Path planner for the Phoenix environment
Package decription

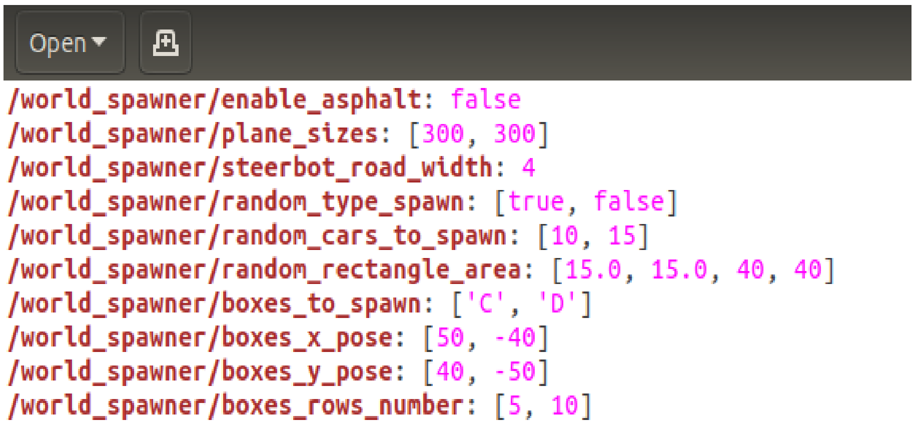
Vadym Ostapovych

May 6, 2021

1.Parameters

1.1. Intro

We'll go through the parameters with the help of *world.yaml* file in the *config* folder of the package All the



```
/world_spawner/enable_asphalt: false
/world_spawner/plane_sizes: [300, 300]
/world_spawner/steerbot_road_width: 4
/world_spawner/random_type_spawn: [true, false]
/world_spawner/random_cars_to_spawn: [10, 15]
/world_spawner/random_rectangle_area: [15.0, 15.0, 40, 40]
/world_spawner/boxes_to_spawn: ['C', 'D']
/world_spawner/boxes_x_pose: [50, -40]
/world_spawner/boxes_y_pose: [40, -50]
/world_spawner/boxes_rows_number: [5, 10]
```

Figure 1: Example of .yaml file for the package

parameters have the prefix /**world_spawner** and will be shown in the *rosparam list* only after running the world spawner client or loading from .yaml file.

1.2. Description

Default value are set if the client world spawner runs without before predefined parameters with *rosparam set* or *rosparam load.*

1.3. Meaning

1. /**world_spawner**/**enable_asphalt** - true if to spawn the plane with the aspahl texture, false - to spawn without the texture. Geometry properties of both planes are the same.

2. /**world_spawner**/**plane_sizes** - [length, width] of the plane.

3. /**world_spawner**/**steerbot_road_width** - distance between the parking boxes, the same one horizontally or vertically(for now).

4. /**world_spawner**/**random_type_spawn** - defines what type of car random spawning to do.

Parameter name	Data type	Default value
/world_spawner/enable_asphalt	boolean	true
/world_spawner/plane_sizes	int(>0) list of 2 elements	[100, 100]
/world_spawner/steerbot_road_width	int(>0)	4
/world_spawner/random_type_spawn	boolean list of 2 elements	[true, false]
/world_spawner/random_cars_to_spawn	int(>0) list of 2 elements	[15, 20]
/world_spawner/random_rectangle_area	float list of 4 elements	[-30, -40. 15, 20]
/world_spawner/boxes_to_spawn	string list of N elements	['C']
/world_spawner/boxes_x_pose	float list of N elements	[50]
/world_spawner/boxes_y_pose	float list of N elements	[40]
/world_spawner/boxes_rows_number	float list of N elements	[10]

Table 1: **world spawner client** parameters

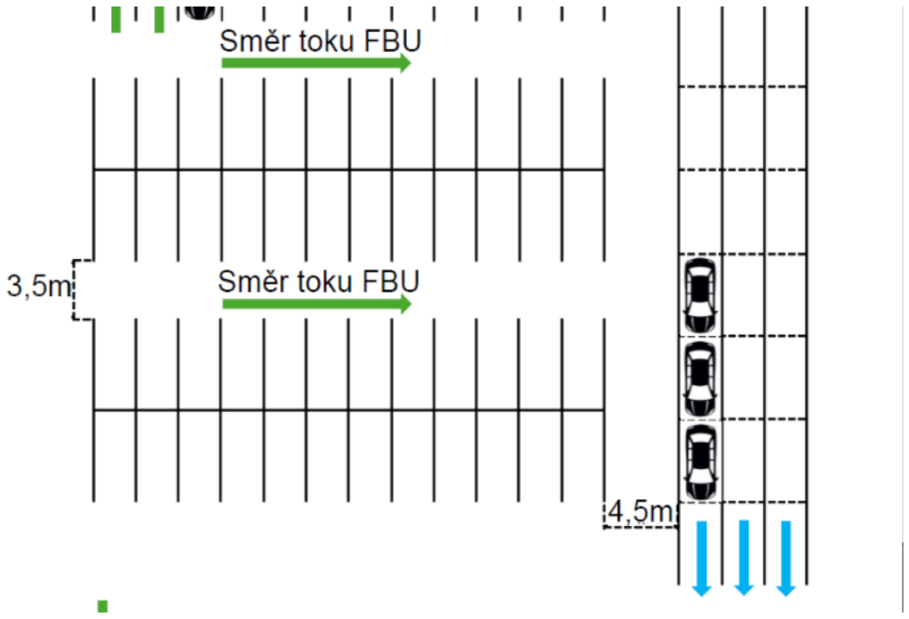


Figure 2: Distance between the parking boxes(3.5 m, 4.5 m)

1.1. By slots. To spawn cars on the spawned parking slots.

1.2. By area. To spawn car on the rectangle area.

5. **/world_spawner/random_cars_to_spawn** - number of cars to spawn in a car random spawn mode = by slot, by area.

6. **/world_spawner/random_rectangle_area** - defines the rectangle area for spawning cars by area = $[x_{coord}, y_{coord}$, length, width], **length and width** must be greater than zero.

7. **/world_spawner/boxes_to_spawn** - defines what type of parking boxes to spawn.

8. **/world_spawner/boxes_x_pose** - array of x_{coord} positions of the parking boxes. Each element defines the x coordinate of i-th box.

9. **/world_spawner/boxes_y_pose** - array of y_{coord} positions of the parking boxes. Each element defines the y coordinate of i-th box.

10. **/world_spawner/boxes_rows_number** - each element defines the number of rows in a i-th parking box. All values are greater than zero.

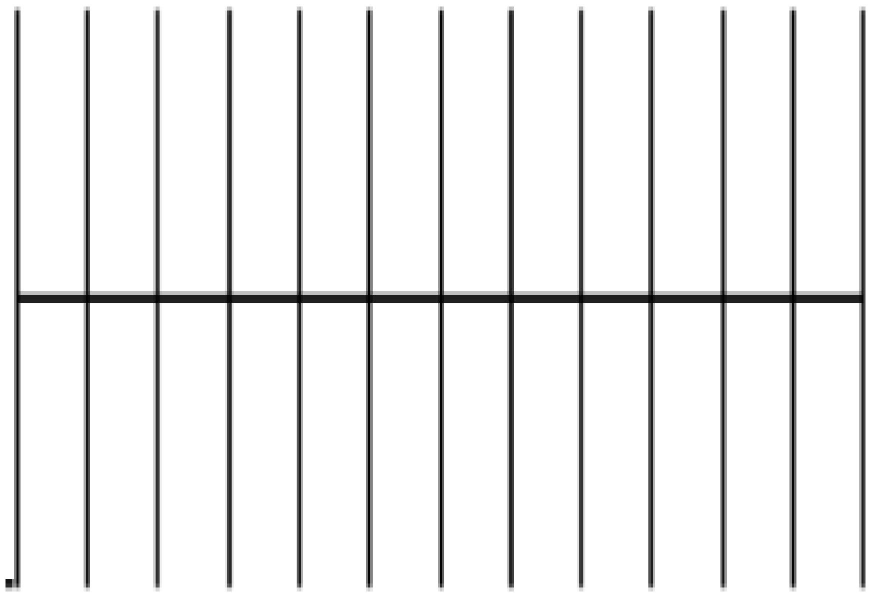
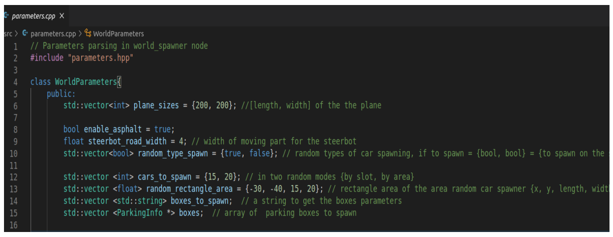


Figure 3: Example of the parking box

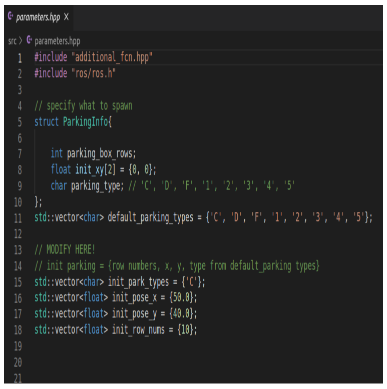
1.4. Parking boxes types

You can find the list of basic boxes types in parameters.hpp as $std :: vector < char > default\_parking\_types$. And also here you can change the default values of the parameters by modifying the vectors $init\_park\_types$, $init\_pose\_x$, $init\_pose\_y$, $init\_row\_nums$, but remember that the sizes of these four arrays must be the same and also follow the parameters conventions from 1. Other parameters are defined as the attributes of the $classWorldParameters$ in parameters.cpp, except $boxex\_to\_spawn$ and $boxes$ (**Don't modify these two ones!**)



```
// Parameters parsing in world_spawner node
#include "parameters.hpp"

class WorldParameters{
    public:
        std::vector<int> plane_sizes = {200, 200}; //[length, width] of the the plane

        bool enable_asphalt = true;
        float steerbot_road_width = 4; // width of moving part for the steerbot
        std::vector<bool> random_type_spawn = {true, false}; // random types of car spawning, if to spawn = {bool, bool} = {to spawn on the

        std::vector <int> cars_to_spawn = {15, 20}; // in two random modes {by slot, by area}
        std::vector <float> random_rectangle_area = {-30, -40, 15, 20}; // rectangle area of the area random car spawner {x, y, length, widt
        std::vector <std::string> boxes_to_spawn;  // a string to get the boxes parameters
        std::vector <ParkingInfo *> boxes;  // array of  parking boxes to spawn

```

(a) parameters.cpp



```
#include "additional_fcn.hpp"
#include "ros/ros.h"

// specify what to spawn
struct ParkingInfo{

    int parking_box_rows;
    float init_xy[2] = {0, 0};
    char parking_type; // 'C', 'D', 'F', '1', '2', '3', '4', '5'
};
std::vector<char> default_parking_types = {'C', 'D', 'F', '1', '2', '3', '4', '5'};

// MODIFY HERE!
// init parking = {row numbers, x, y, type from default_parking types}
std::vector<char> init_park_types = {'C'};
std::vector<float> init_pose_x = {50.0};
std::vector<float> init_pose_y = {40.0};
std::vector<float> init_row_nums = {10};



```

(b) parameters.hpp

Figure 4: Default parameters in C++ code

Boxing types

1. 'C' - consists of other parking boxes, middle part of the parking, spawns separately from D.

2. ’D’ - left part of the parking, consists of other parking slots, spawns separately from C.
3. ’F’ - full parking, ’C’ and ’D’ block

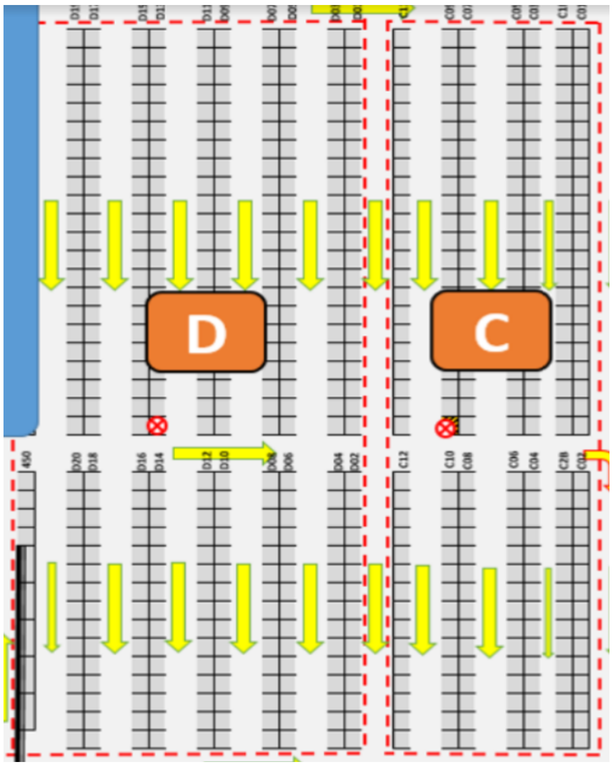


Figure 5: ’F’ type

4. ’1’ - Left closed with one column
5. ’2’ - Right closed with one column, mapping symmetrical about the vertical axis of the ’1’.

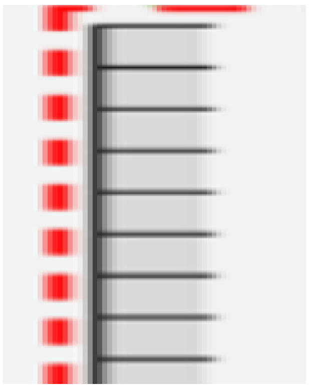


Figure 6: ’1’ type

6. ’3’ - Left closed with two columns, mapping symmetrical about the vertical axis of the ’4’.
7. ’4’ - Right closed with two columns.

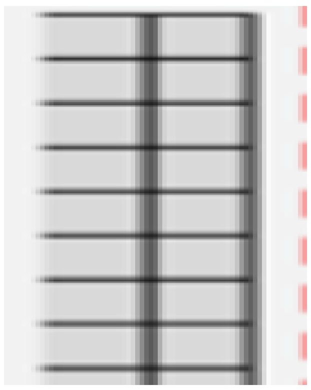


Figure 7: ’4’ type

8. ’5’ - Opened with two columns.

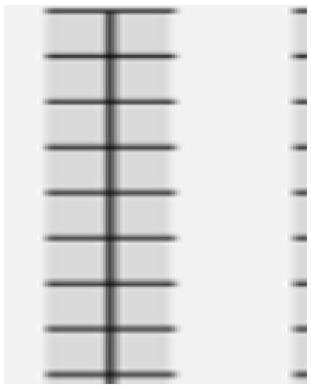


Figure 8: ’5’ type

1.5. Parsing of the incorrect parameters

All parameters before **/world_spawner/boxes_to_spawn** are not monitored for the correct input, so it’s possible to get an error caused by wrong parameters format. The other ones are parsed so that if one of the elements in the list is incorrect or sizes of arrays are not equal, all the parameters are set to default ones. **Be aware of incorrect input parameters!**