Requirement 35 Report - Integrate UI with current models for wood and stone collection

Model Integration

Integrating the GUI with our currently trained models was successful, and their corresponding placeholders have been replaced with the appropriate pytorch model files under "\src\GUI\models." A demo video of models successfully swapping is available under the same directory as this report (docs\reports\requirement-35), as *model_swap_demo.mp4*.

Multiprocessing / Multithreading

Initial efforts by Lincoln targeted implementation of multithreading to allow the GUI and minerl environment to operate simultaneously. However, due to the overhead of context switching, this drastically reduced the performance of both components. Thus, a multiprocessing approach was implemented, which allowed simultaneous operation of both components without a loss in performance.

Strategy / Implementation

A new *controller.py* file contains an AgentController object that handles the status and properties of the currently running agent by swapping models, loading the minerl environment, and running the actions taken by the agent. This AgentController acts in a process separate from the GUI, and the two components utilize queues for interprocess communications in a producer / consumer architecture. The role of each component changes based on the action occurring, such as the GUI producing signals for the AgentController to swap the model, versus the AgentController producing data for the GUI to display.

Some GUI components have been altered to facilitate clearer purpose, such as starting the agent, and reloading environments to reset the agent. Objective changes are now done instantly when the corresponding radio button is selected. An updated GUI image can be found under "src\GUI\GUI Examples" as *ID 35 UI.png*. The GUI has also been separated into *main.py* and *gui_design.py* files. The *gui_design.py* file contains code generated from the *ML4MC.ui* file, and hosts the UI_MainWindow class. The *main.py* file contains code that applies the GUI's functionality by importing the UI_MainWindow class from *gui_design.py*. This allows frequent regeneration of the GUI design, without having to rewrite the functionality each time the design is altered. *main.py* is also the entry point for the program, and initializes the AgentControllers, queues, and backend process for the minerl environment.

Future Features

- Tracking agent statistics is still viable. Minerl environments offer observables that provide the desired information, which can then be returned to the GUI via the AgentController.
- Pausing and resuming the agent is still viable, as the GUI can signal the AgentController to place a pause and wait on its main loop that selects and renders the agent's action.
- Recording video of the agent is still viable, as the gym dependency offers a wrapper for recording videos. Recording an interactor will require a screen-recording API.

- Toggling script has been deemed unnecessary. Custom scripts for certain actions will instead be activated directly when desired. Toggling actions on and off interferes too much with the models.