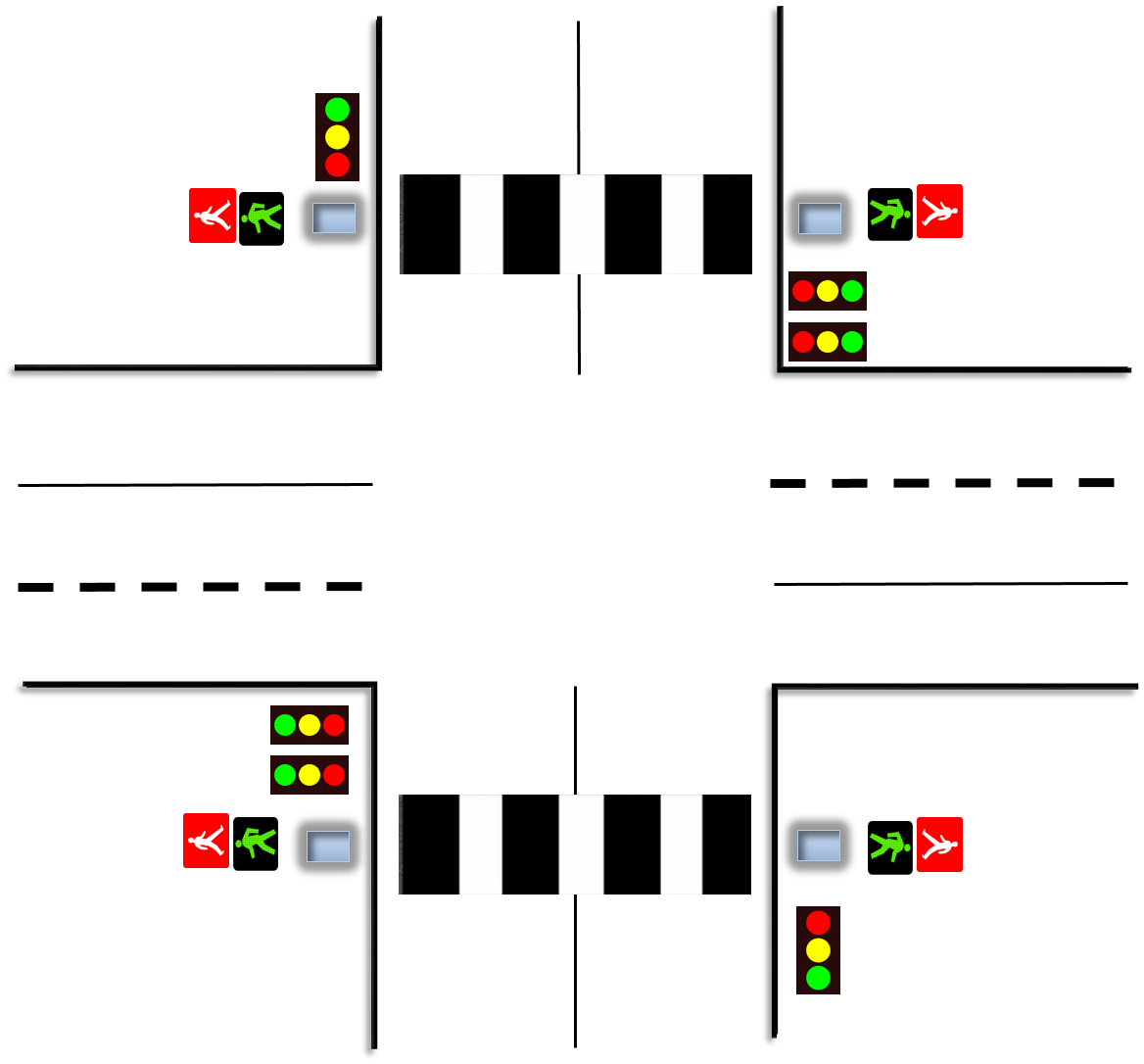
 Test Plan



* Introduction

The Test Plan documents and tracks the necessary information required to effectively define the approach that will be used in the testing of the project’s product. The document is created during the planning phase of the project. It is not static and it’s updated on an on demand basis.

* Testing Objectives

### 

The purpose of testing is to verify the functionality of all components.

We are going to test the following features:

* Add crossing
* Remove crossing
* Add an element (sensor or traffic light)
* Remove element
* Save file
* “Save as” file
* Open an existing file
* Change the green time of the traffic light
* Set the flow for cars/pedestrians
* Start simulation
* Clear the screen
* Testing Approach

### 

For now we are going to use *Static Testing* which test the component or specifications without execution of that software. This approach is usually done as soon as acceptance criteria or business requirements are ready for review before code implementation,

Later on the SDLC line, we are going to switch on System Testing. The purpose of the system testing is to validate that the complete and integrated system compiles with functional requirements and business requirements.

* Criteria’s

## eNTRY CRITERIA

The testing may start when the following criteria have been satisfied:

1. All code have been unit tested and passed
2. Test environment including software have been setup and configured correctly.

## sUSPENSION CRITERIA

The testing will be suspended under the following condition:

1. Critical error(s) found preventing test completion.
2. Change of the requirements.
3. Change of environment components or technology including different versions.

## Exit criteria

1. All the requirements have been tested.

**Functional testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test №** | **Test name** | **Expectations** | **Result** | **Description** |
| **1.** | Add a crossing | Insert crossing on the simulation screen. | Inserts crossing on the simulation screen | None. |
| **2.** | Remove a crossing | Removes the desired crossing from the simulation screen. | Removes the desired crossing  From the simulation screen. | None. |
| **3.** | Add an element (sensor or traffic light) | Insert the element on the simulation screen. | Inserts the element on the simulation screen | None. |
| **4.** | Remove an element |  |  | None. |
| **5.** | Open/Load | Loads the selected simulation file of the user on the application. | Loads the selected simulation file. | None. |
| **6.** | Change the greentime of the traffic light | Changes the green time of the traffic light to the selected value | Changes the green time of the Traffic Light to have selected value. | Missing |
| **7.** | Set the flow for cars/pedestrians | Increase/Decrease the traffic of all crossings of the exact simulation. | Increase/Decrease the traffic of all crossings of the exact simulation. |  |
| **8.** | Start the simulation | Car objects should start moving and stop/go on red/green traffic lights respectively | Car objects start moving and stop/go on red/green traffic lights respectively. | None. |
| **9.** | Pause the simulation. |  |  |  |
| **10.** | Clear the screen |  |  |  |

Test case ID: 1.a

Preconditions: Program is running

|  |  |  |  |
| --- | --- | --- | --- |
| Step No. | Step description | Test data | Expected result |
| 1. | User clicks on the button to add a crossing |  | Types of crossings are showed |
| 2. | User clicks on the type of crossing he wants to add | Type1 crossing | Crossing is loaded and ready to be placed |
| 3. | User selects the place to add the crossing | Place crossing | Crossing is drawled on the defined spot |

**Post condition: A new crossing is inserted**

Test case ID: 1.b

Test case description: Insert/add crossing

Preconditions: Program is running

|  |  |  |  |
| --- | --- | --- | --- |
| Step No. | Step description | Test data | Expected result |
| 1. | User clicks on the button to add a crossing |  | Types of crossings are showed |
| 2. | User clicks on the type of crossing he wants to add | Type1 crossing | Crossing is loaded and ready to be placed |
| 3. | User clicks the remove button | Loaded crossing is unloaded | Exception is thrown Error message is displayed |

**Post condition: A new crossing is inserted**

Test case ID: 1.c

Test case description: Insert/add crossing with already 12 crossings on the plot

Preconditions: Program is running and 12 crossings are visible on the screen

|  |  |  |  |
| --- | --- | --- | --- |
| Step No. | Step description | Test data | Expected result |
| 1. | User clicks on the button to add a crossing |  | Types of crossings are showed |
| 2. | User clicks on the type of crossing he wants to add | Type1 crossing | Crossing is loaded and ready to be placed |
| 3. | User selects the place to add the crossing | Place crossing | Error message is being displayed which states that the maximum number of crossings is reached |

**Post condition: A new crossing is not inserted**

Test case ID: 2.a

Test case description: Remove crossing

Preconditions: Program is running and there are already inserted crossings on the plot

|  |  |  |  |
| --- | --- | --- | --- |
| Step No. | Step description | Test data | Expected result |
| 1. | User clicks on the button to remove a crossing |  | A list of all inserted crossings is displayed |
| 2. | User clicks on the type of crossing he wants to remove and clicks OK | First crossing | Crossing is removed from the plot |

**Post condition: The desired crossing is removed**

**Post condition: A new crossing is inserted**