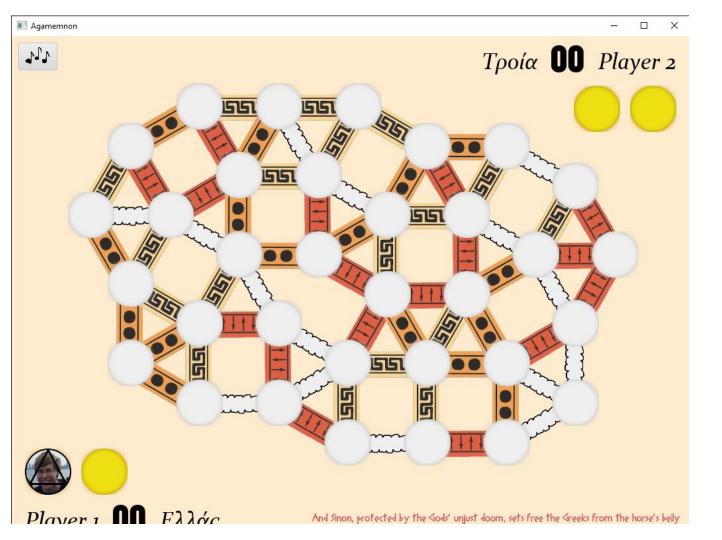


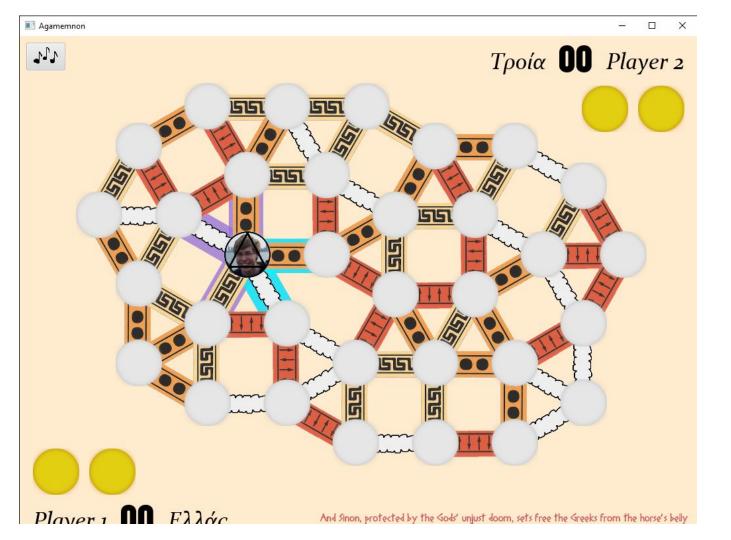












GAME DEMONSTRATION