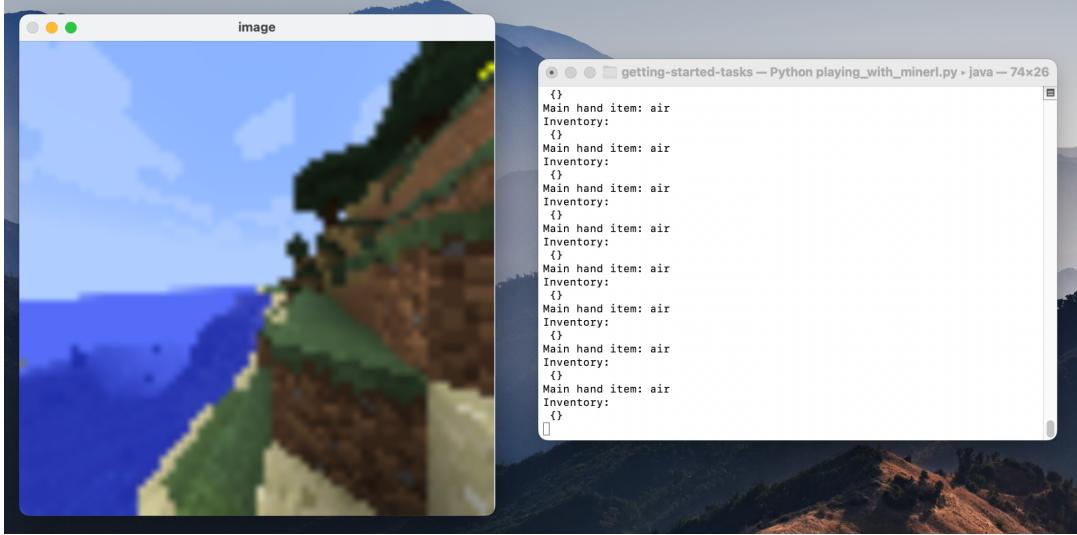


Requirement 4 Report - Getting Started With MineRL

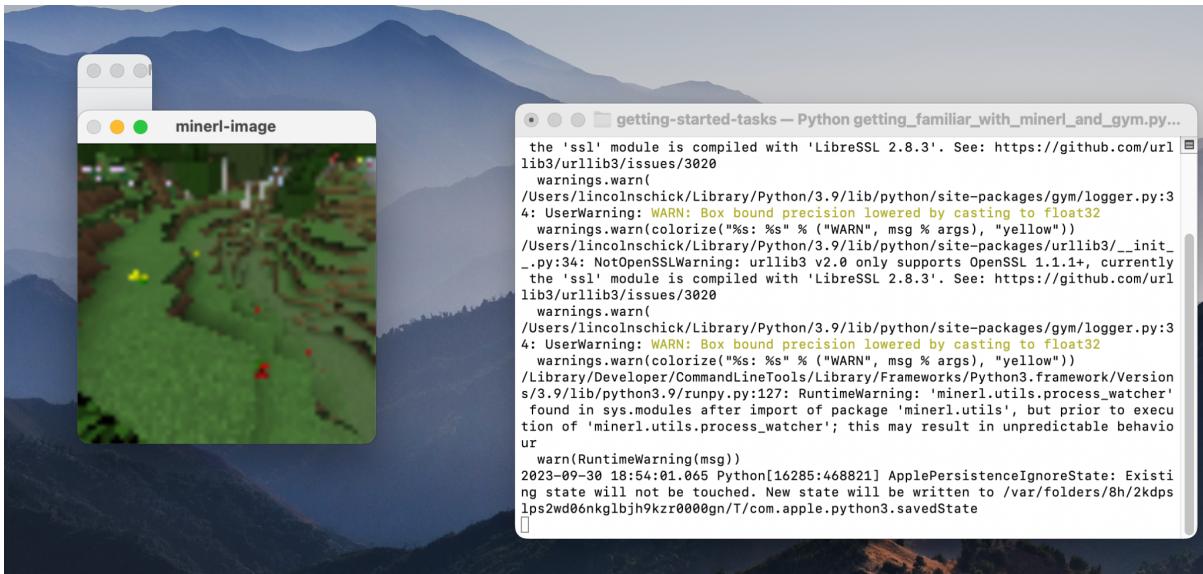
Lincoln

- Screenshot from running playing_with_minerl.py



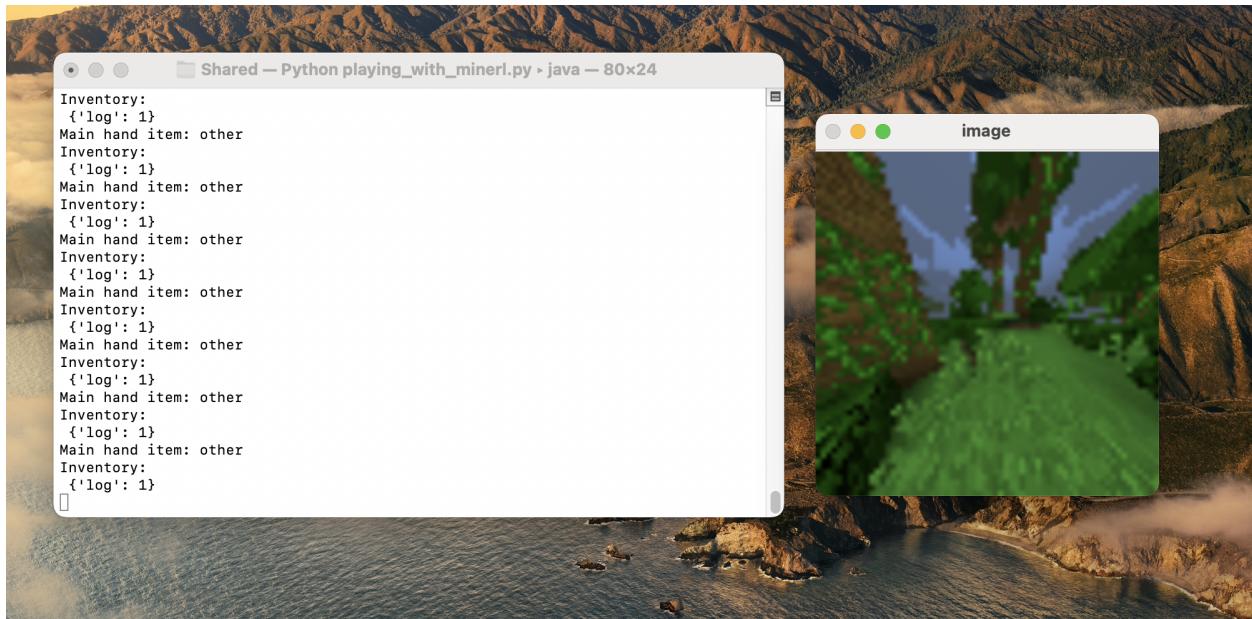
- CartPole-v0 environment
 - What is the goal?
 - The goal is to prevent the pole from falling down
 - What is the state information (what does the state from env.step(action) mean)? You may want to inspect environment.observation_space.
 - The state information tells you about the cart and pole at a particular time. Specifically, it is an array containing the cart position, the cart velocity, the pole angle, and the pole velocity.
 - What are actions (what does the action in env.step(action) do)? What are the different actions? You may want to inspect environment.action_space.
 - Actions are moves that can be taken in the game. env.step(action) will update the game with one's move. This is a discrete variable either a 0 or a 1 where a 0 moves the cart left and a 1 moves the cart right.
 - What is the reward?
 - The reward is just the number 1. This is because the goal is to stay alive and prevent the pole from falling down, so an AI would try to maximize the total reward by making moves that prevent the game from ending.
 - What is the starting state? Does it change between different episodes (games)?
 - The start state is simply an array containing initial values for the cart position, cart velocity, pole angle, and pole velocity, and it does change between different games.
 - Is the environment deterministic or stochastic? Environment is deterministic if you can perfectly predict what happens with every action in every state.
 - The environment is deterministic. Since the environment is purely based on laws of physics which are invariable, one can predict exactly what changes after each action.

- Screenshot from running `getting_familiar_with_minerl_and_gym.py` with the MineRLTreechop-v0 environment



Cody

- Screenshot from running playing_with_minerl.py



- CartPole-v0 environment
 - What is the goal?
 - The goal is to keep the stick balanced upon the cart, preventing the angle of the pole from reaching a value greater than 15 degrees.
 - What is the state information (what does the state from env.step(action) mean)? You may want to inspect environment.observation_space.
 - The state information is an array consisting of four values that denote the position and velocity of the cart and the pole at any given instance of the episode.
 - What are actions (what does the action in env.step(action) do)? What are the different actions? You may want to inspect environment.action_space.
 - Actions are the moves available to the AI to interact with the environment, modifying the state for the next iteration/step of the environment.
 - In this example, the AI has the ability to move the cart right or left depending on the value passed into the step() function. A value of zero denotes left and a value of one denotes right.
 - What is the reward?
 - The reward is the value 1.0. This is the reward for the AI performing an action that does not lead to the pole falling down. This leads to an AI agent trained via reinforcement learning in this environment to learn to maximize the value by keeping the pole balanced.
 - What is the starting state? Does it change between different episodes (games)?
 - The starting state is an array containing four float values denoting the position of the cart and the pole in the environment.
 - No, the starting state does not change between episodes.

- Is the environment deterministic or stochastic? Environment is deterministic if you can perfectly predict what happens with every action in every state.
 - This environment is deterministic. The only influences on the state are the actions of the machine learning agent and the values from the previous iteration of the state (i.e. velocity causing continued motion of either the cart or the pole). Thus, it is always possible to completely predict the next state from knowledge of the agent's actions and the previous state.
- Screenshot from running getting_familiar_with_minerl_and_gym.py with the MineRLTreechop-v0 environment

co minerl_intro.ipynb - Colaboratory + colab.research.google.com/drive/1pZ6HJECLfSSzgtL6itpVHTWqjkpPbZ9e#scrollTo=MHzrW1DxErxl

minerl_intro.ipynb File Edit View Insert Runtime Tools Help All changes saved Comment Share C

+ Code + Text

```
[23, 45, 12]],  
...,  
[[[ 0, 0, 0],  
[ 0, 0, 0],  
[ 34, 68, 19],  
...,  
[ 37, 73, 20],  
[ 33, 66, 18],  
[ 34, 68, 19]],  
[[ 0, 0, 0],  
[ 34, 68, 19],  
[ 34, 68, 19],  
...,  
[ 14, 27, 8],  
[ 33, 65, 18],  
[ 26, 51, 14]],  
[[ 21, 42, 12],  
[ 34, 67, 19],  
[ 36, 72, 20],  
...,  
[ 31, 62, 17],  
[ 11, 21, 6],  
[ 33, 65, 18]]], dtype=uint8})  
Reward: 0.0  
Completion: False
```

Requirement 10

Executing (4m 1s) <cell line: 77> > main()

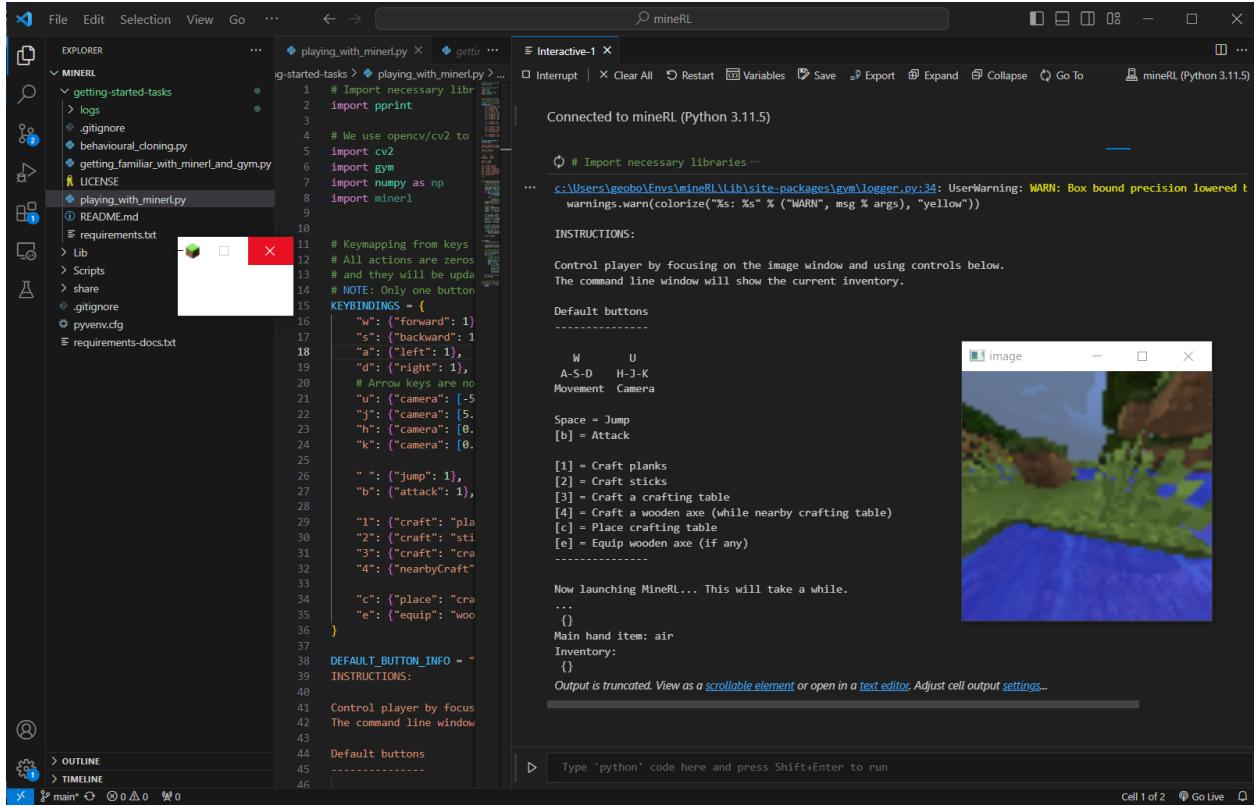
Media Player

00:00:04 0:04:20

1696706284.5795844

Jack

- Screenshot from running playing_with_minerl.py



The screenshot shows a Jupyter Notebook interface with the following details:

- EXPLORER:** Shows the project structure under "MINERL".
- CELL:** The code cell contains the following Python script:

```
1 # Import necessary libraries
2 import pprint
3
4 # We use opencv/cv2 to
5 import cv2
6 import gym
7 import numpy as np
8 import minerl
9
10 # Keymapping from keys
11 # All actions are zeros
12 # and they will be updated
13 # NOTE: Only one button
14 KEYBINDINGS = {
15     "w": ("forward", 1),
16     "s": ("backward", 1),
17     "a": ("left", 1),
18     "d": ("right", 1),
19     # Arrow keys are no
20     "u": ("camera", [-5, 0]),
21     "j": ("camera", [5, 0]),
22     "h": ("camera", [0, -5]),
23     "k": ("camera", [0, 5]),
24
25     "e": ("jump", 1),
26     "b": ("attack", 1),
27
28     "1": ("craft", "planks"),
29     "2": ("craft", "sticks"),
30     "3": ("craft", "cra"),
31     "4": ("nearbyCraft", "cra"),
32     "c": ("place", "cra"),
33     "e": ("equip", "woo")
34 }
35
36 DEFAULT_BUTTON_INFO = {
37     "Control player by focus": "The command line window"
38 }
39
40 INSTRUCTIONS:
41 Control player by focus
42 The command line window
43
44 Default buttons
45 -----
```
- Interactive-1:** A terminal window showing the output of the script. It includes instructions for controlling the player and a list of available commands. It ends with a message about launching MineRL.
- Image:** A 3D rendering of a Minecraft environment showing a player character in a grassy area near a body of water.

- CartPole-v0 environment

- What is the goal?
 - The goal of CartPole-v0 is to balance a wooden pole on a cart. The model has decisions it can make about how the cart is moved which may help balance or destabilize the pole depending on the model's previous actions.
- What is the state information (what does the state from env.step(action) mean)? You may want to inspect environment.observation_space.
 - The state information the model receives the certain metrics on the position and motion of the cart, including its current position along the surface, the angle of pole, and the velocity of both of those objects.
- What are actions (what does the action in env.step(action) do)? What are the different actions? You may want to inspect environment.action_space.
 - There are two actions the cart can take - to move left and to move right, which can be passed to env.step() to apply the action at each step. These actions may be passed randomly by sampling the action space or in a controlled manner by passing the action explicitly using some logic.
- What is the reward?
 - The reward is a constant value that is passed each time an action is made, rewarding the model for keeping the pole balanced.

- What is the starting state? Does it change between different episodes (games)?
 - The starting state is to have the pole balanced straight up in the air. This does not change between different episodes.
- Is the environment deterministic or stochastic? Environment is deterministic if you can perfectly predict what happens with every action in every state.
 - The environment is chaotic but deterministic. Since there is a consistent starting condition, a model performing the exact same inputs at every step should perform the same across different tests.
- Screenshot from running getting_familiar_with_minerl_and_gym.py with the MineRLTreechop-v0 environment

The screenshot shows a Jupyter Notebook interface with two code cells and a game window.

Code Cell 1:

```

13 # playing_with_minerl.py
14 # getting_familiar_with_minerl_and_gym.py M
15
16 # Get familiar with the environment CartPole-v0. Important things you need to figure out:
17 # What is the goal?
18 # What is the state information (what does it tell us about the world)
19 # What are actions (what does the action in the environment do)
20 # What is reward?
21 # What is the starting state? Does it change between episodes?
22 # Is environment deterministic or stochastic?
23
24 Now that you know how CartPole-v0 works, change the environment to MineRLTreechop-v0
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

Code Cell 2:

```

from time import sleep
c:\Users\geobo\Envs\minerl\lib\site-packages\gym\logger.py:34: UserWarning: WARN: Box
warnings.warn(colorize("%s" % ("WARN", msg % args), "yellow"))
Step: 0
Action: OrderedDict([('attack', array(1)), ('back', array(1)), ('camera', array([ -1
Observation: {'pov': array([[ [ 25, 33, 17],
[ 29, 38, 19],
...
[ 19, 38, 11],
[ 19, 38, 11],
[ 18, 35, 10]],

[[ 36, 47, 24],
[ 25, 33, 17],
[ 28, 37, 19],
...
[ 20, 39, 11],
[ 20, 39, 11],
[ 18, 36, 10]],

[[ 34, 45, 23],
[ 36, 47, 24],
[ 28, 37, 19],
...
[ 19, 38, 11],
[ 17, 34, 10],
[ 15, 31, 9]],

...
[ 24, 35, 16],
[ 24, 35, 16],
[ 0, 0, 0]]], dtype=uint8)}
Reward: 0.0
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings.

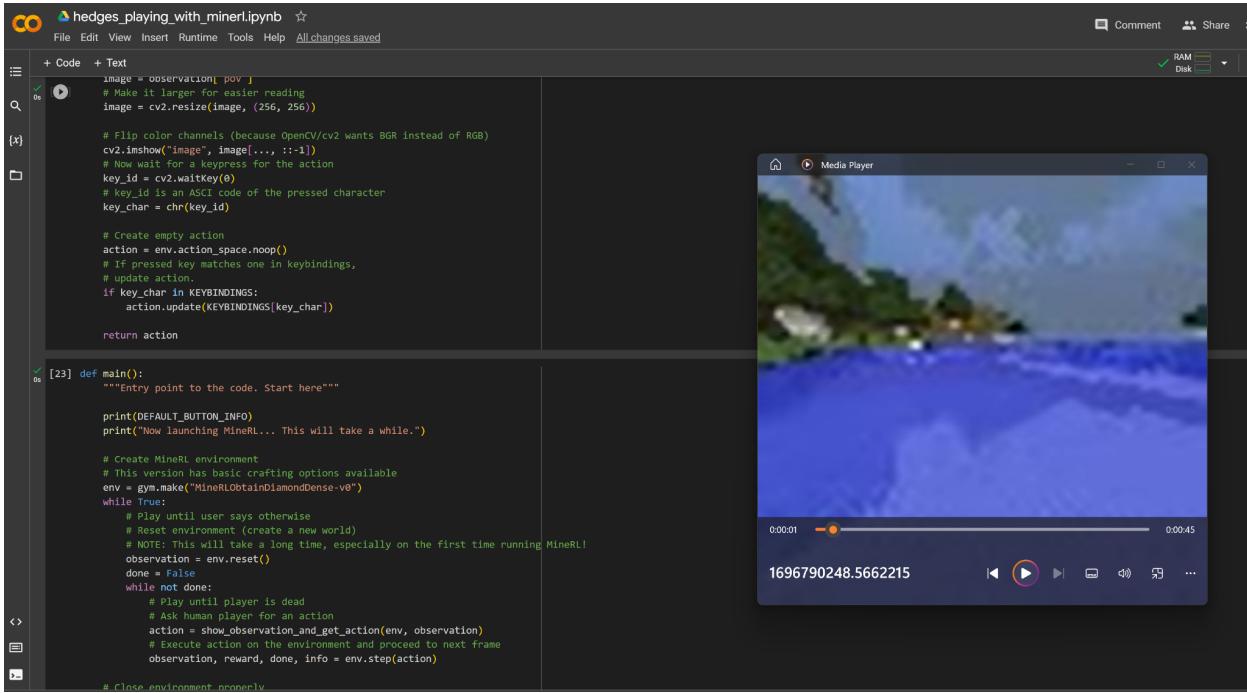
```

Game Window:

A small window titled "minerl-image" shows a Minecraft-like environment where a tree is being chopped down. The game window is overlaid on the Jupyter interface.

Faith

- Screenshot from running playing_with_minerl.py:



The screenshot shows a Jupyter Notebook interface with a code cell containing Python code for interacting with a MineRL environment. To the right of the notebook is a Media Player window displaying a video frame from the game, which appears to be a first-person view of a MineRL world.

```
hedges_playing_with_minerl.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
0s
# Make it larger for easier reading
image = cv2.resize(image, (256, 256))

# Flip color channels (because OpenCV/cv2 wants BGR instead of RGB)
cv2.imshow("image", image[...,:-1])
# Now wait for a keypress for the action
key_id = cv2.waitKey(0)
# key_id is an ASCII code of the pressed character
key_char = chr(key_id)

# Create empty action
action = env.action_space.noop()
# If pressed key matches one in keybindings,
# update action.
if key_char in KEYBINDINGS:
    action.update(KEYBINDINGS[key_char])

return action

[23] def main():
    """Entry point to the code. Start here"""

    print(DEFAULT_BUTTON_INFO)
    print("Now launching MineRL... This will take a while.")

    # Create MineRL environment
    # This version has basic crafting options available
    env = gym.make("MineRLObtainDiamondDense-v0")
    while True:
        # Play until user says otherwise
        # Reset environment (create a new world)
        # NOTE: This will take a long time, especially on the first time running MineRL!
        observation = env.reset()
        done = False
        while not done:
            # Play until player is dead
            # Ask human player for an action
            action = show_observation_and_get_action(env, observation)
            # Execute action on the environment and proceed to next frame
            observation, reward, done, info = env.step(action)

    # Close environment properly.
```

- Answers to the following questions about the CartPole-v0 environment
 - What is the goal?
 - To maximize the amount of time that the simulated pole is balanced on the cart, via motions of the cart that the agent controls.
 - What is the state information (what does the state from env.step(action) mean)? You may want to inspect environment.observation_space.
 - It is a matrix containing the cart linear position, cart linear velocity, pole angular position, and pole angular velocity
 - What are actions (what does the action in env.step(action) do)? What are the different actions? You may want to inspect environment.action_space.
 - Actions are the manner in which an agent is able to interact with an environment; they are the rules that the entity must abide by while striving for a maximal reward. The action space in cartpole is limited, since it is a simple testing environment: move the cart a linear increment to the left or to the right.
 - What is the reward?
 - In general, rewards are points that a model aims to maximize. In this game, the reward is given based on the amount of time spent with the pole in the optimal range; the game ends once it falls out of this range (or the cart moves to a lateral out-of-range position), preventing a higher score.
 - What is the starting state? Does it change between different episodes (games)?
 - For cartpole, the starting state is always the same, since consistency provides simplicity, the goal of this test.

- Is the environment deterministic or stochastic? Environment is deterministic if you can perfectly predict what happens with every action in every state.
 - It is deterministic, since the laws of physics are simulated within this environment; given an input cart velocity, the radial pole velocity can be precisely calculated.
- Screenshot from running `getting_familiar_with_minerl_and_gym.py` with the MineRLTreechop-v0 environment:

The screenshot shows a Jupyter Notebook interface with a code cell and a media player window.

Code Cell Content:

```
# Start by using CartPole-v0 for fast debugging and tuning!
# Once code works, change to "MineRLTreechop-v0"
environment = gym.make("MineRLTreechop-v0")
# Add a code that renders the events
environment = Recorder(environment, './video', fps=30)

# Reset the environment (NOTE: With MineRL, this will take time, but with CartPole-v0 it's fast)
observation = environment.reset()

# raise NotImplementedError("Implement 'step-loop' here to play one episode, and then close the environment")

# TODO step-loop
# A while-loop until game returns done == True:
#   - Pick a random action: 'environment.action_space' tells what kind actions
#     - You can use 'environment.action_space.sample()' to get a random action
#   - Step game with environment.step(action) . See documentation: http://gym-
#   - Print what kind of values you got from the game (observation, Reward, done)
#   - Limit interaction to 1000 steps when using a MineRL environment to speed up training
done = False
step_counter = 0
while not done and step_counter < 1000:
    #random action
    action = environment.action_space.sample()

    #step
    observation, reward, done, info = environment.step(action)
    print(f'Observation: {observation}, Reward: {reward}, Done: {done}')
    step_counter += 1

# Save the latest video, show it and then close environment
environment.release()
# environment.play()
environment.close()
```

Media Player Window:

A video player window titled "Media Player" displays a scene from the MineRLTreechop-v0 environment. The video shows a first-person view of a character in a forest-like setting, with a blue sky and green trees. The video player has a progress bar showing 0:00:31 to 0:00:02. Below the video, the file name "1696804639.5621119" is displayed along with playback controls.