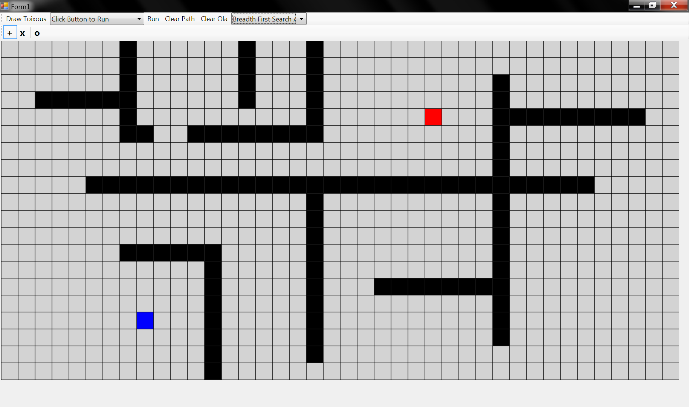
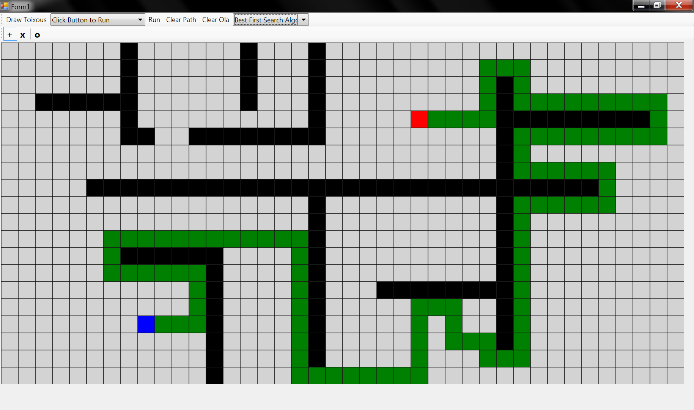
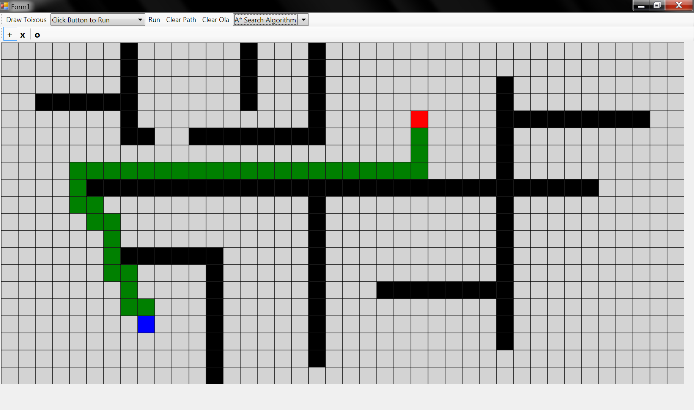
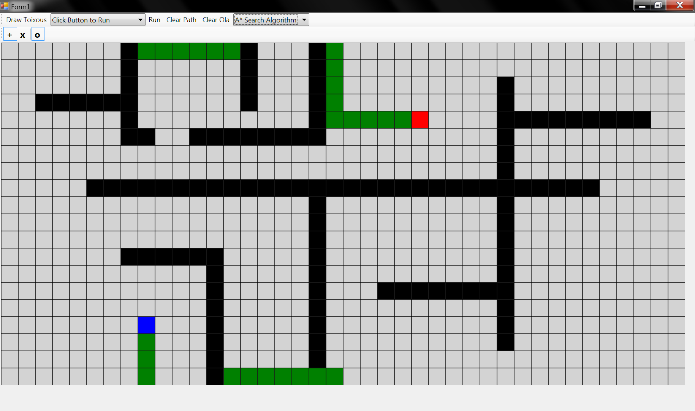
Εύρεση διαδρομής σε γράφημα τύπου Grid (Grid Path Finding)

Σε αυτό το πρόγραμμα χρησιμοποιούνται 3 αλγόρυθμοι για την εύρεση διαδρομής σε γράφημα:

1. Best First Search
2. Breadth First Search
3. A\* Search

Το γράφημα είναι grid και η απόσταση ανάμεσα σε 2 γειτονικά κελιά θεωρείται πάντα ίδια. Για αυτό μπορούν να χρησιμοποιηθούν οι 2 τελευταίοι αλγόριθμοι οι οποίοι είναι ευριστικοί (κάνουν εκτίμηση για την απόσταση 2 κελιών).

Στην εφαρμογή υπάρχουν επιλογές για κίνηση διαγώνια και κίνηση εκτός του grid συνεχίζοντας από την αντίθετη πλευρά. Επίσης η διαδρομή μπορεί να υπολογίζεται κάθε φορά που ο χρήστης κουνάει το ποντίκι πάνω από το grid.

Εικόνα 4: Εύρεση Διαδρομής με Α\* σε torus επίπεδο

Εικόνα 3: Εύρεση Διαδρομής με Α\*

Εικόνα 2: Εύρεση Διαδρομής με Best First Search

Εικόνα 2: Ο λαβύρινθος

*Ενδιαφέρον: Ο Best First δεν επιστρέφει πάντα την βέλτιστη (μικρότερη σε απόσταση) διαδρομή σε αντίθεση με τον Α\* και φαίνεται στις φωτογραφίες.*

This is .NET Windows Forms application. It is based on the following project:

In this application 3 path-finding algorithms are tested. The user can draw walls in the map which is a 2-dimensional grid plane and is displayed by the grid.

Its is based on the .NET Framework 3.5, written in Visual Basic. It uses WPF for its core GUI functionality (the grid display). This functionality is integrated using the WindowsFormsIntegration .NET library and the ElementHost (System.Windows.Forms.Integration.ElementHost) control.

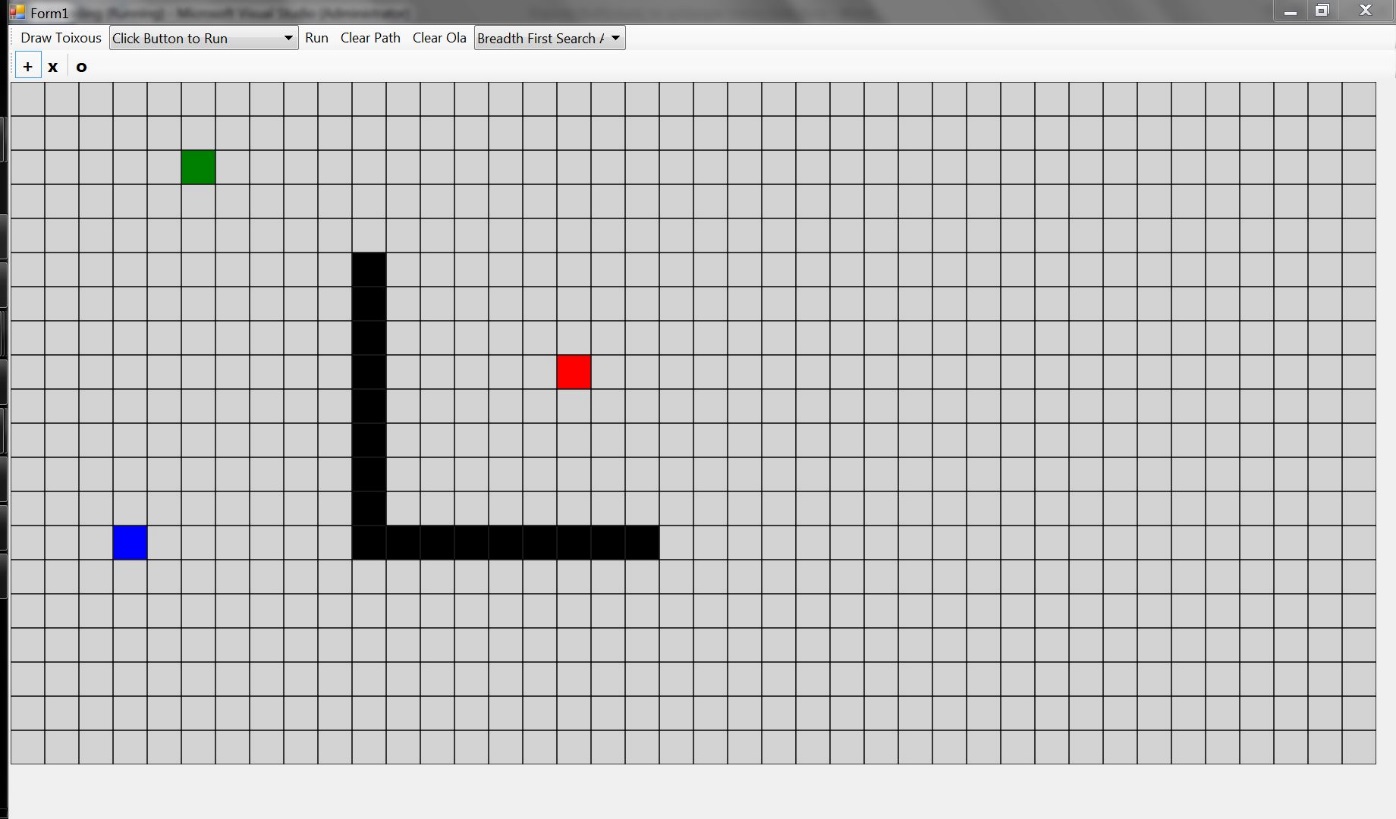
The user can:

* draw obstacles (walls) with the mouse
* set the origin (blue) and the destination (red)
* select what type of grid to use:
  1. bounded rectangle
  2. torus (top-down and left-right sides are adjacent)
* select which algorithm to use to find path:
  1. Best First Search
  2. Breadth First Search
  3. A\* Search

In more detail:

* draw obstacles (walls) with the mouse while in draw-mode (having selected the Draw button)
* set the origin (blue) and the destination (red) by clicking the right or left mouse button respectively
* select what type of grid to use:
  1. bounded rectangle
  2. torus (top-down and left-right sides are adjacent)
* select which algorithm to use to find path:
  1. Best First Search
  2. Breadth First Search
  3. A\* Search
* select how to start the path-finding process
  1. only when clicking the button "Run"
  2. when clicking on the map to set origin or destination
  3. when hovering and also holding down the the left or right mouse button, thus changing origin and/or destination and simultaneously calculating the new path.

# Instructions



Enter and leave Draw Mode by selecting this button.

Select adjacency by selecting any of these 2 buttons **"+"** and **"x"**.

Obstacle / Wall

Destination

Hovered Location

Origin

Clear everything.

Execute the path-finding algorithm.

Origin and destination must be set.

Select which path-finding algorithm to use.

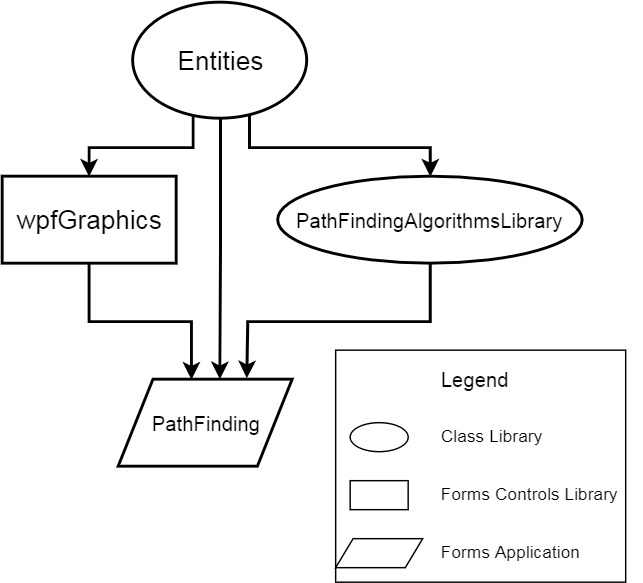
Clear only the path.

Select grid type by selecting this button **"o"**.

Select how to start the execution of the path-finding algorithm

# Dependency Diagram

The structure of the application is shown in this diagram:



made using [draw.io](http://www.draw.io/) desktop application

* The Entities class library contains basic definitions like the GridCoordinates class.
* The wpfGraphics forms controls library contains GUI elements. Currently it has only the RectangleGrid class which displays a grid using WPF graphics.
* The PathFindingAlgorithmsLibrary contains the implementation of the 3 path-finding algorithms. It may be extended in the future to include a Djikstra implementation.
* Finally, The PathFinding is the executable that needs to reference all the above libraries to work. The rest of the GUI is materialized here.