TreasureHunter created by Gagandeep Thapar (he/they) Aerospace Engineering

RUNNING THE GAME:

- The “main” function is written in the TreasureHunter.java file and will call mainMenu.java. The mainMenu file (once complete) will call Game.java. There are objects that are instantiated in mainMenu so Game cannot be run directly.
- All supporting files have been included in “CSC203_Project3_ThaparGagandeep.ZIP” folder.
- Any JAR libraries used can be found in “SupportingFiles/JAR_LIBS”
- Only the main JAR libraries used in CSC-203 were used

OVERVIEW:

- Aim of the game is to collect all of the coins (win condition) before the enemy entities catch you (lose condition)
- Use WASD or Arrow Keys to move around
- Mouse position relative to the player model will set the direction and will affect the running and attack animations and will set the attack direction
- “Attacking” (Clicking) will slag the ground. Any enemies caught in the slag will “die” and will respawn in the middle of the screen.
- Each enemy is instantiated with a different pathing algorithm: A-Star, SingleStep, and a custom basic algorithm that closes the vertical-gap before closing the horizontal-gap
- The mainMenu will set the player model (and therefore the enemy models)

PROJECT 3 BASE SPECIFICATIONS and COMPLETION CRITERIA:

- World must look entirely different
  - Custom map, textures, and grid setup (continuous instead of discrete grids)
- World changing event
  - Attacking will briefly change the background of adjacent cells to indicate the player attacked
  - Entities caught in the event will “die” and will respawn at the portal with its predetermined pathing strategy
- Use Cursor keys to control the player model
  - Cursor and WASD keys offered to user to control player

PROJECT 3 EXTRA CREDIT SPECIFICATIONS and COMPLETION CRITERIA:

- Use Cursor keys to control the player model
  - Cursor and WASD keys offered to user to control player
- 3 different kinds of entities with unique pathing algorithm
  - One of 4 entities used as player model, other 3 are used as enemy entities
  - Using AStar, Single Step, custom pathing algorithm to control the enemies
- Use Factory Design Pattern
  - Using Factory Design Pattern to create/store coins (collectibles), enemies, obstacles
- Submit short video
  - see “CSC203_Project3Video_ThaparGagandeep.mov”
  - see below for Google Drive Link to video

KNOWN ISSUES
- Because the area A-Star searches is so large, the game can often become incredibly slow when the player model and the A-Star enemy are far apart. This can often be circumvented by either switching the algorithm and not using A-Star or keeping close to A-Star enemy. Changes to the A-Star algorithm to optimize time and space complexity were not attempted.
- M1 Macs have a weird error message get printed to the terminal. The error message does not interfere with the game, but is something to note. It appears to be a Java-issue and its implementation on Apple silicon

LINK TO VIDEO:
https://drive.google.com/file/d/1MBNj_t95I7ZfnIU2OktOehMygUupsXRx/view?usp=sharing