PEAS Description

Performance Measures – Win rate, acceleration of win rate

Environment – Risk Board Game (World Domination standard rules. See instructions)

Actuators – Any valid move for a normal player (see instructions)

Sensors – Can observe the environment internally the same as any human player. The only unobservable characteristics are that you do not know the status of cards you don’t possess.

ODESDA-K Description

O – Partially Observable. (you see the entire state minus the state of individual cards that haven’t been turned in)

D – Stochastic. Risk is heavily influenced by the outcome of die roles, with no way to determine whether or not you will succeed in a particular endeavor. You can however tilt the odds in your favor.

E – Episodic. The Risk board game is an episode that will repeat in full after each game. The outcome of the game has no effect on the outcome of the next game.

S – Static. The environment will not change while the agent is deliberating. You take turns in Risk.

D – Discrete. Everything in Risk can be represented in finite integers.

A – Multi-agent. There are more than one fully capable agents.

K – Known. The rules of the environment are not just known but enforced by the rules.

Hasbro. *Risk Instructions*. Web <https://www.hasbro.com/common/instruct/risk.pdf>

Working Documentation

**Intro - Risk Reinforcement Learning**

This package is meant to bring the environment of the board game Risk into a research setting for AI, particularly reinforcement learning (RL). The package will provide tools for building RL agents, data collection about agents, and tools for debugging agents. This is strictly meant to be used in an academic, research setting and is not allowed to be ported for commercial use in any way.

**The Game**

The version of Risk this package plays follows (for the most part) the standard ruleset of “World Domination” Risk. And can be found in the instruction manual here:

<https://www.hasbro.com/common/instruct/risk.pdf>

A quick overview of the rules and options will be described:

* Territories may be picked by players or dealt randomly
* There are a maximum of 6 of any kind of player
* Turn order options are:
  + Highest Roll goes first, then clockwise (sequentially by player number)
  + Highest roll determines order
  + Input the turn order of players
  + Card set trade ins options
    - Standard 4,6,8,10,15,…,60
    - By one 4,5,6,etc
    - Custom by user input

NOTE: Card set trade in values are independent of the faces of the cards

* + - Cards from defeated players can be set to taken, or not

**Playing the Game**

There are 2 ways to play the game

* Pygame GUI
* Console (GUIless build for training)

As such you can start a game of the Risk RL from either. Use play.py to run the GUI, and play\_headless.py for no GUI. Either can be run with human players or pure computer players.

This first thing that happens is the setting configuration. The location of code for this depends on whether you are using the GUI or non-GUI, but the basic structure is the same and found in config.py. During this portion you will decide upon the ruleset by which to play the game. All rules (except card trade in values, and random territory assignment) are built into the state representation. The territories will then be divvied up based on the chosen setting. If players choose territories the agent will receive a list of valid territories to choose from, and must return one of those territories. This will occur according to player order until all territories are occupied. Each territory occupied will have one troop placed in it for now, and this will subtract from the initial amount giving depending on how many players there are (3:35,4:30,5:25, and 6:20). After all territories are taken each player must place all remaining troops is their territories. This functions the same as the troop recruitment phase at the beginning of each players turn, and each player will be asked a territory to places troops one at a time until all troops are placed.

After the beginning of the game is truly set up, the standard turns will take place until a winner is declared. The turns have three phases:

* Recruitment and placement
* Attack
* Reinforce

During the recruitment phase the player gets troops to place according to the rules. The player will be prompted to place troops into territories one by one. At this time a player may trade in valid card sets to receive troops equal to the current trade in value. The BaseAgent has methods built-in that allow for the detection of unique card sets. The player will be prompted whether or not they want to turn in a card set if they possess a valid one. If the player wants to it must select which arrangement of cards it would like to turn in. Valid unique arrangements are the returned value of BaseAgent.get\_sets(). If the player decided to turn in card sets, they will be prompted again to place troops in territories.

Once the recruitment and placement phase is over, the player may choose to attack a rival territory. The player will receive a list of valid territories to attack from, and must return one of them. Included in these valid territories is a “blank” that if chose no attack will be made and player will skip (or finish) the attack phase.

**Agents**

Agents are designed using the BaseAgent class in agent.py. BaseAgent contains most generic functions that are useful for all agents, such as observing the game and determining valid card sets. Agents with advanced functionality should be subclasses of the BaseAgent. Included are subclasses Human and RL. A Human object can be used to take the spot of a player and make the game prompt the console for input for the decisions made by the player. RL is the built in RL agent that can play by itself.