Presentation:

- Reinforcement learning in general - Fu
    - What is it?
    - How does it differ from supervised learning?
    - Rewards to actions
    - Delayed gratification for later rewards
    - Single agent vs multi-agent and large action-space lead to complex learning
- Curriculum learning - Fu
    - What is it - the idea - analogy to students learning
        - Different environments - e.g. new actions
        - Progression function to modify difficulty of environment automatically when learning in 'easier' environment is achieved

Script (Fu):
Let's start with Reinforcement Learning. Reinforcement Learning is one type of machine learning which an agent is placed in an environment and has a goal to reach. The goal of the agent is to learn an optimal series of actions, which is known as the optimal policy, to maximise the cumulative sum of rewards. The interaction of an agent with its environment can be modeled as a Markov Decision Process. (Should we explain the tuple as well?)
Ok, it is an easy bit, everyone knows it.With the basic understanding of RL. Let's talk about Curriculum learning.
Imagine a student is given a task to solve a differential equation , without any knowledge about mathematics, also without any assistance. Certainly he can eventually solve this problem, given an infinite time of trial and error. But "no" the teacher said. " We don't have time." The student only has three years to solve this question. Certainly there must be a way to speed up the learning process. Right?
How about letting a student start on an easy task, 1+1=2, and then given a task to solve elementary algebra, x+2=5, and eventually solving the differential equation. In this way, the student can learn from the previous task and then apply the knowledge onto the next task, and then eventually the target task. This is curriculum learning and applying it to reinforcement learning is becoming popular in this recent year.
.

Let's go back to the student scenario , These tasks, including the differential equation, combine together and we can call it a curriculum. Do you notice the problem right there? In order to let the student solve the differential equation, which is our goal, we need to create several more intermediate tasks to train the student, and we need to create them in a way such that all the tasks should be relevant to the final task. This is the first element of curriculum learning: task generation. These tasks can be pre-defined or generated automatically.
The second element is sequencing. Although we create several intermediate tasks to train the student, we have to order the tasks in a way that facilitates learning . In RL, we sequence the

tasks to minimise the training time of a curriculum, while finding the optimal policy. As mentioned before, tasks can be pre-defined or generated automatically and form a curriculum. These tasks may share the same MDP model , or may involve two agents, or even having different MDP models. For each type of curriculum we have to use different sequencing approaches to give us the best performance or training speed. These Sequencing approaches can be done by humans or automatically..

The third element is transfer learning. When the student completes a task, the knowledge that he gains from this task must transfer to the subsequent tasks . In RL, the tasks may have a different MDP model. We have to transfer knowledge from the source tasks to the target task. These transferred knowledge can be a policy, a value function, or a model.
ALthough there is also some usage of curriculum learning in supervised learning, this is beyond the scope of this presentation.

Let's go back to discuss a curriculum which involves multi-agent in the same environment.

Autocurricula - what is it? - Rayhan

- Introducing multiple agents in an environment already generates a curriculum automatically
    - The actions of the new agents increase the complexity of the environment and encourage further learning
    - Can use single player solitaire vs two players leading to the game of Go as an example

Script (Rayhan):
Another concept associated with curriculum learning is autocurricula.
Multi-agent systems can provide naturally emerging curriculums, an autocurricula due to competition and cooperation. The solution of one task can often lead to new tasks, continuously generating new challenges which encourages further innovation.

We can consider the limitations of acquiring knowledge of a single agent through an example. Consider a solo player, playing a game on a 19 x 19 grid. The goal is to place black stones such that they surround as much territory as possible. After some trial and error, the optimal solution of placing all black stones on the perimeter will be reached. Beyond this, there is nothing left to learn.
However, say we introduce a new player to the game. They place a white stone after each black stone. The white stones and the territories they cover become barriers to the territories covered by the black stones, preventing additional expansion. The game of Go is born accordingly.

- Introduce the OpenAI environment as an example to explore curriculum learning - Luke
    - Explain the hide and seek concept
    - Seekers and hiders rewarded by finding/hiding from hiders/seekers respectively
    - Explain the action space
        - Move blocks, lock boxes so they can't be used by other team

        - Ramp surf to go over walls
    - Video of environment - https://cdn.discordapp.com/attachments/775381829054234639/803184120716328970/Single_Hider_and_Seeker_Example.mp4
    - Uploaded here: https://www.youtube.com/watch?v=WY4mTBrboPg

Script (Luke):
The hide and seek concept is exactly what the name suggests, a game between agents, consisting of agents who hide and agents who seek. The agents seeking are rewarded by finding the hiders, who are in turn rewarded by evading the seekers. As the hider is rewarded it will learn the sequence of actions taken, and the weight given to this sequence is based on the reward value. The agents exist within the OpenAI environment, which can be randomly generated, or designed appropriate to our needs, using the MuJoCo-Worldgen module. Curriculum learning can be applied by incrementally increasing the difficulty of the environment and seeker behaviour, in the hope that the hider can use previous knowledge to aid their evasion. The environment can be adjusted by adding/removing ramps, boxes and walls (moveable and immovable). The boxes and ramps can be moved by any agent, but if locked by the hider, then the seeker cannot move them. This opens the potential for hiders to learn strategies that prevent seekers from moving objects.

- Progress - Rayhan
    - We've created some environments
    - Background reading - very complex topic
    - Installing MuJoCo was a nightmare
    - Basically nothing

Script (Rayhan):
As for progress made regarding the project. We have done some complicated background reading on curriculum learning and the OpenAI environment. We've set up the Reinforcement Learning OpenAI environment to prepare to train the agents. This required installing MuJoCo which is a physics engine used via python to simulate 3d modelling and movement. It was quite tricky for all of us to get this working since it's quite particular about operating systems and python versions.
We've also created some environments of varying complexity, through change in both number of objects, and object types such as some with boxes and some including ramps.

- Plan to Completion - Jack
    - Further reading and understanding of curriculum learning
    - Understanding the code and being able to implement our own stuff
        - Creating more environments
        - Understanding the action space
        - Creating policies
        - Running simulations and collecting data

    - Alternatively, could do a more theoretical less programming approach, comparison of different RL approaches
        - Q-learning
        - Sarsa learning
        - Curriculum learning
        - Deep learning
        - Other
    - Everything basically

So far, using the provided policy, we have been able to run the hide and seek simulation at varying levels of environment complexity, which is one dimension of the variability we are aiming to provide to the learning agent. The other being that we plan to have our agent learn against opposing agents of varying difficulty, we will do this by seeding these opposition agents with policy at various phases of expertise within the environment. By measuring the time to convergence, the time it takes for the agent to develop its policy sufficiently for change in outcome, we plan to assess the value of the concept of curriculum learning by comparing it to the learning technique used by OpenAI.

The existing method took 380 million episodes to learn its terminal policy which is aptly named ‘box surfing’, this came some 270 million episodes after the emergence of the previous distinct strategy, ‘ramp defense’. The other three significant policy advancements were made in the remaining 110 million episodes, and the quickest of these transitions, from ‘ramps’ to ‘ramp defense’ only took 10 million episodes. This is the policy transition that we will likely target the most heavily in our simulations, due to being restricted in compute power to the limits of our local machines.

By creating our own policies targeting the policy transition, with various levels of seeded curricula, we aim to compare the speed of convergence. We hypothesise that with the optimal curricula it will take fewer episodes, than taken by the OpenAI algorithm, to reach the next significant phase of policy. In order to be sure that we are assessing the value of the curriculum learning techniques alone, we have restricted the number of hider and seeker agents to just one of each agent, and of these agents only the hider agent will be able to learn. The seeker agent will be seeded initially with the policy at the moment of transition, and since this agent will not be able to learn, we know that the improving hider policy will be tested against opposition of constant difficulty - which is an essential part of curriculum learning.