# Project Progress

Table of contents

[Project Progress 1](#_Toc47827041)

[Building the simulation 1](#_Toc47827042)

[Architecture 1](#_Toc47827043)

[Sanity test 2](#_Toc47827044)

Figures list

[Figure 1 – simulation architecture 2](#_Toc47826834)

[Figure 2 – environment set-up for zombie master sanity check with optional actions 3](#_Toc47826835)

[Figure 3 – actions distribution along different ranges of episodes 4](#_Toc47826836)

[Figure 4 – Total zombies survived vs. episodes (blue) with its moving average (orange) 4](#_Toc47826837)

## Building the simulation

### Architecture

Building a simulation for Reinforcement learning purposes is mostly a manner of creating an environment and throw there some entities that follow certain rules.

In our case the entities thrown into the environment are zombies and light.  
The managers of the above entities are the zombie and light masters. The zombies master has the ability to place a zombie at some starting position as he wishes, and the light master has the ability to place the light somewhere on top the board.

Once the zombie master ordered to place a zombie, the only rule for the zombie is move straight to the other side of the board.

Hence, the two parts of the simulation are the: environment and entities.

#### Environment

As said above, the environment is storage place of all entities that are going to join the simulation. Since we are facing a 2D board game, the environment implemented as a grid with cells. Each cell can accommodate up to one zombie.

On top of the grid, there is the environment class which is able to query and contact the zombies inside the grid. In addition, all the outer communication from the environment is managed by the environment manger. Its purpose is to pass the environment to the learning agents command and process/reshape the environment state before sending forward.

Over view of the above:

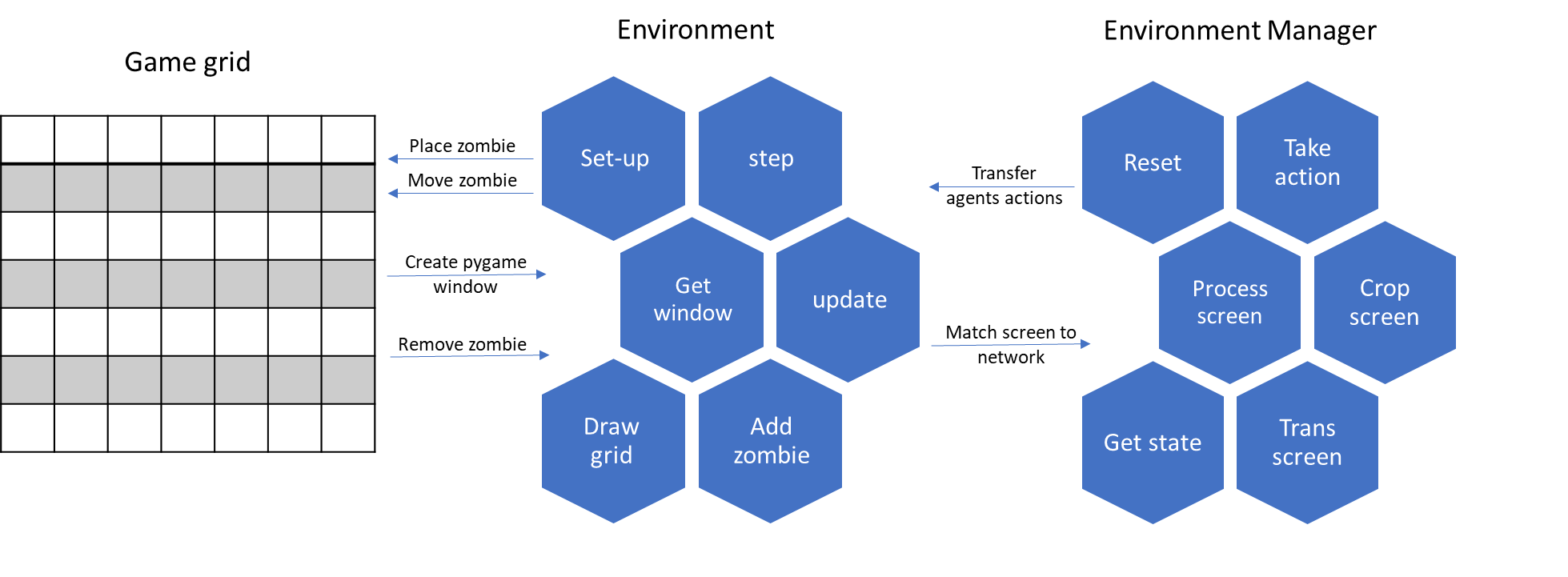


Figure 1 – simulation architecture

#### Entities

There are four types of entities living in our environment that are:

* Light master
* Light
* Zombie master
* Zombie

The light and zombie masters are intelligent agents with the ability to create zombies and move the light accordingly. On the other hand, the zombie and light per se are only data entities that are using to visualize the state in some conditions.

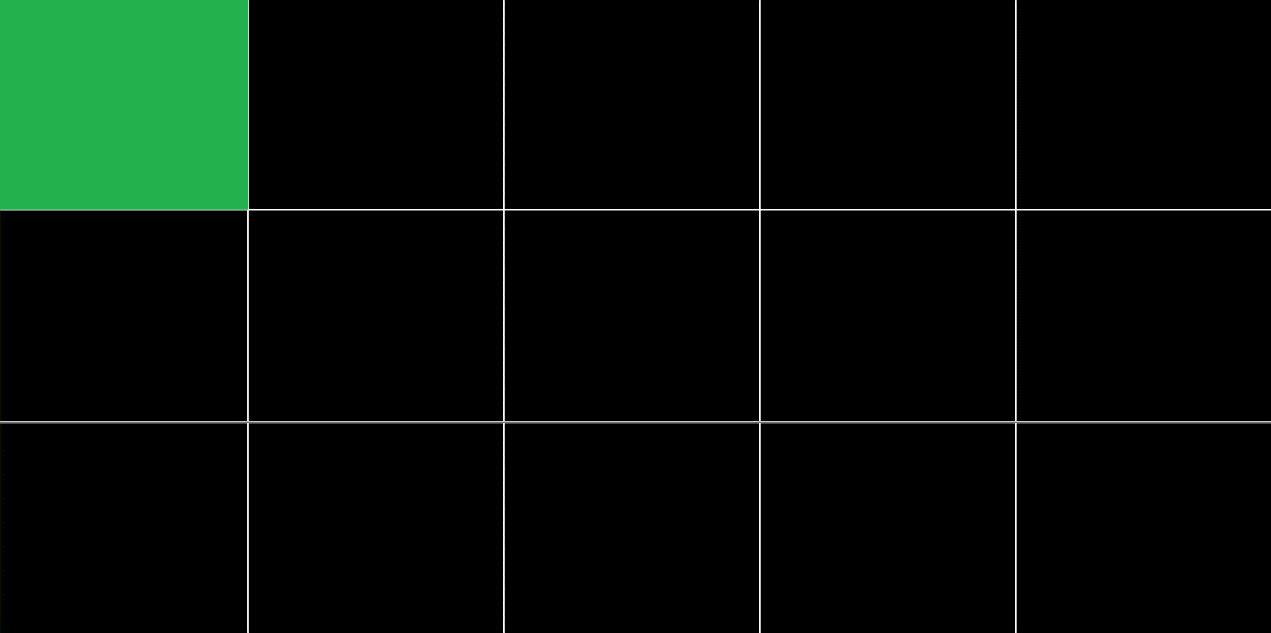
### Performance test

Usually in RL projects, we will use some known and tested environment, since that’s not the case, we have to test the performance of the environment with some simple scenarios in order to prove sanity and stability.

#### Zombie master test on a 3x5 board

As of the first test of learning, consider the zombie master that learns alone while the light master is forced to take some predetermined action.

At the beginning, we implemented the DDQN algorithm for the zombie agent with grid of 3x5 that look like:



0

1

2

Figure 2 – environment set-up for zombie master performance check with optional actions

As we can see in Figure 2, the light master is forced to take the top left cell as action in every step as stated above.

Consider the following parameters for the learning process:

|  |  |
| --- | --- |
| Light action | 0 |
| Target update | 10 |
| Num episodes | 1000 |
| Steps per episode | 200 |
| Batch size | 256 |
| gamma | 0.999 |
| Eps start | 1 |
| Eps end | 0.05 |
| Eps decay | 0.00001 |
| Memory size | 1000 |
| Learning rate | 0.001 |

With a deep NN of three layers, all fully connected (called 'Linear' in pytorch formulation): Linear (15,32), Linear (32,16), Linear (16,3).

We achieved a convergence in the amount of times the zombie master chose to send a zombie from the top row (the worst decision it could make):

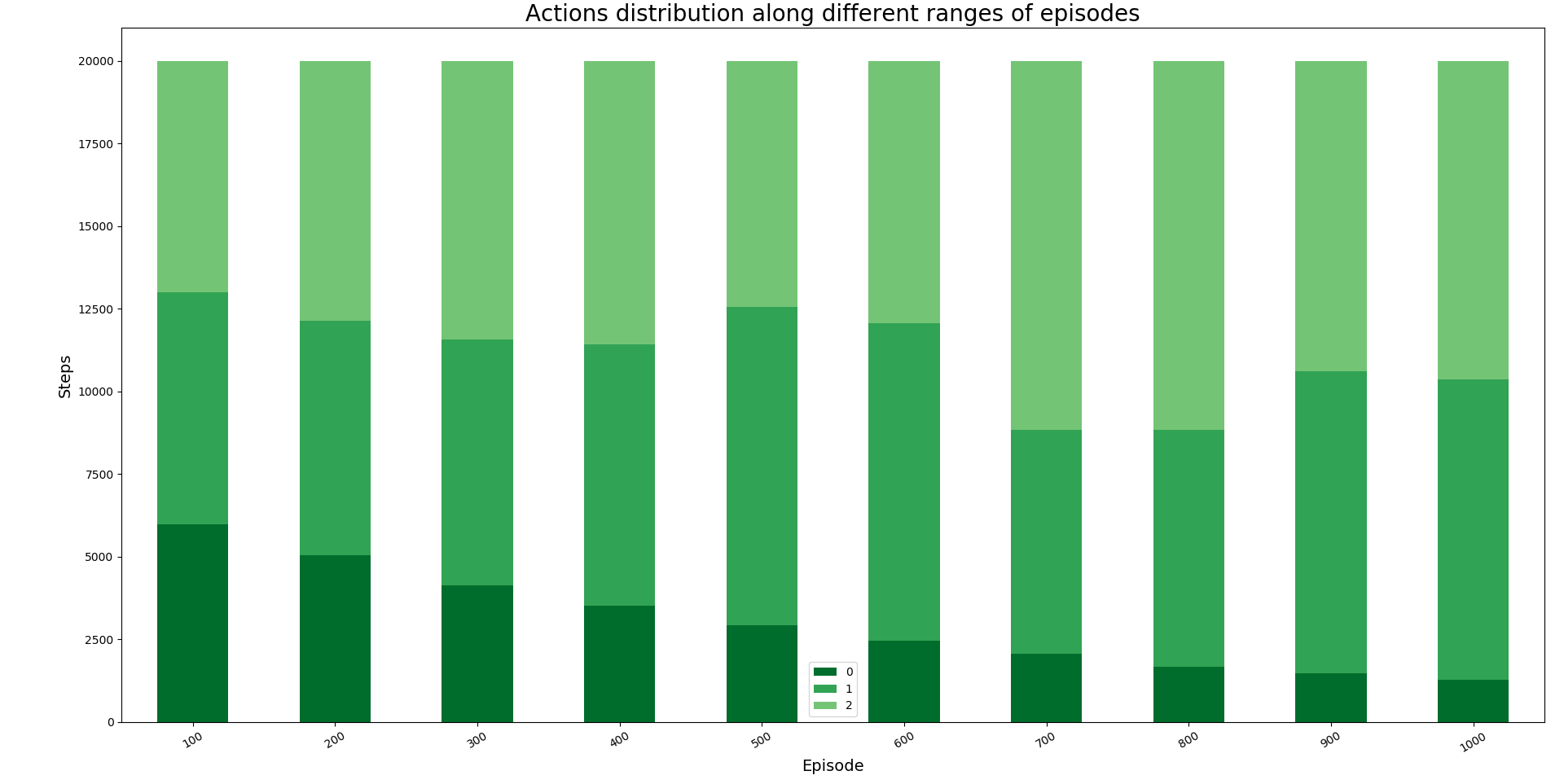


Figure 3 – actions distribution along different ranges of episodes

In Figure 3 we can notice the fading of the darker green which means that the zombie master decides to choose action one or two outright as the episodes go on.

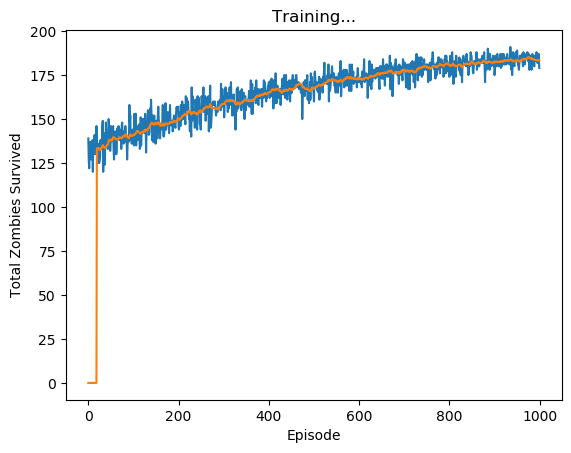
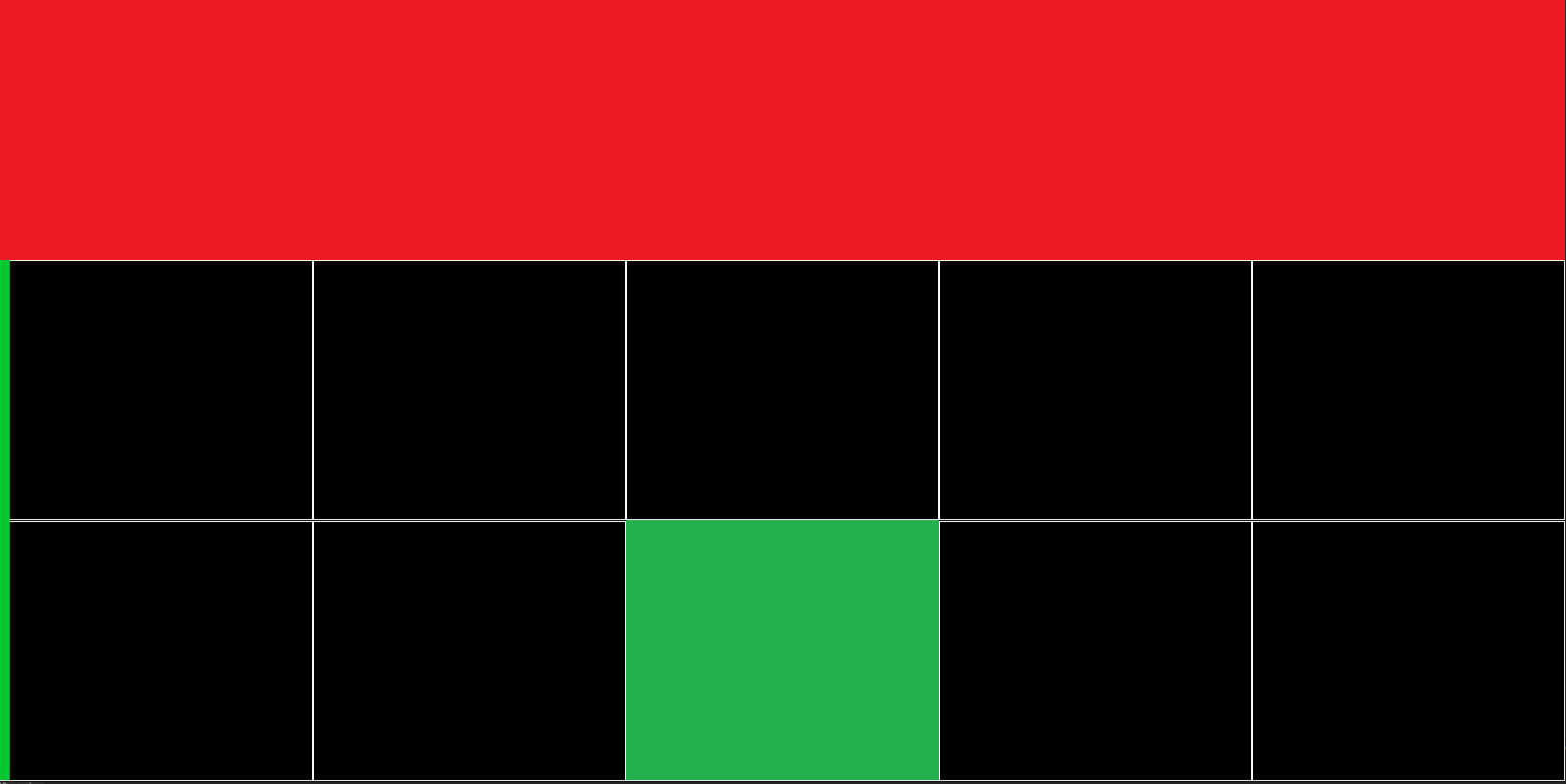


Figure 4 – Total zombies survived vs. the episodes (blue) with its moving average (orange)

Plus, we can tell from Figure 4, the zombie master reward is almost at the maximum he can get – reaches 185 zombies from possible of 195 (there are 200 steps with grid width of 5), something that can be explained be the epsilon greedy ending parameter at 0.05 which means that at the end of the learning, the zombie master is still taking random steps with 5% chance.

#### Light master test on a 3x5 board



7

8

6

5

9

0

1

2

3

4

10

11

13

14

12

Figure - environment set-up for light master performance check with optional actions

Figure 5 illustrates the simulation while testing the performance of the light master.   
As we can see, the zombie master takes only the action 0 (predetermined for simplicity) which only affects the exit of the zombies from the upper cell