Requirement 11 Report - Overwolf or Alternative GUI

MineRL loads a "gym" environment of Minecraft via dependency Project Malmö. Overwolf initializes by recognizing a game when it is launched through a compatible game launcher, such as the Minecraft Launcher. Therefore, is Overwolf capable of detecting the environments used by MineRL and launching?

Overwolf is not capable of detecting the environments used by MineRL, it is only capable of detecting instances of Minecraft launched from the official Minecraft Launcher. This was tested and verified by running MineRL instances in various ways and seeing if Overwolf would recognize Minecraft as a launched game. It could not, thus an alternative GUI library will be needed, and our software architecture will need to be updated. Furthermore, our selected version of MineRL, v0.4.4, utilizes the 1.11.2 version of Minecraft, which is not supported by Overwolf's games.events API. While Overwolf's other API's could be usable, an overlaying UI would not be feasible through Overwolf if the library cannot detect the window of the running game instance.

If necessary, what other libraries are available for creating a user interface that can interact and communicate with the MineRL model?

There are a large number of GUI programming libraries available for Python [1]. Thus, our options will be kept to popular libraries that allow use in open source projects, and that support the same platforms that MineRL can be expected to run on (Windows, Linux, MacOS) [2]:

- PySimpleGUI [3]
    + Wraps tkinter, Qt, WxPython and Remi GUI libraries to interact with their widgets in a user-friendly manner.
    + Code written on one library can work on another by changing imports.
    + Extensive documentation and examples, courses available.
    - Each library is a distinct "port" that must be installed separately.
    - Not all "ports" are fully complete.
    - Somewhat limiting.
- Toga [4]
    + Uses OS native widgets that are better optimized than generics..
    + Built to be Python native to allow usage of language level features.
    + Tutorials and documentation.
    - In early development.
- tkinter [5]
    + Standard Python Interface to the Tcl/Tk GUI toolkit.
    + Supports standard layouts and widgets, and complex GUI widgets.
    + Cross-platform but widgets can look outdated on certain platforms.
    - No built-in support for GUIS driven from data sources, databases, or for displaying or manipulating multimedia or hardware.
- PyQt [6]
    + Drag and drop to implement visual elements.

    + Python code handles logic, QML defines the structure and behavior of the UI.
    + Can embed Javascript code to handle events and animations.
    - Only commercial / closed-source projects need to purchase a license.
- WxPython [7]
    + Uses native widgets on most platforms to improve UI appearance.
    + Wraps the wxWidgets cross-platform library.
    + Under active development.
    - Known to have platform-specific quirks.
    - Doesn't provide as much abstraction as other libraries.
    - Being reimplemented as "WxPython Phoenix".

References

[1] "GUI Programming in Python", *Python Wiki*, https://wiki.python.org/moin/GuiProgramming (accessed Oct. 7, 2023).

[2] Chaetognathan, Punition, "Which Python GUI library should you use?", *PythonGUIS*, https://www.pythonguis.com/faq/which-python-gui-library/ (accessed Oct. 7, 2023).

[3] "PySimpleGUI", https://www.pysimplegui.org/en/latest/ (accessed Oct. 7, 2023).

[4] "Toga", *BeeWare,* https://beeware.org/project/projects/libraries/toga/ (accessed Oct. 7, 2023).

[5] "tkinter", *Python Docs*, https://docs.python.org/3/library/tkinter.html (accessed Oct. 7, 2023).

[6] "PyQt", *Riverbank Computing*, *https://www.riverbankcomputing.com/software/pyqt/ (accessed Oct. 7, 2023).*

[7] "WxPython", *https://wxpython.org/index.html (accessed Oct. 7, 2023).*