General Info

1. Use +/- key to double/half the simulation speed. In order to prevent accidently breaking the game due to speed, the maximum simulation speed is 16 and the minimum simulation speed it 0.25.
2. Use the 'space' key to pause / and resume the simulation. When the goal is reached, press 'space' key will restart the simulation from initial state.
3. The following external assets are used:
    a. **Stone Floor Texture Tile:**
        i. https://assetstore.unity.com/packages/2d/textures-materials/roads/stone-floor-texture-tile-18683
        ii. Used as the floor texture
    b. **Optimize, SD Kohaku-Chanz!:**
        i. https://assetstore.unity.com/packages/3d/characters/optimize-sd-kohaku-chanz-84992
        ii. Used as the model for the thief, player, and traders.

Player Action Types Design

Generally, the following actions with parameters are used, the parameter is a vector of spice numbers following the order: $(Tu, Sa, Ca, Ci, Cl, Pe, Su)$ as in the specification. And the number if from the perspective of player: e.g. $(-2, 1, 0, 0, 0, 0, 0)$ means take 2 Tu from the player and give 1 Sa to the player.

1. $\boldsymbol{ExchangAction(spiceVector, location)}$:

    Pre-condition: The player must have enough items to exchange. That is, for every i in $[0, 7)$ such that $spiceVector[i] < 0$, then $playerStorage[i] \geq abs(spiceVector[i])$. Also, the sum of spice after the exchange cannot be larger than 4: $sum(playerStorage + spiceVector) \leq 4$

    Action: pathfinding to location and do the exchange

    Effect: $playerStorage += spiceVector$

2. $\boldsymbol{PutAction(location)}$:
    Pre-condition: Player must have at least one spice: $sum(playerStorage) > 1$
    Action: pathfinding to location and put items
    Effect: $caravanStorage[i] += playerStorage[i],\ \text{playerStorage}[i] = 0\ \forall i \in [0,7)$
3. $\boldsymbol{TakeAction(spiceVector, location)}$
    Pre-condition: Caravan must have enough storage $caravantorage[i] \geq spiceVector[i], \forall i \in [0,7)$. And the sum of spice after the exchange of player cannot be larger than 4: $sum(playerStorage + spiceVector) \leq 4$
    Action: pathfinding to location and take items
    Effect: $caravanStorage\ -= spiceVector, playerStorage += spiceVector$
4. Trade Action: deprecated and not used, originally used to implement a more optimized hybrid GOAP.

Player Actions

Let L_{ti} be the location of trader i, and let L_c be the location of the caravan:

1. exchange actions:
    a. $\boldsymbol{ExchangAction}\big((\mathbf{2},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_{t1}}\big)$:
    b. $\boldsymbol{ExchangAction}\big((-\mathbf{2},\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_{t2}}\big)$:
    c. $\boldsymbol{ExchangAction}\big((\mathbf{0},-\mathbf{2},\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_{t3}}\big)$:
    d. $\boldsymbol{ExchangAction}\big((-\mathbf{4},\mathbf{0},\mathbf{0},\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_{t4}}\big)$:
    e. $\boldsymbol{ExchangAction}\big((-\mathbf{1},\mathbf{0},-\mathbf{1},\mathbf{0},\mathbf{1},\mathbf{0},\mathbf{0}),\boldsymbol{L_{t5}}\big)$:
    f. $\boldsymbol{ExchangAction}\big((-\mathbf{2},-\mathbf{1},\mathbf{0},-\mathbf{1},\mathbf{0},\mathbf{1},\mathbf{0}),\boldsymbol{L_{t6}}\big)$:
    g. $\boldsymbol{ExchangAction}\big((\mathbf{0},\mathbf{0},-\mathbf{4},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{1}),\boldsymbol{L_{t7}}\big)$:
    h. $\boldsymbol{ExchangAction}\big((\mathbf{0},-\mathbf{1},\mathbf{0},-\mathbf{1},-\mathbf{1},\mathbf{0},\mathbf{1}),\boldsymbol{L_{t8}}\big)$:
2. Take actions:
    a. $\boldsymbol{TakeAction}((\mathbf{2},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_c})$
    b. $\boldsymbol{TakeAction}((\mathbf{0},\mathbf{2},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_c})$
    c. $\boldsymbol{TakeAction}((\mathbf{4},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_c})$
    d. $\boldsymbol{TakeAction}((\mathbf{1},\mathbf{0},\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_c})$
    e. $\boldsymbol{TakeAction}((\mathbf{2},\mathbf{1},\mathbf{0},\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_c})$
    f. $\boldsymbol{TakeAction}((\mathbf{0},\mathbf{0},\mathbf{4},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_c})$
    g. $\boldsymbol{TakeAction}((\mathbf{0},\mathbf{1},\mathbf{0},\mathbf{1},\mathbf{1},\mathbf{0},\mathbf{0}),\boldsymbol{L_c})$
    h. $\boldsymbol{TakeAction}((\mathbf{1},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}),\boldsymbol{L_c})$
3. Put action
    a. $\boldsymbol{PutAction}(\boldsymbol{L_c})$:

p.s. basically, take actions are designed based on the demands of the traders, however, $2.h$ is designed to deal with the situation that one Tu is taken from the thief (since we can only get even number of Tu from the trader). Note that even if this action was not considered in the plan, the player would still be able to find the goal, only takes longer steps that involve $2.d$. So, this action is more to optimize the actions.

Player State design

$$state = (playerStorage, caravanStorage)$$

Player Goal State

$$goal\ state = (playerStorage = any, caravanStorage = (2,2,2,2,2,2,2))$$

Player Search Strategy

1. **Heuristic:** a weight function based on the value of the spice following the rule:

$$val(Tu) = 1$$

$$Val(S) = (value\ of\ spices\ used\ to\ trade\ S) + 1$$

We can then get a weight vector:

$$w = (1,3,7,5,9,11,29)$$

Then the heuristic is calculated w.r.t. the $caravanStorage$ of a state s be C_s, and the $caravanStorage$ for the goal be $C_g = (2,2,2,2,2,2,2)$

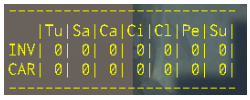
$$h(s) = sum\left(w(i) \times abs\left(C_s(i) - C_g(i)\right)\right), for\ i \in [0,7)$$

2. **Search:** I used a standard *Heuristic Search* algorithm (since the **Heuristic** is not admissible). That is, I consider both the **Heuristic** described above and the action cost. (All actions have equal cost)

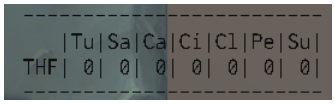
Thief Actions and plan description

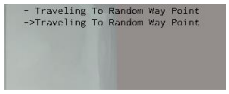
Thief behaviors are designed using behavior-tree-like algorithms. she basically just travels between different pre-set waypoints. Every 5 seconds, it draws a random number n from $[0,1)$. If $n \leq 0.165$, she tries to steal from the caravan. If $n > 0.165\ and\ n\ \leq 0.33$, she tries to steal from the player. Else she does nothing and continuing wondering. Once steals two items, she stops and only travels between random waypoints.

Representations

1. Trader + trader number: 
2. Caravan 
3. Player: 
4. Thief: 
5. Player and caravan inventory: 

6. Player log (read from top to the bottom): 

7. Theif inventory: 

8. Theif Log (read from top to the bottom): 

9. Simulation Speed: