

Design Document

First version



Bilger yahov

oLEKSANDR SUPRUNENKO

ILIA NIKUSHEV

GEORGI CHISHIRKOV

LYUBOMIR DIMOV

MENGCHUAN LIU

Traffic Lights System

Table of Contents

[Introduction 2](#_Toc446414621)

[Class Diagram 4](#_Toc446414622)

[Renderable inheritance overview 4](#_Toc446414623)

[Component classes 4](#_Toc446414624)

[Crossing classes 5](#_Toc446414625)

[Simulation classes 6](#_Toc446414626)

[Crossing container (Recycle bin and saved crossings) 7](#_Toc446414627)

[Undo classes 7](#_Toc446414628)

[7](#_Toc446414629)

[Traffic manager Grid and System state classes 8](#_Toc446414630)

[Description of the classes and their members [early draft] 9](#_Toc446414631)

[Sequence Diagrams 10](#_Toc446414632)

[Select a crossing to place 10](#_Toc446414633)

[Place a crossing 10](#_Toc446414634)

[Remove a crossing 11](#_Toc446414635)

[Create a simulation 12](#_Toc446414636)

[Save a simulation 13](#_Toc446414637)

[Load a simulation 14](#_Toc446414638)

[Edit a road traffic flow 15](#_Toc446414639)

[Start a simulation 15](#_Toc446414640)

[Stop a simulation 16](#_Toc446414641)

[Pause a simulation 16](#_Toc446414642)

[Restart a simulation 17](#_Toc446414643)

[Undo an action 17](#_Toc446414644)

[Redo an action 18](#_Toc446414645)

[Save simulation results 19](#_Toc446414646)

[Show the help window 19](#_Toc446414647)

[Exit application 20](#_Toc446414648)

[Override simulation 21](#_Toc446414649)

[Show simulation result 21](#_Toc446414650)

[Select crossing’s component to make changes 22](#_Toc446414651)

[Set current active crossing 22](#_Toc446414652)

[Start simulating pedestrians 23](#_Toc446414653)

[Access “Saved” crossing 24](#_Toc446414654)

[Access “Removed” crossings 25](#_Toc446414655)

[Empty the recycle bin 25](#_Toc446414656)

# Introduction

This document gives information about the class diagram for the *“Traffic Lights”* application. Description of the classes and the attributes and methods in each class is given. Furthermore, some sequence diagrams of the application are presented.

The class diagram is a static diagram. It represents the static view of the application. Our class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system.

Our class diagram can be mapped directly with object oriented languages. It shows a collection of classes, interfaces, associations, collaborations and constraints.

The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community.

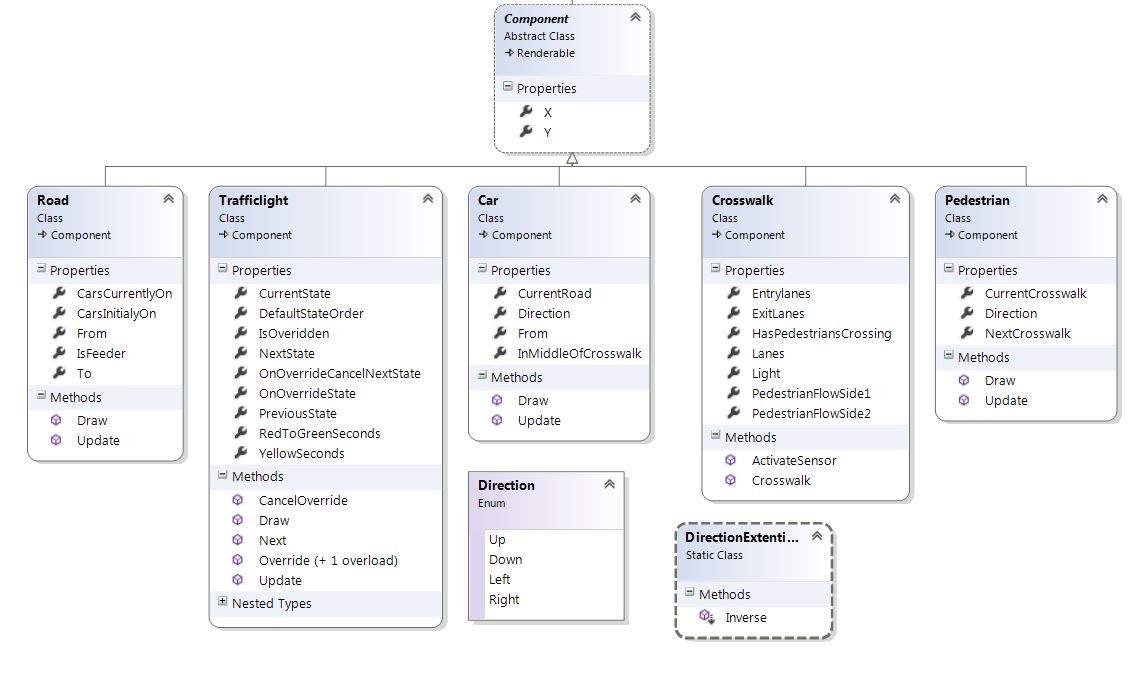
In the document can be found sequence diagrams which purpose is:

* Draw the activity flow of a system.
* Describe the sequence from one activity to another.
* Describe the parallel, branched and concurrent flow of the system.

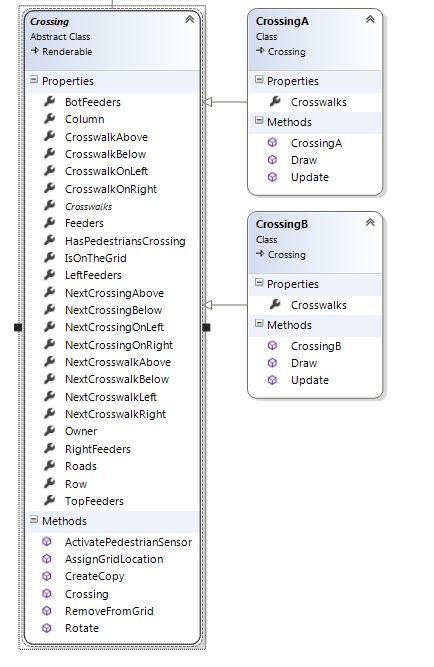
# Class Diagram

## C:\Users\user\Desktop\ProCP\Renderable.JPGRenderable inheritance overview

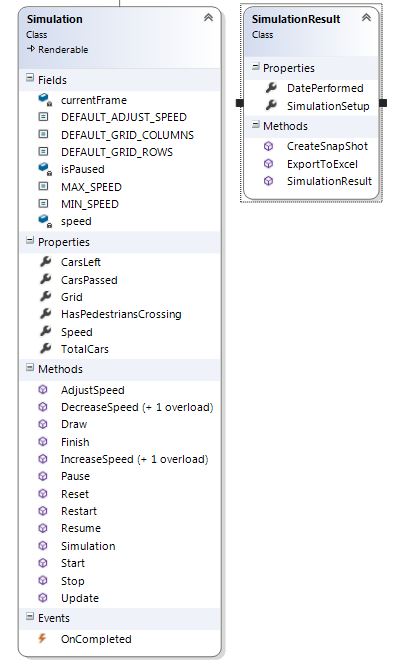
## Component classes



## Crossing classes



## Simulation classes

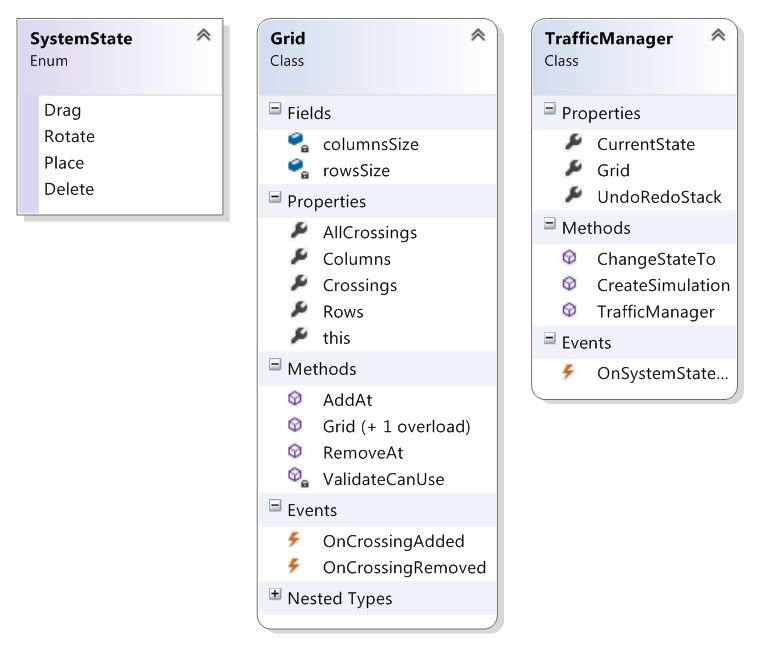


## C:\Users\user\Desktop\ProCP\Bin and Cpaste.JPGCrossing container (Recycle bin and saved crossings)

## Undo classes

## C:\Users\user\Desktop\ProCP\UndoRedo.JPG

## Traffic manager Grid and System state classes



# Description of the classes and their members [early draft]

ActionStack - Container that handles Undo and Redo of actions

AddCrosswalkAction, MoveCrosswalkAction, UpdatePropertiesAction, RotateAction – various actions that can be undone and redone in the system

Car – a renderable component within the simulation that moves on a road

Component – abstract class containing coordinates for various objects within the simulation

Crossing – abstract class defining the basic properties and methods for the crossings

CrossingA – implements Crossing and is the basic type, with no crosswalks

CrossingB – implements Crossing, similar to type A but has 2 parralel crosswalks on each side

SimulationResults – extracts the results from a simulation into an Excel

Trafficlight – a component that controls the movement of cars and pedestrians

UndoableAction – abstract class containing basic logic for undoing and redoing of actions

SystemState – current state of the system

Pedestrian – a component that moves on the sidewalks and crosswalks of the roads

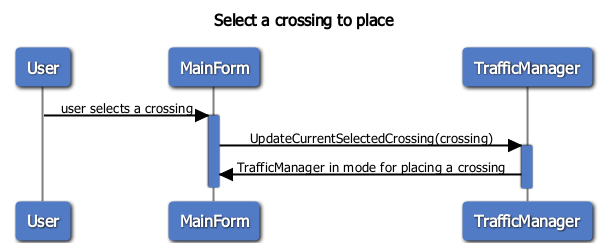
Grid – represents 1 square of the 3x3 grid within the system

RecycleManager – the recycle bin where removed crossings are stored

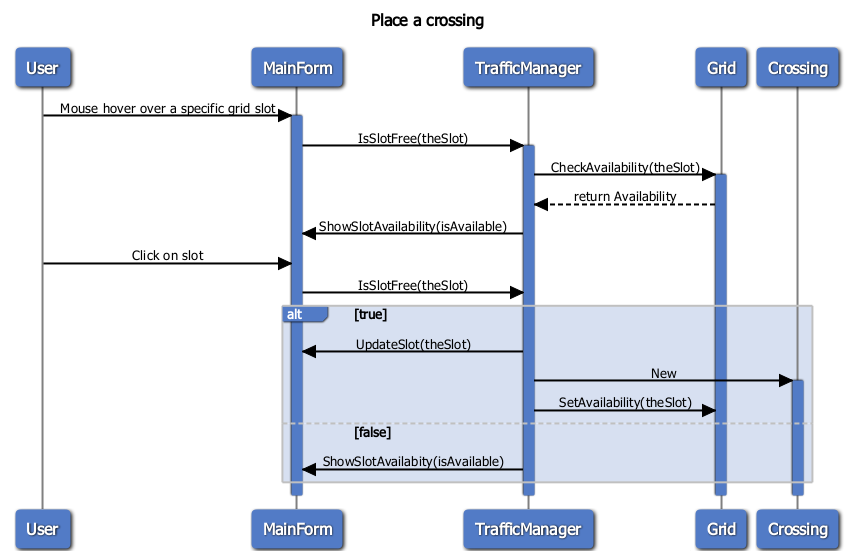
SavedManager – the place where user saved crossings are stored with predefined properties

# Sequence Diagrams

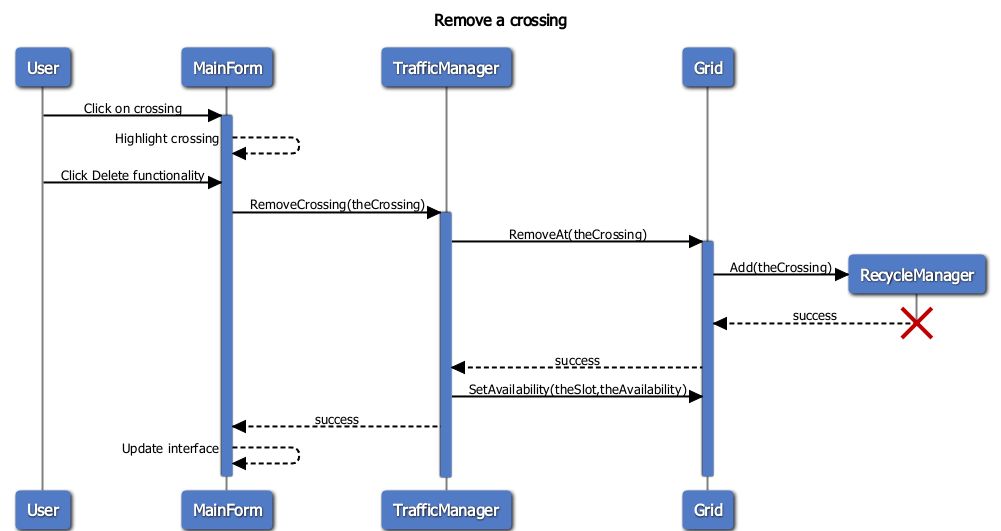
## Select a crossing to place



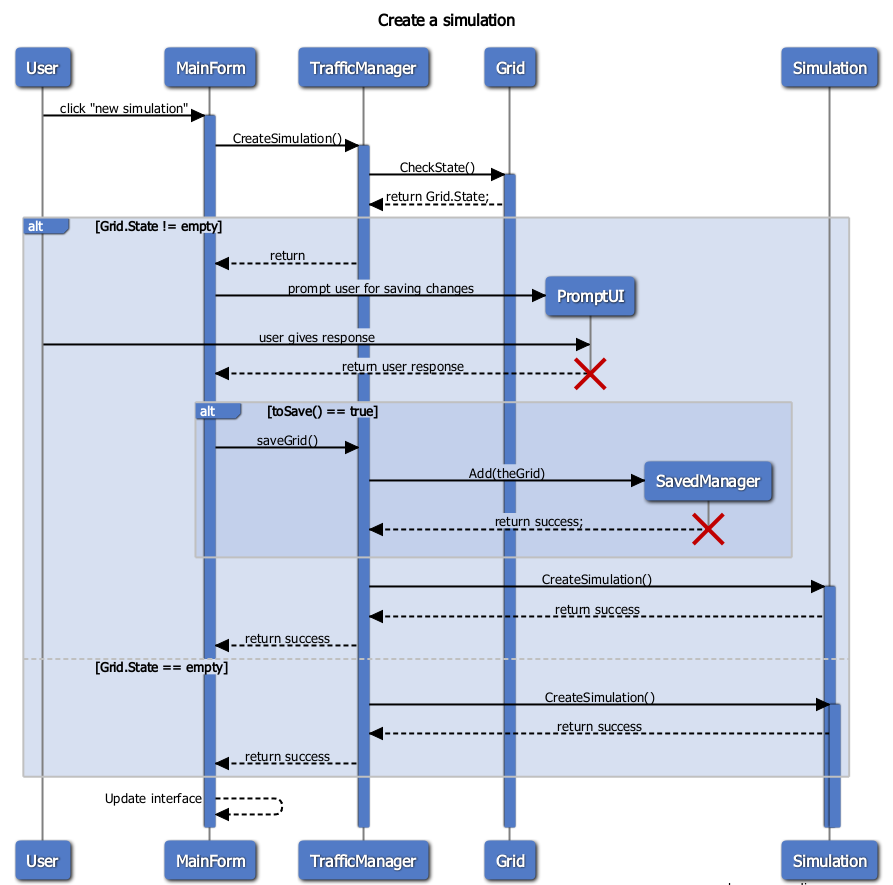
## Place a crossing



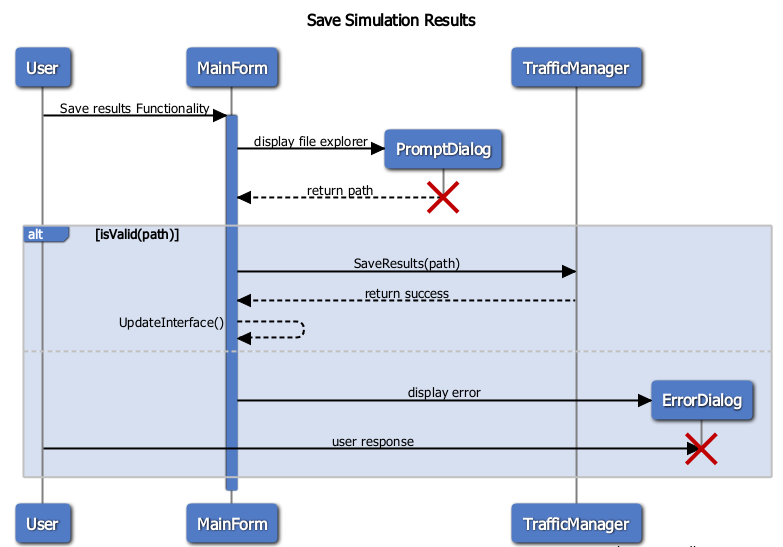
## Remove a crossing



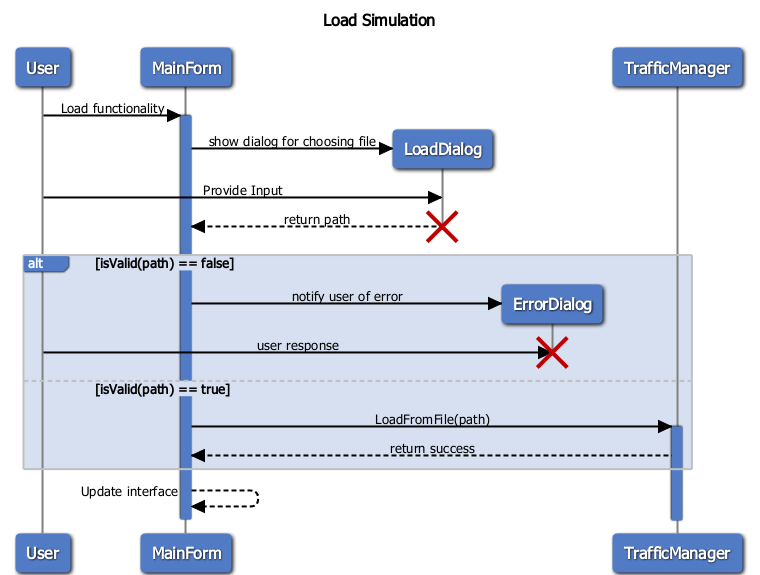
## Create a simulation



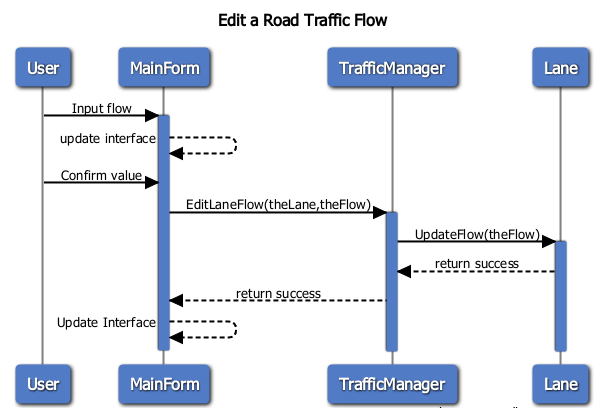
## Save a simulation



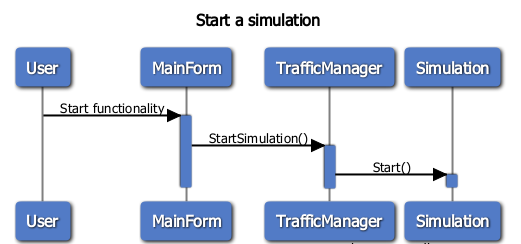
## Load a simulation



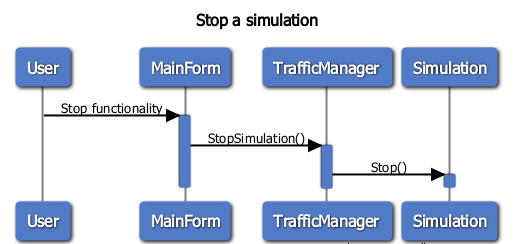
## Edit a road traffic flow



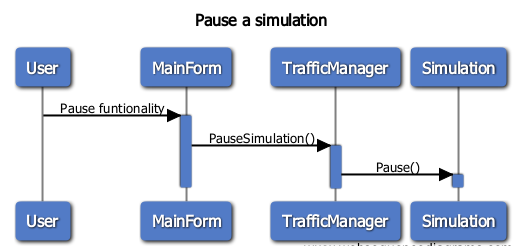
## Start a simulation



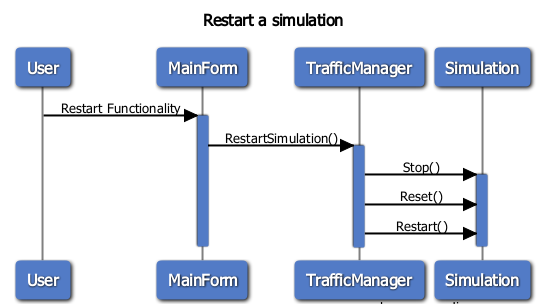
## Stop a simulation



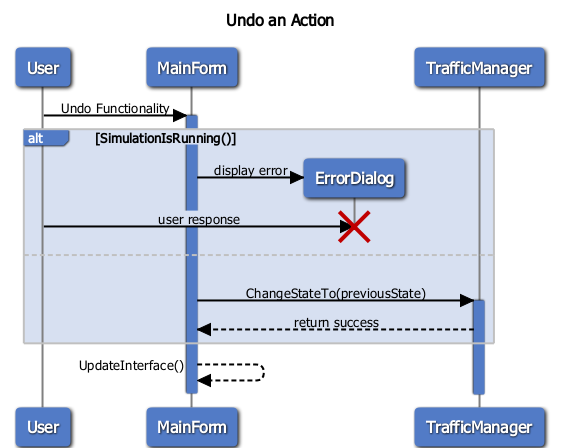
## Pause a simulation



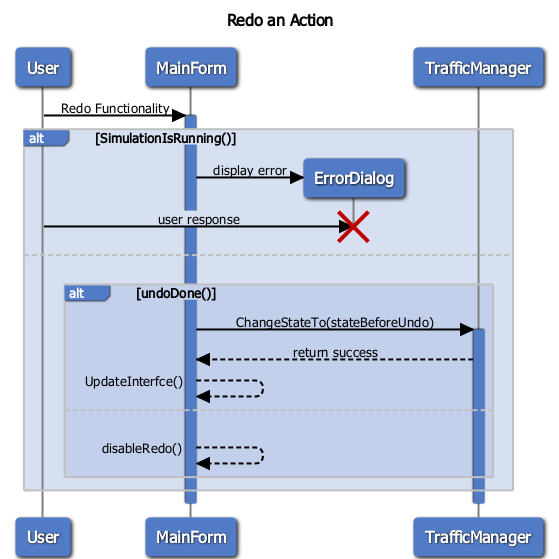
## Restart a simulation



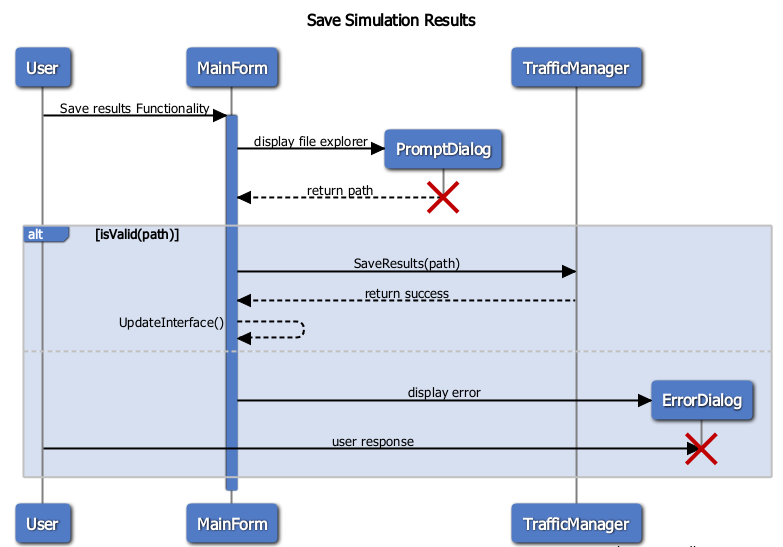
## Undo an action



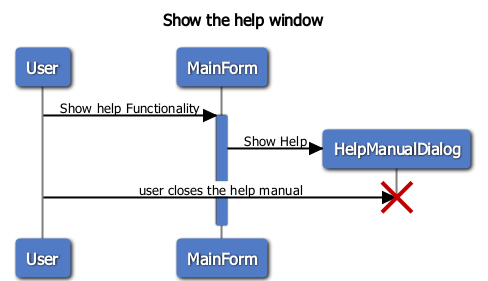
## Redo an action



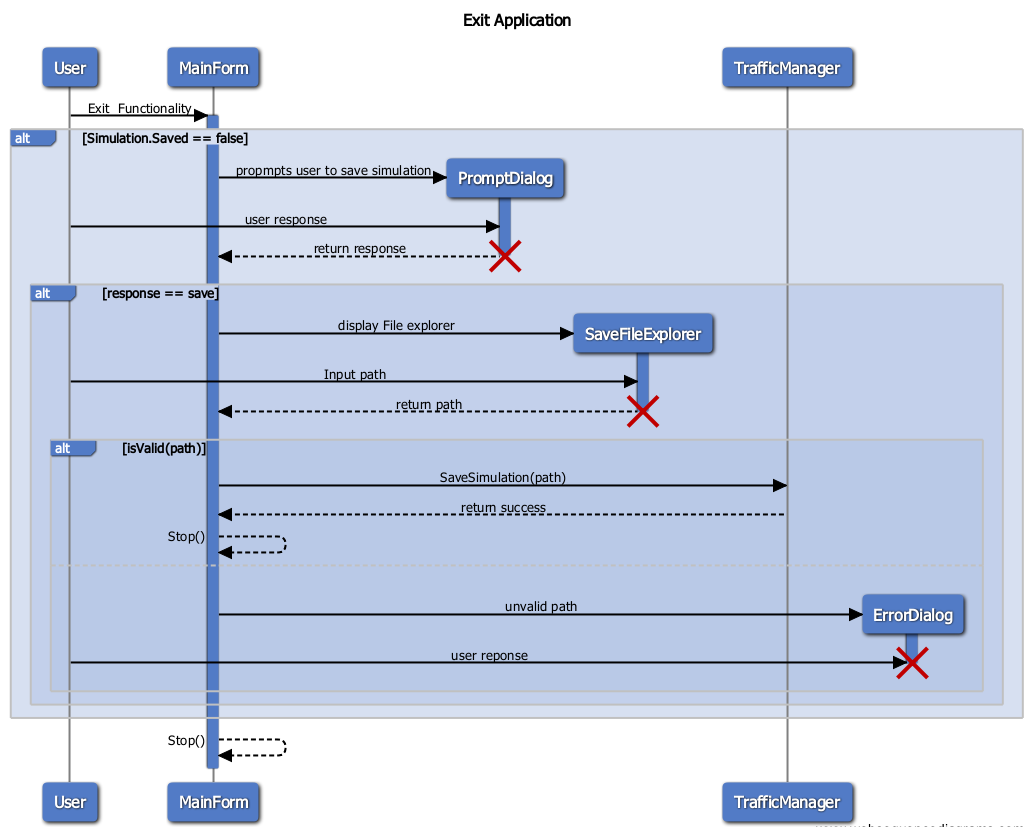
## Save simulation results



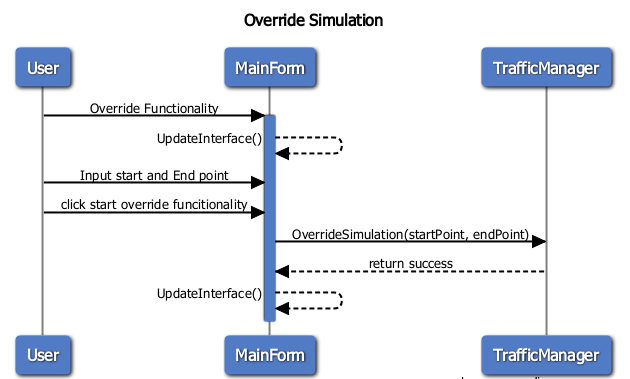
## Show the help window



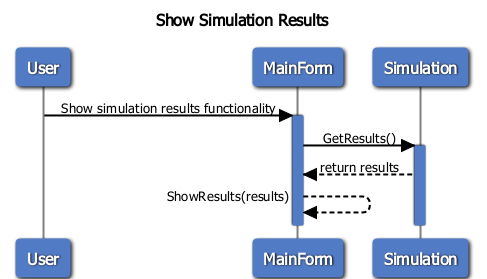
## Exit application



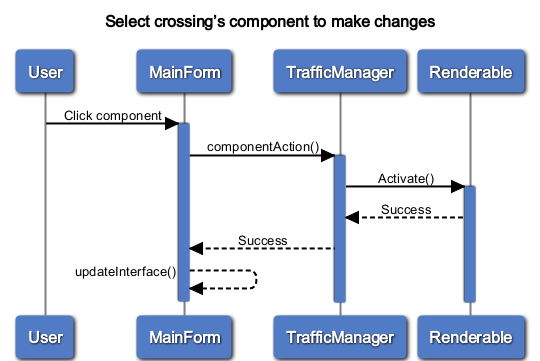
## Override simulation



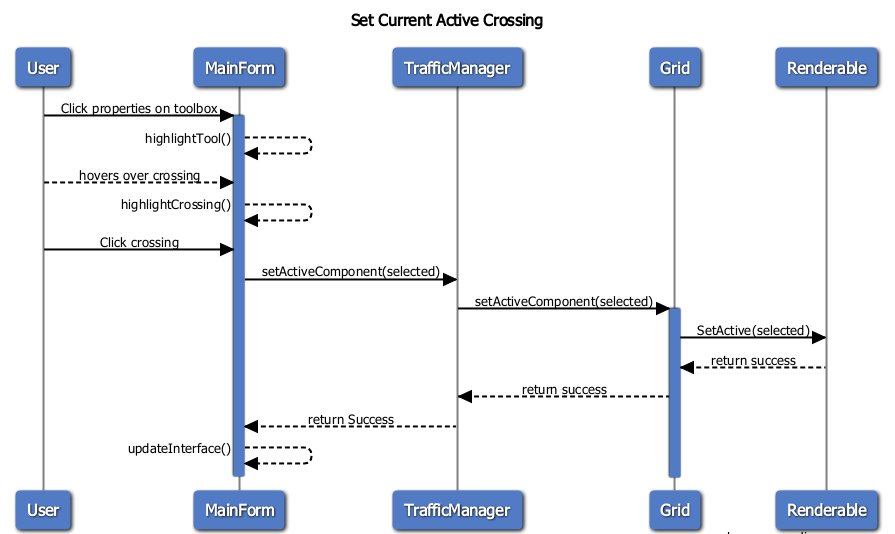
## Show simulation result



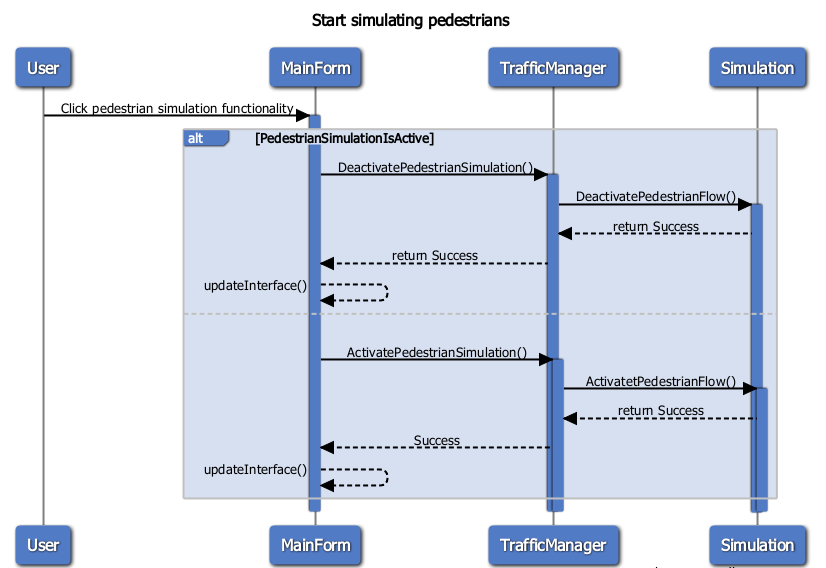
## Select crossing’s component to make changes



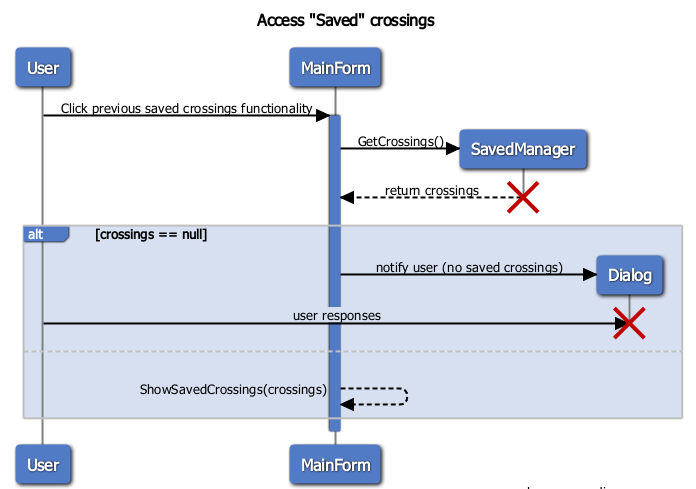
## Set current active crossing



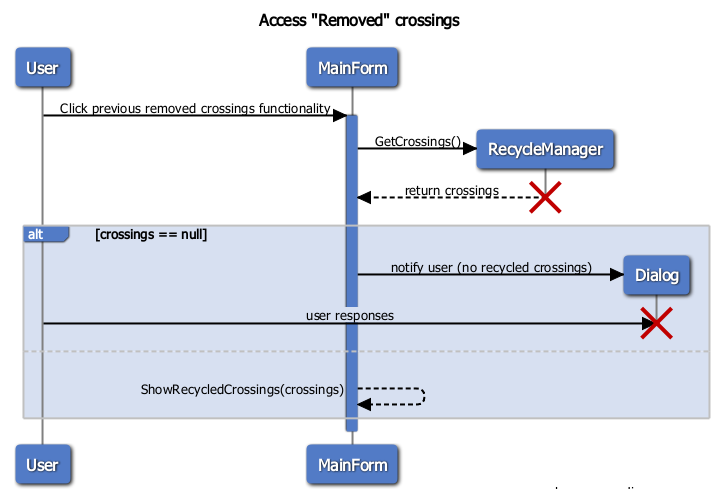
## Start simulating pedestrians



## Access “Saved” crossing



## Access “Removed” crossings



## Empty the recycle bin

