LIGHTBOT

An Adaptive Traffic Light Control Solution

CODING STANDARDS DOCUMENT

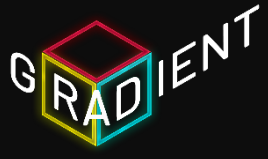

Table of Contents

INTRODUCTION

The LightBot system is designed to improve the flow of traffic at an intersection. This is done by using an Reinforcement Learning algorithm to decide how to change traffic lights in various scenarios. The system is divided into multiple components, each designed to perform specific tasks in the system. For more details regarding the various components, please refer to the Software Requirements Specification Document.

SYSTEM COMPONENTS

The components of the LightBot system which require attention are:

- Web Application
- Web Server
- Traffic Simulation

Various coding standards are applied to all of the components, as described in the following sections

For any assistance, please contact us at teamgradient301@gmail.com.

WEB APPLICATION

The web application was implemented with the following programming languages and frameworks:

- JavaScript
    - React

The following coding standards and conventions were adhered to during development:

- Variable names are defined using camel case
- Spaces around operators and after commas are used
- Code indentation is two whitespace characters
- No semicolons were used
- Single quotation marks are used
- Arrow functions are preferably used where possible

Prettier is used in VS Code editor to maintain the coding standard for JavaScript

WEB SERVER

The web application was implemented with the following programming languages and frameworks:

- JavaScript
    - NodeJS
    - SocketIO

The following coding standards and conventions were adhered to during development:

- Variable names are defined using camel case
- Spaces around operators and commas are used
- Code indentation is two whitespace characters
- No semicolons were used
- Single quotation marks are used
- Arrow functions are preferably used where possible

Prettier is used in VS Code editor to maintain the coding standard for JavaScript

Comments are used to accurately describe the code for the generation of the Doxygen documentation.

TRAFFIC SIMULATION

The Traffic Simulation makes use of the SUMO (Simulation of Urban MObility) application. The SUMO application uses XML to create traffic intersections.

The Traffic Simulation was implemented with the following programming languages and packages:

- Python
    - Tensorflow
    - SUMO Library
    - TraCI (Traffic Control Interface)

The following coding standards and conventions were adhered to during development:

- PEP 8 -- Style Guide for Python Code is followed as close as possible
- Four space characters have been used for the indentation for the code for convenience and consistency
- Variable and function names appropriately describe what they represent and do respectively, and use underscores to separate words.

Comments, such as below, should be used to accurately describe the code for the generation of the Doxygen documentation.

Below is an example of the coding standards followed using Python:

```
## @package pyexample
#  Documentation for this module.
#
#  More details.

## Documentation for a function.
#
#  More details.
def func():
    pass
## Documentation for a class.
#
#  More details.
class PyClass:

    ## The constructor.
    def __init__(self):
        self._memVar = 0;

    ## Documentation for a method.
    #  @param self The object pointer.
    def PyMethod(self):
        pass

    ## A class variable.
    classVar = 0;
```