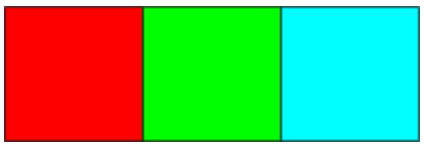
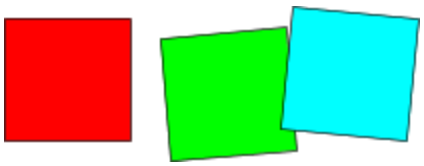
I summarize the discussion we have on Tuesday:

- We want to teach Diana a few new concepts, such as “create a row of blocks”
- We first show to Diana a few demonstrations of a performer doing the action.
- We need to tell Diana of where is the start and end of one demonstration, or we allow Diana to reason on the delay between different demonstrations.
- A few assumptions that Diana needs to make:
    - Assumption 1: If the input sentence doesn’t specify any block in particular, Diana assumes that there is no distinction between block.
    - Assumption 2: Next blocks to move in one episode is different from blocks already moved.
    - Assumption 3: There is a creation of a new concept, object that Diana doesn’t know before, she makes assumption that the new concept is created by the shape of the moved blocks.
- What Diana observes:
    - Move(blue, L1), Move (red, L2)
    - Move(red, L1), Move (blue, L2), Move (yellow, L3)
    - Move(red, L1), Move (yellow, L2), Move (blue, L3)
    - L1, L2, L3 is block transformation (has location and orientation)
    - She might not be able to see what we really intend her to see, because visual input is fuzzy:
    - What we saw as the target result:



    - What Diana saw as the target result:

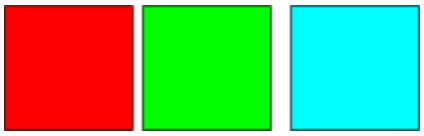


- Let’s call a basic action Diana can do is Move(X,L), where L is a block location and orientation on the table.
- A *target* is a the shape of the result configuration (assumption 3). I think we can find two ways to represent here:
    - Fuzzy shape: the convex polygon of the moved blocks at the end (probably with invariance to rotation).

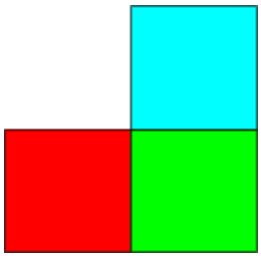
- A qualitative graph-based shape, like this:

More correctly, the target function for learning policy would be a function measuring how similar the result shape to the set of training shapes.
I think both ways of representations have merits, and should be combined.

- The fuzzy representation might not help to distinguish with:



- The graph-based representation might not be able to established (see the fuzzy example), and also it might not help to distinguish with:



RL problem statement

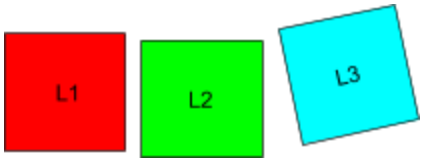
- An *episode* of simulation is a sequence of action Move(X1,L1), Move(X2, L2) …, the target is to match *one* of the final configurations of blocks (just forget about the sequential representation and focusing on the final result).
- A *parameterized policy* is to generate the next block to move (categorically) and the next point/orientation to move to (Linear regression + Gaussian). Because of assumption 1 and 2, the policy is simplified to the next point/orientation (L_n) to move to.
- We can think of the policy as following:
  - π(L_n | L_n-1, …, L_0, $\boldsymbol{\theta}$, **w**) = π_2(L_n | $\boldsymbol{\theta}$) * π_1($\boldsymbol{\theta}$ | L_n-1, …, L_0, **w**)
  - π_1 is a linear regression: $\boldsymbol{\theta} = [$L_n-1, …, L_0$]^T * \mathbf{w}$
  - π_2 is a Gaussian with (small) fixed, predefined variance, so $\boldsymbol{\theta}$ = mean of L.
  - The policy can be straightly used for the command “add one more”, because Diana just need to generate the next action, by applying π_1 with the same set of parameter w and the last **n** locations -> it has the effect of iterative application (it

means that even Diana doesn't observe a row of 4 blocks, she can still generate one more block moving with this policy).

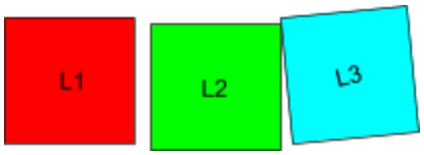
- Using Monte Carlo Policy-gradient method (REINFORCE), we have the following update:
  - $\Delta\boldsymbol{\theta} = \alpha * (\text{reward} - \text{reward\_baseline}) * (L - \theta)$
  - Reward_baseline is average of observed rewards (a better baseline would be one that estimate value-function for this exact state)

Expert correction

- A sample dialog:
  - Diana: Is this correct?



  - Rahul: Could you move blue block **TOUCH** green block?
  - Diana: (move block). Is this correct?



  - Rahul: (Thumb up)
- On the learning side, it equals to an adding of an enforced constraint between Block 2 and Block 3 (Block 2 EC Block 3). There are probably two ways:
  - Add the resulted state as one training example.
  - Add the constraint into the policy.

  Both approaches are quite straightforward.

Notes:
In learning from demonstration (LfD), a human purposely demonstrates how to perform a task or an action, expressly to teach a computer agent how to perform the same task or mission. Learning from demonstration is a specialization of LfO. LfO is a more general learning approach, where the actor being observed need not be a willing participant in the teaching process. (Montana 2011)

Montana, J., and A. Gonzalez. "Towards a unified framework for learning from observation." *IJCAI Workshop on Agents Learning Interactively from Human Teachers*. 2011.
http://www-anw.cs.umass.edu/~barto/courses/cs687/Policy%20Gradient-printable.pdf Policy gradient.