drawDriverless.py is a function class (to be used on Map class objects) for visualizing an ‘instance’.

For handling window creation and interaction (in both Windows and Linux), the pygame library is used. drawDriverless.py contains the function class pygameDrawer and several pygame handling functions. These pygame functions serve to create-, destroy- and handle user input (UI) for an ‘instance’. The pygameDrawer class has functions for rendering elements on screen like drawCar(), drawCones() and drawPathLines(). It also allows for drawing text, like window framerate and car statistics, as well as visual debugging, like drawing points, lines and arcs. The Map class exclusively deals in values with units (meters in this case, but that is not hardcoded), so to interface with it, the pygameDrawer class has the functions realToPixelPos() and pixelsToRealPos(), which convert between pixel- and real (meters) coordinates. These functions also allow advanced ‘camera’ (subsection of the map the window shows) controls, like zooming, dragging (moving the ‘camera’), rotating and centering around the car. In the ‘carCam’ mode, the car is at the center of the window and all objects are drawn with respect to it.

In addition to local visualization, drawDrivereless is also used for remote Map visualization (remoteMapViewer.py). In those cases, a pygameDrawer boolean attribute called ‘isRemote’ is set to True, and all UI instructions are transmitted (using mapSockRecv.py) to the ‘instance’ that is being shown. The most significant function for this transmission is remoteInstructionSend().

The pygameDrawer class is designed to allow for multiple ‘instances’ to exist in one window. There are no examples of this usage (anymore) at time of writing, because there has been no explicit need for it yet, but this multi-instance-window could potentially be useful for showing largescale simulations with slight variances, or an AI model training (by evolutionary iteration or otherwise). This multi-instance possibility is why functions why handleWindowEvent() are structured the way they are, and why currentPygamesimInput() exists.

For instructions on how to use the UI (the effects of specific buttons, what certain visual elements mean), please use the appropriate README.md file on the HARD driverless Github.