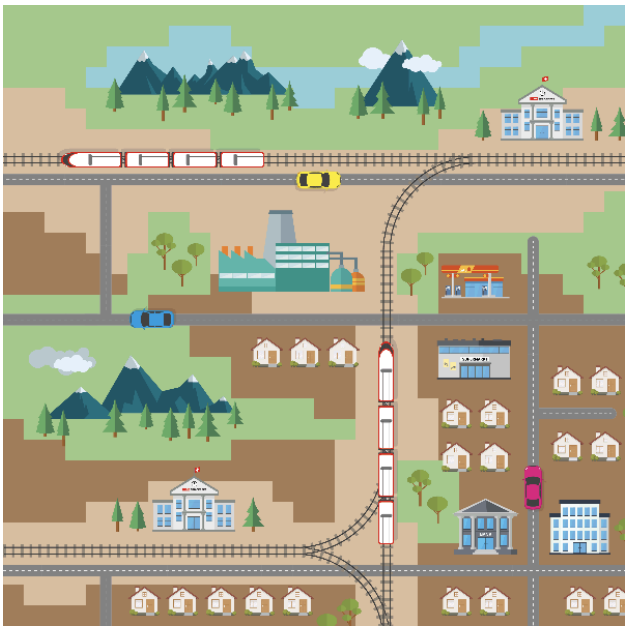
Flatland-RL Visualization Specification

Version 0.1 21 March 2019



Scope

This doc specifies the software to meet the requirements in the Visualization requirements doc.

References

Visualization Requirements
https://docs.google.com/document/d/1Y4Mw0Q6r8PEOvuOZMbxQX-pV2QKDuwbZJBvn18mo9UU/edit#

Core Spec
https://docs.google.com/document/d/1RN162b8wSfYTBblrdE6-Wi_zSgQTvVm6ZYghWWKn5t8/edit

Interfaces

Interface with Environment Component

- Environment produces the Env Snapshot data structure (TBD)
- Renderer reads the Env Snapshot
- Connection between Env and Renderer, either:
    - Environment “invokes” the renderer in-process
    - Renderer “connects” to the environment
        - Eg Env acts as a server, Renderer as a client
- Either
    - The Env sends a Snapshot to the renderer and waits for rendering
- Or:
    - The Env puts snapshots into a rendering queue
    - The renderer blocks / waits on the queue, waiting for a new snapshot to arrive
        - If several snapshots are waiting, delete and skip them and just render the most recent
        - Delete the snapshot after rendering
- Optionally
    - Render every frame / time step
    - Or, render frames without blocking environment
        - Render frames in separate process / thread

Environment Snapshot

Data Structure

A definitions of the data structure is to be defined in Core requirements or Interfaces doc.

Example only

Top-level dictionary

- World nd-array
    - Each element represents available transitions in a cell
- List of agents
    - Agent location, orientation, movement (forward / stop / turn?)
    - Observation
        - Rectangular observation
            - Maybe just dimensions - width + height (ie no need for contents)
            - Can be highlighted in display as per minigrid
        - Tree-based observation
            - TBD

Existing Tools / Libraries

1. Pygame
    a. Very easy to use. Like dead simple to add sprites etc. (https://studywolf.wordpress.com/2015/03/06/arm-visualization-with-pygame/)
    b. No inbuilt support for threads/processes. Does get faster if using pypy/pysco.
2. PyQt
    a. Somewhat simple, a little more verbose to use the different modules.
    b. Multi-threaded via QThread! Yay! (Doesn't block main thread that does the real work), (https://nikolak.com/pyqt-threading-tutorial/)

How to structure the code

1. Define draw functions/classes for each primitive
    a. Primitives: Agents (Trains), Railroad, Grass, Houses etc.
2. Background. Initialize the background before starting the episode.
    a. Static objects in the scenes, directly draw those primitives once and cache.

Proposed Interfaces

To-be-filled

Technical Graphics Considerations

Overlay dynamic primitives over the background at each time step.

No point trying to figure out changes. Need to explicitly draw every primitive anyways (that’s how these renders work).