Protocol for the third ARP assignment

Version 0.1

Contents

1 Protocol specifications

1.1 General message characteristic's

- All messages exchanged are null terminated strings.
- Every message has a maximum length of 1024 characters. The maximum length of the messages is specified by a constant:

```
MAX_MSG_LEN = 1024;
```

- Every number exchanged through sockets must be represented like a float or double number with a maximum of 3 decimal places in order to not inflate too much the string dimension. To achieve this a suggestion would be to use sprintf:

```
sprintf(dest_buffer, "%f.3", double_or_float_number);
```

- Every message must have an echo from the receiving end

1.2 Code conventions

- Every 2d vector exchanged through sockets, being the dimension of the window or the position of a target or obstacle must be expressed in a row-column or x-y manner. This means that the axes are located on the top left of the map window and with this shape:

```
            y
+----------->
|
|
|
|
V x
```

- A fixed maximum dimension for the array containing the targets has been established:

  ```
  MAX_TARG_ARR_SIZE = 20;
  ```

  It is possible to have the target process send a number which is less or equal than this value, but not higher.

- A fixed maximum dimension for the array containing the obstacles has been established:

  ```
  MAX_OBST_ARR_SIZE = 20;
  ```

  It is possible to have the obstacles process send a number which is less or equal than this value, but not higher.

- The targets process must generate a first list of targets at the moment of connection with the server and then only when the server sends the “GE” message. The “GE” message is documented later in the document.

- The obstacles process must send a new set of obstacles every N seconds where N can be decided freely. The only suggestion is that if obstacles are completely regenerated then N should be big enough.

1.3 Messages formatting

Target and Obstacles processes will always put a character that identifies themselves at the beginning of the message that they need to send.

- Target process is identified by a ’T’
- Obstacles process is identified by a ’O’

1.3.1 Identification message

The identification message is just:

- “TI” → in case of the target process
- “OI” → in case of the obstacles process

These are the first messages that need to be sent when a connection is established.

1.3.2 Window size message

Once the server has received from the target or obstacles processes the identification message it will need to respond to both with the window dimension for which they will need to generate the targets or obstacles. This message is the HEIGHT-WIDTH of the window, comma separated. Please note the order. For example:

```
"500.234,444.332"
```

The targets and obstacles positions must be generated as random values between 0 and the received values.

1.3.3 Target or Obstacles list message

The first character of the string is as previously specified the identifier of the process who is sending data to the server. Let's suppose that we are considering now the Obstacles process, but the format is exactly the same also for the targets process apart from the first character.

After the identifier the number of points sent by the process must be specified inside square brackets. So for example if the obstacle process is sending 13 points then the first part of the message would be:

```
O[13]
```

From this point on obstacles are sent. Points are sent by always adhering to the row-cols convention. X and y components are separated by a comma, and two different obstacles are separated by a pipe. An example of a message could be:

```
O[2]123.456,234.567|23.234,0.000
```

1.3.4 Game end message

This message must be sent by the server when the drone has reached all the targets and triggers the generation of new targets from the targets process. This message has nothing to do with the obstacles process. This message is as follows:

```
"GE"
```

1.3.5 Stop message

To have a clean termination for all the processes a notification message for which every process must die has been chosen. This message is:

```
"STOP"
```

Upon receiving this message every process must terminate after having echoed the server.

2 Example of communication

The following table represent an example of the communication between the server and the target process.

TRANSMITTER	MESSAGE	TIME INSTANT	MESSAGE TYPE
Target	TI	0	normal
Server	TI	0	echo
Server	500.234,444.332	1	normal
Target	500.234,444.332	1	echo
Target	T[2]123.456,234.567|23.234,0.000	2	normall
Server	T[2]123.456,234.567|23.234,0.000	2	echo
Server	GE	15	normal
Target	GE	15	echo
Target	T[2]444.333,222.222|111.111,333.233	15	normal
Server	T[2]444.333,222.222|111.111,333.233	15	echo
Server	STOP	27	normal
Target	STOP	27	echo

- Transmitter is the process who is transmitting the message
- All messages are strings
- Time instant is represented here in seconds. It's not accurate. It's just a way to give an idea of the timing of messages for this particular example.
- Message type just specifies if it's an echo or not

Just for completion also the example for the obstacle process is shown but it's almost the same as the previous one without the "GE" message because it's not needed for obstacles generation. Please note that in this case the time interval between two obstacle generations is 10 seconds.

TRANSMITTER	MESSAGE	TIME INSTANT	MESSAGE TYPE
Obstacles	OI	0	normal
Server	OI	0	echo
Server	500.234,444.332	1	normal
Obstacles	500.234,444.332	1	echo
Obstacles	O[2]123.456,234.567|23.234,0.000	2	normall
Server	O[2]123.456,234.567|23.234,0.000	2	echo
Obstacles	O[1]333.333,222.222	12	normal
Server	O[1]333.333,222.222	12	echo
Obstacles	O[2]444.333,222.222|111.111,333.233	22	normal
Server	O[2]444.333,222.222|111.111,333.233	22	echo
Server	STOP	27	normal
Obstacles	STOP	27	echo