Explaining the code that we are using for the LIT project.

By Jelle Schroijen, 27/02/2020



**Contents**

[Description 3](#_Toc33385690)

[The Graphical namespace 3](#_Toc33385691)

[The Graphs namespace 3](#_Toc33385692)

[Objects 4](#_Toc33385693)

[Graphical Processing 4](#_Toc33385694)

[Graphs 5](#_Toc33385695)

[UML Diagrams 6](#_Toc33385696)

[Graphical Processing 6](#_Toc33385697)

[Graphs 6](#_Toc33385698)

# Description

We have made a C# library containing many of the objects we will be using to make the LIT program, we have chosen to do this so that if we make more than one version, we can base them all on the underlying framework, and when we update that framework all of the versions will be updated as well.

The library consists (as of now) of two namespaces:

*Library.Graphical* and *Library.Graphs*.

## The Graphical namespace

The graphical namespace, as its name suggests, will take care of turning the ‘imaginary’ objects into something visual. We made three base objects, 2DPoint; 2DLine and 2DCircle, of which 2DFiniteLine is a subclass of 2DLine. The 2DPoint class will be the base class for objects like the fire extinguisher and person class because those don’t really represent more than a point in the layout. The 2DCircle will be the base class for a fire object because the size of it will increase over time, and the 2DFiniteLine will be the base class for the wall object because it will be easy to implement it based on a starting point and ending point.

Using these base classes also makes it easy to add any objects in the future. This will ensure code consistency if we make sure they are extending the right base class and implementing the right interfaces.

We also used two interfaces: the IChangeableObject and the IPaintable. The objects like the fire and person will implement the IChangeableObject interface, this will allow us to easily ‘animate’ these objects as we can just store an array of these IChangeableObjects and call the Tick function inside of them. The objects like fire, person, fire extinguisher and wall will all implement the IPaintable interface, this will allow us to easily paint all these objects into something like a picture box as we can just store an array of these IPaintable objects and call the paint function inside of them.

## The Graphs namespace

The graphs namespace will be taking care of the graphs that we might use to calculate the shortest path from one object to another. Currently, there’s one limitation to the code we have implemented; when making a graph object, all objects inside it must be the same type. We can take care of this by making a base class that all the objects we will use in the graph will be based on.

The Graph object will store the roots of the trees we can use to do pathfinding. It is possible to add root nodes after the initialization of the Graph object. An Edge object represents the connection to a node. This object is one-directional and will hold a weight value and the Node object it is referencing. The Node object will store all the Edges it is connected to, except for the one that is referencing itself.

# Objects

## Graphical Processing

*Note: the class names that start with a number are called prefixed with an underscore.*

Interfaces

IChangeableObject

IPaintable

Classes

2DPoint

2DLine

2DFiniteLine

2DCircle

FlammablePoint

FireExtinguisher

Fire

Person

Wall

2DFiniteLine extends 2DLine

FlammablePoint extends 2DPoint

FireExtinguisher extends FlammablePoint

Fire extends 2DCircle

Person extends 2DPoint

Wall extends 2DFiniteLine

2Dline uses 2DPoint

2DFiniteLine uses 2DPoint

2DCircle uses 2DPoint

Person uses FireExtinguisher

FireExtinguisher implements IChangeableObject

Person implements IChangeableObject

FireExtinguisher implements IPaintable

Fire implements IPaintable

Person implements IPaintable

Wall implements IPaintable

## Graphs

Classes

Edge

Graph

Node

Edge uses Node

Graph uses Node

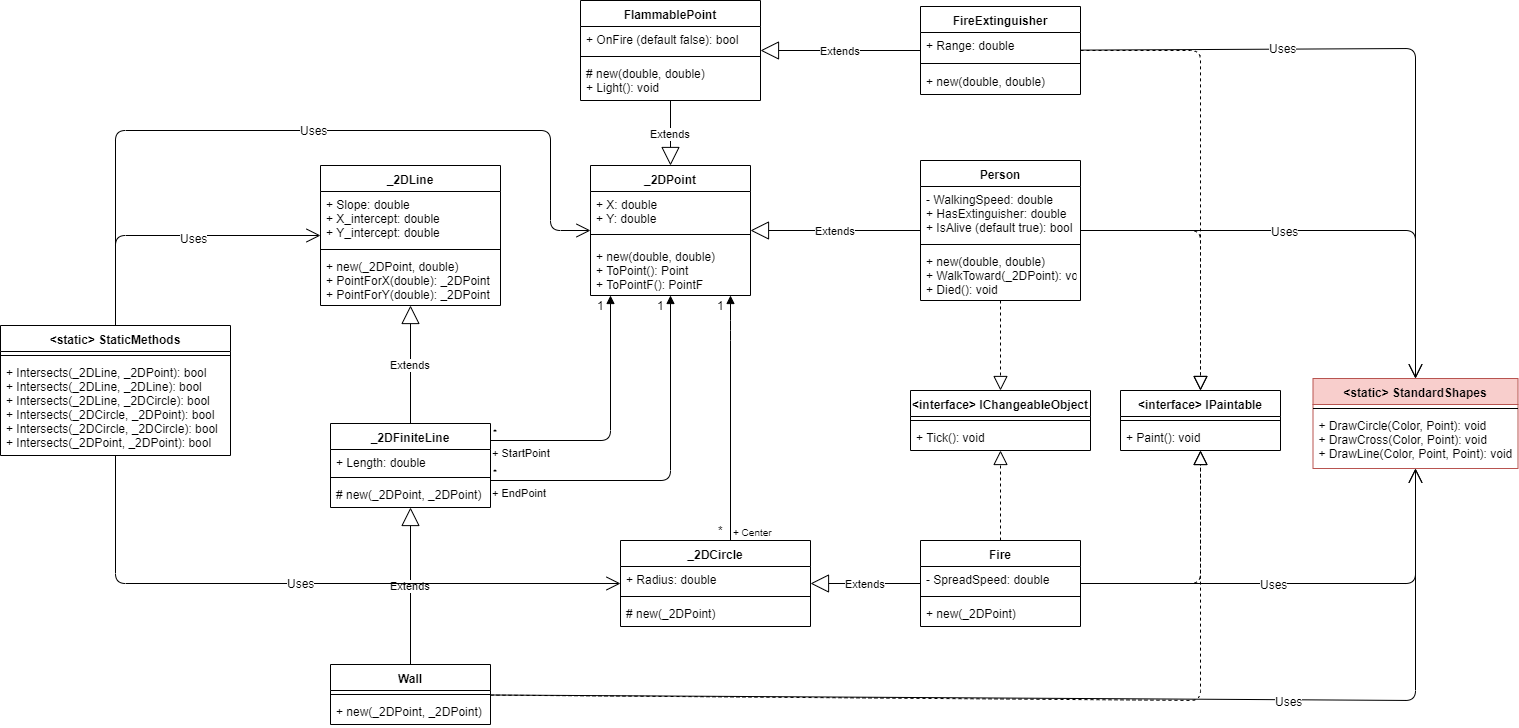
Node uses Edge

# UML Diagrams

*Note: these are, by any means, not the final versions of the object diagrams. They will surely be changed* ***a lot*** *in the future.*

## Graphical Processing

*Note: the objects with a red outline are yet to be implemented, they are just here to note that they would be a handy addition.*



## Graphs