1. Perform action
   1. Assuming some logic has been discovered for ASP, use this to select regions of the domain within which actions can be performed in (an example would be if a tower is leaning left, it would restrict the domain to only placing blocks on the right).
   2. Choose an action randomly from all allowed action, or from all actions if there is no logic yet from ASP.
   3. The end goal here would be that actions would eventually mainly be chosen just through ASP.
2. Return a reward
   1. Proportional to the number of blocks previously placed in a run OR
   2. A negative value if the tower falls (a block is lower than it was previously). If the tower has fallen reset the simulation and end an episode
3. Update reward function using returned reward.
   1. I’m currently planning on using semi-gradient q-learning, due to the space being continuous.
4. At the end of an episode, use final state of the tower before simulating any physics to learn logic. Not 100% certain how to do this.
   1. I can just put it in as prior knowledge, such as by using the concepts in the paper you sent me not long ago (<https://www.frontiersin.org/articles/10.3389/frobt.2019.00125/full>). So as an example for the ‘lean’ one it would just check the average position of blocks in tower, compared to the actual position of the lowest block, and then if these are different enough classify it as lean. I am not too keen on this approach because I feel it is putting too much logic coded in. I am trying to come up with another method, but currently haven’t thought of one.
   2. I would not be updating the ASP logic after each action, it instead using the final state as I believe

Current issues:

How to represent states – either an image (so output of the simulation screen) or by block locations.

How to decide what features to use for the semi-gradient learning (as far as I can tell features need to be supplied)

Whether or not to put in prior logic (i.e. the lean option I explained earlier).