**Title:**

**Topological Navigation Editor**

**(ToNE)**



By

Laxwell Nyama

MComp Computer Science

2015/2016

Supervised by

Dr Marc Hanheide

Jaime Pulido Fentanes

1. **Abstract**

Graphic user interfaces have grown in popularity and use over the past few years with almost over three quarters of the world using them. This project seeks to develop an innovative graphical user interface to be used in mobile robotics for topological map edition. The intention is to develop a topological map editor graphical user interface (GUI), which is robust, effective and accessible, to be used by people with no technical knowledge or background, to control mobile robots.

This paper will present the implementation details of the topological editor using PyQt4. Some background on topological and metric maps will be provided and the evaluation of the artifact‬‬‬‬‬‬‬‬‬‬‬‬‬‬ and also critical reflection on the project as a whole.

1. **Acknowledgement**

A long and ardours journey it has been, towards the attainment of my BSc (Hons). It would not have been possible without the help of all those I came across at various stage of my project. I would like to extend my immense gratitude to all those who lent me their support in diverse ways during the trials and tribulations I encountered in the course of this project, those who helped me through cold winters, those who brought a smile to my face, gave me advice, motivation and encouragement to do the best of my ability.

First and foremost I would like to thank my supervisor, Dr Marc Hanheide. His mentorship, perseverance and willingness to help made it more bearable. His advices, guidance and support encouraged me to evolve from a struggling student into an ambitious researcher. He taught me how to effectively use communication tools available to improve collaboration. Thanks to many hours of discussion with him, I learnt how to undertake a research project. His aim for perfection helped me improve my research process and polish my skill-set.

I also extend my gratitude to J who is my co-supervisor for his contributions to the success of my project. As a pair-programming partner, I learnt a lot from him regarding the problem domain.

Lastly I would like to thank my family for the love, support and faith they showed in me, not to forget their encouraging words of wisdom.

**Table of Contents**

**Page**

1. Introduction 3
2. Aim and Objectives
   1. Aims 4
   2. Objectives 4
   3. Motivation 8
3. Requirements Engineering
   1. Overview
   2. Requirements
      1. Functional
      2. Non-functional
      3. Use case
4. Background Research 9
   1. Overview 9
   2. Maps
   3. Graphical User Interface 10
   4. Autonomous mobile robots and Navigation
   5. PyQt
5. Literature review
   1. Overview
   2. Maps
   3. Interface
   4. PyQt
6. Design 14
   1. Overview 14
   2. Design 14
   3. Product Design
      1. Paper prototype
      2. Class design
7. Implementation

6.1 Methodology

1. Evaluation
   1. Overview
   2. Systems Evaluation
   3. Methodology Evaluation
2. Resources/ Toolset Analysis 18
   1. Overview 18
   2. Development Platform 18
   3. Hardware requirements 18
   4. Software requirements 19
3. Conclusion 20
   1. Overview
   2. Objective Analysis
   3. Limitations
   4. Critical Reflection
   5. Further Development
4. Appendix
5. List of figures
6. References
7. **Introduction**

This project aims to develop a real world topological map editor graphical user interface (GUI) by improving on the existing techniques to advance the robot navigation capabilities, usability and user interfaces. This project is part of STRANDS project; STRANDS aims to enable a robot to achieve robust and intelligent behaviour in human environments through exploitation of long-term experience in security and care scenarios.

“Autonomous Mobile robotics has seen exponential growth in recent years and robot navigation is a growing area of scientific research. Without navigation the creation of self-propelled, household machines, guard robots, or planet surveyors is beyond imagination”. (Richárd Szabó, 2004). An easy and effective way of controlling such robots will vastly increase their applicability. The exponential growth in mobile robotics and robot use has called for easier and simpler ways to interact and control them, however, the control and use of these robots requires a high level of technical knowledge. The way these robots are currently controlled is too complex and technical for an average person, hence the need to design and implement a dedicated graphical user interface that is simple, effective and easy to use for people with little to no technical ability.

One of the basic elements of robot navigation is a map with which a mobile robot can perform localization and motion planning. In order for the mobile robot to operate properly, a map may need to be available a priori or constructed during operation. The two most common types of maps are topological and metric maps, which will be discussed in depth in the background section of this document. “The maps are used for localization, mapping and navigation which is the process of deciding and controlling the direction of travel, derived from localization and a given map.” (IEEE Std 1873, 2015).

“Robotic mapping has been a highly active research area in robotics and AI for at least two decades. Robotic mapping addresses the problem of acquiring spatial models of physical environments through mobile robots. The mapping problem is generally regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots. Despite significant progress in this area, it still poses great challenges.” (Sebastian Thrun, 2002). The development of a robust and effective topological navigation editor that can be used by people with no technical background will play a great part in the deployment of these robots. There is no question that research in the area of user interface software tools has had an enormous impact on current practice of software development. Virtually all applications today are built using window managers, toolkits, and interface builders that have their roots in the research of the 70’s, 80’s, and 90’s.

Below is a project development life cycle illustrating how the development of the editor will be achieved. The first phase is, understanding the user and systems requirements, followed by the second stage, which is planning. This involves but not limited to ordering the tasks identified by objectives and allocating time for completion of each task. The designing stage follows on taking the project a step further to achieving the project aim. Here, the product is prototyped and all the features need articulated in such way that will make it easy to implement the artefact. The documents from the design phase will serve as blueprints of the artefact.

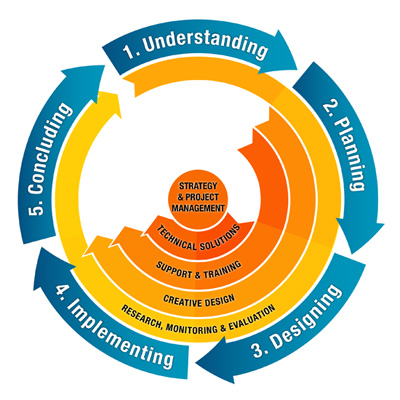


Figure 1: Project Development Life Cycle

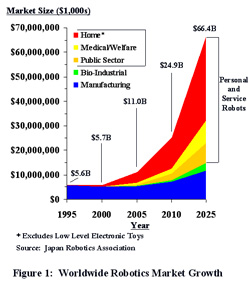
Fourth phase, the blueprints from the design stage are brought to life. The artefact is implemented. The implementation of the artefact will apply extreme programming development methodology as discussed in the methodology section of this paper. Once this stage is complete, the fifth stage is concluding the artefact, this involves but also not limited evaluation of the artefact with the client and the focus group as discussed in later sections of this paper and documentation.

1. **Aims and objectives**
   1. **Aim**: The aim of the project is to develop a dedicated Topological navigation editor user interface for robot systems that can be used by people with no technical background.
   2. **Objectives**

* Gather requirements
* To identify, investigate and evaluate current systems (Topological navigation editors)
* To identify, investigate and evaluate tools and techniques
* Create and evaluate a prototype
* Implement the topological navigation editor user interface
* Evaluate the topological navigation editor
* Artefact to be delivered on time
  1. **Motivation**

The two primary sources of motivation for this project came from the practical applications of autonomous mobile robots, and the more general insight to be gained in the field of robotics. However the main motivation is the current application of autonomous systems. In this day and age, everything is becoming autonomous, from household goods like hoovers, to factory machines and even automobiles. The figure below helps visualise the growth and expected growth in robotics use in different sectors in terms of market size.

Figure 2: Worldwide Robotics Market Growth



\**Excludes low level electronic toys*

*Source: Japan Robotics Association*

The multi-colored areas of the graph that rise above the blue area depict expected growth in *mobile* robotics. The blue area depicts maturation in the fixed robotics area.

In his article “The Robots Are Here” (Feb 2004 *MIT Technology Review*), Dr. Rodney Brooks commented, “I am convinced that robots are today where computers were in 1978. That’s about the year that computers started to appear around us in the way that robots are cropping up today. Of course, it was another 15 years before computers truly became pervasive in our lives. I think that 15 years from now, robots will be everywhere, as email and the Web are now.”

This project will also add greatly to the field of robotics in general, as it is still a growing field with a lot of research still to be undertaken where it pertains the edition of topological navigators.

1. **Requirements Engineering**

**3.1 Overview**

Understanding what is required in a project is not an easy task; and can often be a source of issues when unsuitable systems are delivered. Somerville (2011) describes requirements as: “statements, in natural languages plus paradigms, of what services the system is expected to provide to systems users and the constraints under which it must operate”. The correct identification of requirements is a fundamental part of achieving project success, as Vliet (2008, p12) explains: “The more careful we are during requirements engineering phase, the greater is the chance that the ultimate system will meet expectations”.

Requirements can be generalised into two categories: functional and none functional. “Functional requirements are those that the piece of software must provide when complete. These are based on the development of the piece of software and define what should be delivered.” (Somerville, 2011, p84 & p85). In this project focus will be mainly on functional requirements, “Non-functional requirements are not concerned with the specific systems deliverables. They may however, address issues such as reliability, security or performance.” (Somerville, 2011, p87). Non-functional requirements like usability are however fundamental, since producing an artefact that is not usable will defeat the purpose of the art.

**3.2 Functional requirements**

The functional user requirements involve the creation of tools that will simplify the topological map edition process making the whole system more user friendly. Simplify the use map already created by other applications such as RVIz and ROS (Robot Operating Systems), simultaneous metric and topological map creation and direct map edition. Improve and complete the tools for editing previously created maps using Rviz and ROS and create a simple dedicated graphical user interface for editing the topological map.

The navigation editor design is to be simple for ease of control of the robots by people of all walks of life, with little to no technical ability. The editor will include features like drag and drop, the user will be able to drag and drop a node, edge and also edit these through their properties tab by clicking on the node or edge itself. The user should also be able to edit maps created by other applications like Rviz and also be able to edit maps directly from text files. This implies that the user be able to load maps for editing. Editing of the maps will include but not limited to adding, deleting, moving, naming nodes and edges. However, while the graphical interface is aimed at user with no technical knowledge, expert users will also benefit.

**3.3 Non-functional requirements**

The non-functional requirements will be divided in terms of usability goals and user experience goals. According to (Preece et al, 2011) usability refers to ensuring that interactive products are easy to learn, effective to use and enjoyable from the users perspective. With this in mind, design will also try to cater for those users with various impairments so they can also have an enjoyable experience using the application

Usability goals provide the interaction designer with a concrete means of assessing various aspects of an interactive product and user experience, and just to reiterate this approach will be very effective hence the decision to implement it. “More specifically, usability can be broken into the following goals:

* Effectiveness
* Efficiency
* Safety
* Utility
* Learnability
* Memorability

A diversity of user experience goals has been articulated in interaction design, which covers a range of emotions and felt experiences. These include desirable and undesirable aspects:

Desirables:

* Cognitively stimulating
* Enjoyable
* Exciting
* Helpful
* Motivating
* Satisfying ” (Preece et al, 2011)

There are a few constraints that will need to be taken into consideration when designing the application and these includes things like the cost, technical faults, convenience etc. The main point in this section is the gathering of requirements so as to establish the basis of the design.

The application will need to be able to perform the intended task such as editing nodes, editing edges and drag and drop edges and nodes that are simple and intuitive, but the client from the requirements gathering will decide what and how this information should be presented. Building upon previous research on how autonomous mobile robotics technology is employed in different facets of life and how the graphical user interfaces are designed, great ideas of what functions to include and how to design a user centred design interface aided the manifestation of this artefact.

Using eXtreme Programming (XP) methodology, requirements are gathered through user stories, which are then developed into use cases. Below are the use case diagrams that depict the user stories from the requirements gathering exercise.

**3.4 Use case diagrams**

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). The advantages of use cases are that each use case will provide some observable and valuable result to the users or stakeholders of the system. These use cases will characterise the software’s system interaction with external actors and these interactions are determined from the functional user requirements outlined in the above section.

1. User editing nodes and edges
2. User editing maps from other sources
3. User to load maps from external DBMS

Edit nodes - add, delete, drag and drop, name, colour, size

* Edit edges - add, delete, drag and drop, name, position, colour, size
* Edit Maps - load, drag and drop nodes and edges, name, save.
* Auto-saving
* Feedback
* Robot position

The below diagram show the main use case that are required for the application to accomplish.



Figure 3: Main Use Case Diagram

1. **Background research**
   1. **Overview**

The background research is fundamental in undertaking this project as it helps identify inherent problems in graphical user interface development through case studies. It provides a good overview of the problem domain. A robot doesn’t have complete and accurate knowledge about obstacles; this is where map editing comes into play. It help the robot to have a predefined route to follow, it eliminates the need for a robot to build a new map through scanning.

* 1. **Maps**

“A map data representation of environments of a mobile robot performing a navigation task is specified in this standard. It provides data models and data formats for two-dimensional (2D) metric and topological maps.” (IEEE Std 1873, 2015). Maps are useful for a wide variety in robotics this includes activities like localisation, planning, mobile manipulation and human-robot interaction.

Robotic mapping research has a long history. In the 1980s and early 1990s, the field of mapping was widely divided into metric and topological approaches. Metric maps capture the geometric properties of the environment, whereas topological maps describe the connectivity of different places. An early representative of the former approach was Elfes and Moravec’s important *occupancy grid mapping algorithm*, which represents maps by fine-grained grids that model the occupied and free space of the environment. Topological maps represent environments as a list of significant places that are connected via arcs. Arcs are usually annotated with information on how to navigate from one place to another. However, Sebastian Thrun (2002) claims the distinction between metric and topological has always been fuzzy, since virtually all working topological approaches rely on geometric information. In practice, metric maps are finer grained than topological ones

**4.3 Topological map**

A topological map represents an environment in the form of a graph consisting of a set of nodes (or vertices) and edges connecting the nodes (Choi *et al*, 2011). The edges could represent actions needed to get from one node to the next, or direction and distance. An example from robotics is a map of a building interior, where nodes in the map denote landmarks or distinctive features of the corresponding places, and edges represent possible routes for a mobile robot to navigate from one room to another. Topological maps provide a useful abstraction for robotic navigation and planning. An everyday example of a topological map is the one commonly used for rail or bus networks, where primary information consists of nodes in the map (stations) and the connectivity between them (which rail line or bus line goes between which stations).

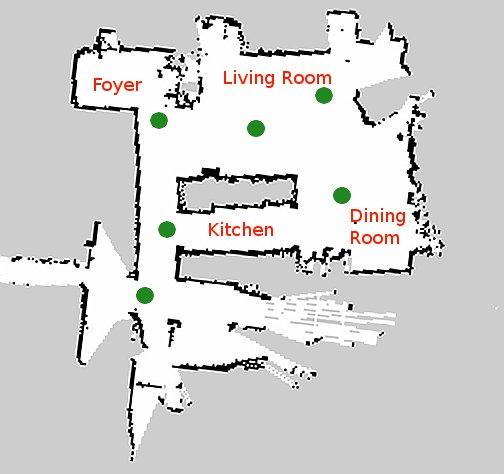


Figure 4 topological map

**4.4 Metric map**

“Metric maps are perhaps the most popular type of maps used in everyday activities; for example, city maps. A metric map explicitly encodes the physical layout of a target space and positions of physical objects therein.” (IEEE Std 1873, 2015). “The features used for metric environmental representation are infinite lines. They are less informative, but have a better probabilistic model with analytical solution and permit a very compact representation of structured geometric environments (i.e. long hallway represented by only two infinite lines)” (Nicola Tomatis *et al*, 2001)

c) Hybrid map

**Graphical User Interface**

“In computer science, a **graphical user interface** or **GUI**, is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs),which require commands to be typed on the keyboard.” (wikipedia).

Steven Levy, Jr. 2015 defines a GUI as “a computer program that enables a person to communicate with a computer through the use of symbols, visual metaphors, and pointing devices.”

“The specific techniques of graphical user interfaces that were first shown to have tremendous commercial impact by the Macintosh (starting in 1984) have been widely adopted with only small variations, and a relatively slow growth of new techniques.” B. Myers *et al* (2000). There are many tools now available on the market used to develop a GUI. PyQt4 is the tool to be used on the project. It is optimized to work well with the Python programming language, which is the language of choice to be used.

A robot, just like a computer system needs a user interface to communicate with users. Graphical user interfaces have become very popular due to their ease of use. WIMP represents the most common graphical elements: window, icon, menu, and pointer. A window is an area on the screen that displays information. The contents of the window are displayed independently from the rest of the screen. “Designing a good user interface is critical to the success of a system. A good user interface encourages an easy and natural interaction between a user and a system. Ideally, a user can forget that they are using a computer and get on with what they want to do.” (Paul Zandbergen, 2003). This is particularly important for this project as it aims to develop a user interface for users with no technical expertise.

“Most GUI programs, once invoked they run their event loop and respond to events. Some events come from the user, for example key presses and mouse clicks, and some from the system, for example timers timing out and windows being revealed. They process in response to requests that are the result of events such as button clicks and menu selections, and only terminate when told to do so.” (Mark Summerfield, 2007)

1. **Literature review**
   1. **Overview**

In this section presented is the literature review undertaken in order to understand the problem domain and allow to design the system to meet the user requirements and critically evaluate already available materials. The point of literature review as identified by Borg and Gall (1989, cited by Saunders et al. 1997: 39) is among other things:

* To refine research questions and objectives
* To highlight research possibilities that have either been explicitly  identified by other authors or have possibly been overlooked in the past.
* To avoid repeating the works of others
* To identify search methods and strategies  This is achieved with references to past and current literature in the field of robotics, user interfaces and Human computer interaction.
  1. **Maps**

There has been a great deal of work on learning maps for mobile robotics. Topological mapping is a well-explored area in the mobile robotics community. A topological map provides: first, a useful representation of the environment that allows robot navigation without necessarily requiring the maintenance of the robot’s pose in a global reference frame and, second, an abstraction provided by the topology information that can aid higher level planning and inference tasks. Early work in this area by Kuipers and Byun, (1991) constructed a topological network description of the environment by identifying, and then linking,

Sometimes it is necessary for a robot to know its location accurately in terms of metric coordinates; in such cases, metric maps are clearly the best choice. In many other environments, such as office buildings with corridors and rooms, or networks of roads, maps that simply specify the topology of important locations and their connections suffice. Such maps are typically less complex and support much more efficient planning than metric maps. Topological maps are built on lower-level abstractions that allow the robot to move along arcs (perhaps by wall- or road-following) and to recognize properties of the locations; they are flexible in allowing a more general notion of state.

As of to date, Metric and Topological are the two major existing paradigms for mobile robot mapping. Approaches in the metric paradigm generate fine grained, metric description of a robots environment. Occupancy grids (Borenstein and Koren, 1991; Elfes, 1989; Moravec, 1988) are probably the most successful example of metric approaches to mobile robot mapping. Other approaches, such as those described in (Chatila and Laumond 1985; Cox 1994; Lu and Milios 1997), describe the geometric atoms such as straight lines (walls) or points (range scan). Approaches in the topological paradigm, on the other hand, generate coarse, graph like descriptions of environments, where nodes correspond to significant, easy-to-distinguish places or landmarks, and arcs corresponding to actions or action sequences that connect the neighbouring places. Examples of topological approaches can be found in (Chown, Kaplan and Kortenkamp 1995; Kortenkamp and Weymouth 1994; Kuipers and Byun 1991; Mataric’ 1990; Shatkay and Kaelbling 1997).

It has been long recognised (Chatila and Laumond 1985; kuipers and Byun 1991; Thrun 1998; Thrun and Bucken 1996) that either paradigm alone, metric or topological, has significant drawbacks. In principle, topological maps should scale better than metric maps to large scale environments, because a coarse grained, graph structured representation is much more compact than a dense array, and more directly suited to problem-solving algorithms (Kuipers and Byun 1991). However, purely topological maps have difficulty distinguishing adequately among different places (Kuipers and Byun 1991; Dudek et al 1991) and have not, in practise, been applied successfully to large environments, but memory and time complexity pose serious problems.

The idea of integrating topological and metric representation was proposed by various researchers (Chatila and Laumond 1985; Kuipers and Levitt 1998). Thrun and Buecken (1996) achieved a successful integration, following the “first metric then topological” paradigm proposed in (Chatila and Laumond 1985). Their method was able to produce metric maps of unprecedented size and precision, which were then analysed to extract a topological map.

* 1. **Graphical User Interface**

Well-designed graphical user interfaces can free the user from learning complex command languages, which is the main aim of the project developing this topological map editor. Graphical user interfaces (GUIs) simplify use of computers by presenting information in a manner that allows rapid assimilation and manipulation. The use of visual constructs (widgets) that mimic physical objects such as `buttons' can speed learning, by providing an intuitive method to provide input to the computer. However, Brian Toby (2001) goes on to say “a GUI is not always an improvement. If the GUI is organized in a counterintuitive manner, or if the menu contents are arranged haphazardly, or if commonly performed operations require several unexpected steps to be performed, then a user must typically invest a significant amount of time in learning how to use the program before the program can be used effectively.”

A good GUI design does not require users to memorize the steps needed to perform an action. This is particularly important for scientific applications, where the goal of the user should be to understand the theory behind a program rather than master the arcane steps needed to perform an action. There is one case where memorization may be unavoidable. It is appropriate that a GUI incorporate shortcuts that simplify multi-step tasks; use of these shortcuts may not always be intuitive and expert users may choose to commit them to memory to speed their work. There should also be an obvious way to perform the same tasks, albeit less efficiently, without use of the shortcut.

“The combination of Python and Qt, "PyQt", makes it possible to develop applications on any supported platform and run them unchanged on all the supported platforms, for example, all modern versions of Windows, Linux, Mac OS X, and most Unix-based systems” (Mark Summerfield, 2007)

* 1. **PyQt4**

“PyQt is based on Qt, an advanced GUI library for Windows and Unix written in C++ by Eirik Eng and Arnt Gulbrantsen of Trolltech in Norway. It's quite easy to wrap C++ or C libraries so they can be used from Python — and when Phil Thompson was looking around for a good GUI library for Python he decided to wrap Qt, producing PyQt.”

The combination of Python and Qt is extremely powerful, and is used in a wide variety of applications. People are scripting OpenGL applications with it, creating complex 3D models, animation applications, writing database applications, games, utilities and hardware monitoring applications. It is used in open source projects, but also, by large companies like Disney Television and Media”, BoudewijnRempt (2001), continues to say “If you're not working on embedded software, hardware drivers or a new operating system, chances are that PyQt is the right choice for you, too.”

The good thing about PyQt widgets is that they can be drawn in *styles*, to make them appear exactly like the native widgets of the operating system the application runs on (or like something different altogether, if you want).

1. **Design** 
   1. **Overview**

The aim and purpose of the design stage is to facilitate the determination of a clear plan for the development stage. User interface design requires properly identifying user needs and expectations. The ISO 9241-11 *Ergonomics of Human System Interaction* provides guidance on usability that is defined as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO 9241-11, 1998). Considering these three criteria, a prototype of the user interface will be made first and submitted to the client for evaluation. The feedback will then be used to design the ultimate interface design. “User interface design is an iterative process where users interact with designers and interface prototypes to decide on features, organisation and look and feel of the system user interface.” (Sommerville, 2007).

“In order to design a system, it is necessary first to understand what it is that the system should be doing. The process of understanding the job to be done by the system is provided by task analysis”(Christine Faulkner 1998). In order to effectively design an effective system, it is fundamental to understand the requirements (user and system). (Preece et al, 1994) goes on to say “understanding requirements involves looking at similar products, discussing the needs of the people who will use the product, and analysing any existing system to discover problems with current designs.”

The ultimate objective of the design process is to present the proposed system so that the resulting models serve as effective implementation blueprints. The design is divided in two areas that are: Product design and Class design discussed below after prototyping section.

* 1. **Low-Fidelity Prototype**

“A Low-fidelity prototype also known as Lo-fi is one that does not look very much like the final product, it uses materials that are very different from the intended final version, such as paper and cardboard rather than electronic screens and metal.” (Preece et al, 2011). Paper prototyping involves use of paper, index cards, cardboard etc. to build up a paper representation of the system. One example of this was the “cardboard computer” (Ehn and Kyng, 1991).

Lo-fi prototyping is used for the benefits it offers such as being cheap and quick to produce. Lo-fi prototypes are also simple, cheap and quick to modify hence they support the exploration of alternative designs and ideas and this is the greatest strength of applying this kind of prototyping for this project. This is particularly important during the user interface conceptual design.

* + 1. **First Iteration**

After conducting the requirements engineering exercise, the first design prototype was produced. In accordance with the extreme programming development methodology used for the project, however not rigid, a review with the client was undertaken to make sure the ideas, features and layout of the interface meet the client and user requirements. As expected the first iteration contained many unwanted features, however the main features were all included as shown by the figure below:

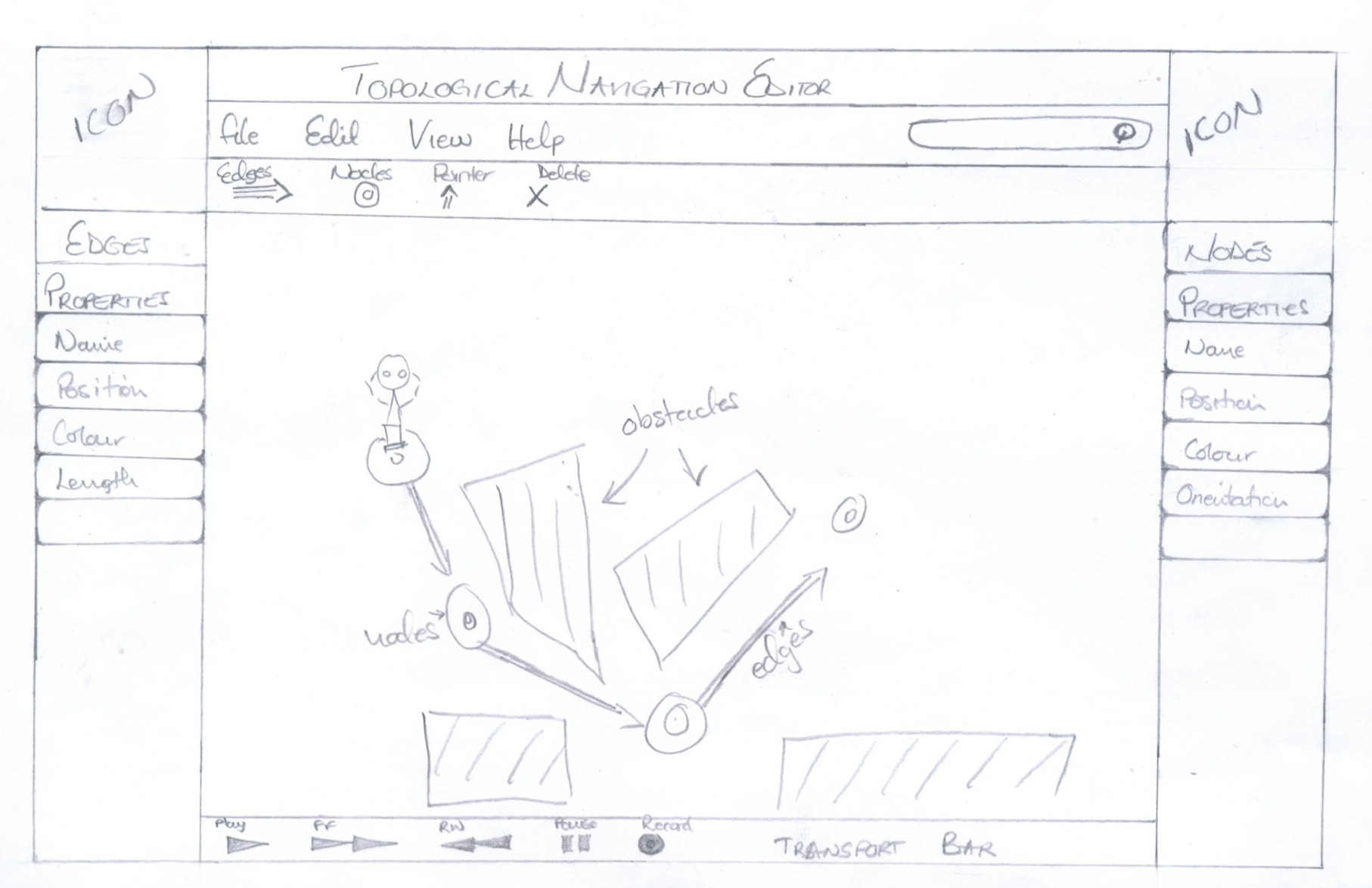


Figure 5: Editor interface prototype

* + 1. **Second iteration**

After a review of the first prototype, a few changes had to be made and this called for a second iteration. In this iteration most of the widgets have been removed as per customer advice and user requirements however, most of the functionality and design was maintained. This removal meant that the user interface would have few widgets reducing unnecessary clatter. The figure below shows the changes made to the first iteration prototype.

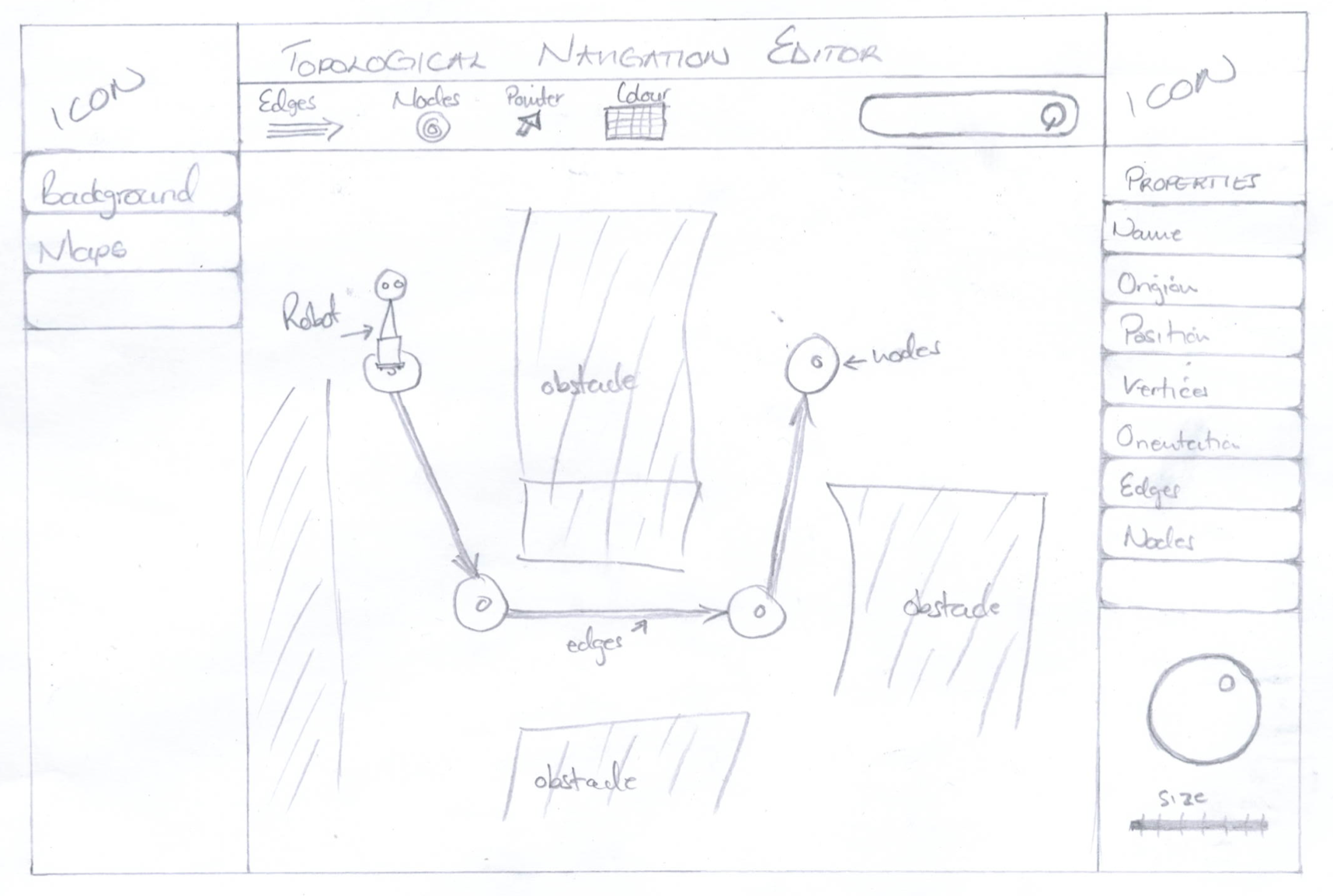


Figure 6: Revised interface design

* 1. **Product design**

The goal of product design, the topic of this section, is to create effective interactions that are structured and arranged effective and also efficiently. This stage will also aid reduce the amount of work required in the next stage of the product development.

**Design structure diagram**



* + 1. **Class design**

The goal of class design, which is the topic of this section, is “to define the class comprising the system along with the interrelationships among those classes.



* 1. **Design evaluation**

Design process needs to take into account basic GUI functionalities like opening and saving files, designing menus and widgets as well as HCI issues including cognitive load and cost of action for users. The GUI should provide capability of editing maps; this capability is gained by implementing add node and edge, drag and drop, mirroring facilities. There are several reasons for modelling software before embarking on writing code. The main reason is that the models allow us to evaluate the quality of the proposed solution without too large an investment of effort and resources. The final evaluation of the artefact prototype was carried out after changes were made in the second iteration. The result was the final design of the artefact to be implemented.

1. **Implementation** 
   1. **Overview**

The literature review presented in the chapter above perhaps goes some way to highlight the complex nature of the project domain and its related issues. It also identified existing systems and approaches for graphical user interface development and topological map editing. The domain could be seen to present some rather unique and significant challenges in terms of design and implementation. The following section details the methodologies adopted in the implementation of the artefact during the project.

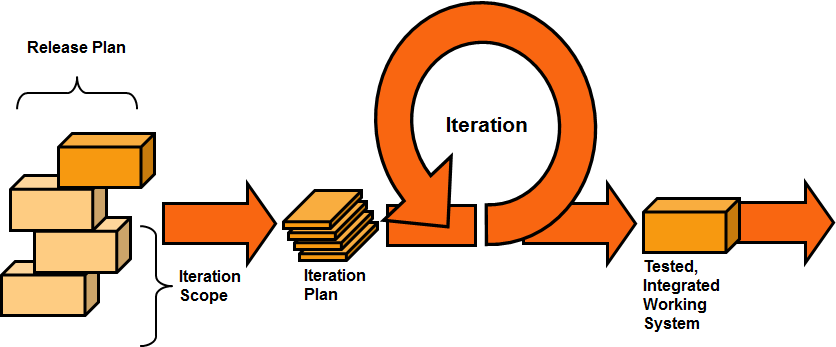
The language of choice for developing the artifact is python as this is a requirement for systems compatibility purposes. “Python is a simple but powerful object-orientated language. Its simplicity makes it easy to learn, but its power means that large and complex applications can be created. Its interpreted nature means that Python programmers are very productive because there is no edit/compile/link/run development cycle.” (The Qt Company Ltd, 2015).

Also as a requirement, PyQt4 is the tool used to develop the graphic user interface. PyQt is a set of Python v2 and v3 bindings for the cross-platform Qt application framework developed in C++ for designing graphical user interfaces, which runs on all platforms supported by Qt. PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python. (The Qt Company Ltd, 2015).

* 1. **Methodology**

With the complexity of the project domain and the nature of the software artefact, flexibility and adaptability could be considered the most significant requirement of the development methodology. With this in mind, a suitable model to apply will be that which offers great flexibility and adaptability to allow for dynamic changes during implementation stage. Considering all the models that can be applied to a project, an agile methodology framework will suit the project, as it will allow for the more important changes during the implementation phase of the cycle. There are however a few agile frameworks like FDD, Crystal offering different features. A look at these models left one favourable, Extreme programming.

The implementation of the interface is carried out using an agile software development methodology known as Extreme Programming (XP). XP is a software engineering methodology, the most prominent of several agile software development methodologies. Like other agile methodologies, Extreme Programming differs from traditional methodologies primarily in placing a higher value on adaptability than on predictability. “XP also focuses on excellent application of programming techniques, clear communication, feedback, courage and respect, which allows us to accomplish things we previously could not even imagine”(Kent Beck and Cynthia Andres, 2005). Below is a figure illustrating XP Programming:



“XP builds on best practices such as unit testing, pair programming, and refactoring” (Erich Gamma, 1999). However, since this is an individual project, pair programming will be practiced but to a lesser extent, as this requires two people to undertake. “Extreme programming uses an object oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities: planning, design, coding and testing.” (Roger S. Pressman, 2010). The reasons for using this framework compared to other will be discussed in the depth in the Evaluation section of this document.

* + 1. **First Iteration**

Main window

The first iteration was just the implementation of the main window with a few widgets for an actual visualisation of how the application will look. In PyQt, any widget can be used as a top-level window, even a button or a label. When a widget is used like this, PyQt automatically gives it a title bar. The code snippet shows how the main window class was created. This class will contain other objects such as the property window for example. It will also help arrange the widgets within it using the inbuilt layout style as shown below.

class MainWindow(QtGui.QMainWindow):

InsertTextButton = 10

# default init function sets up the window properties

def \_\_init\_\_(self):

super(MainWindow, self).\_\_init\_\_()

# main window settings

self.widget = QtGui.QWidget()

self.widget.setLayout(mainlayout)

self.setCentralWidget(self.widget)

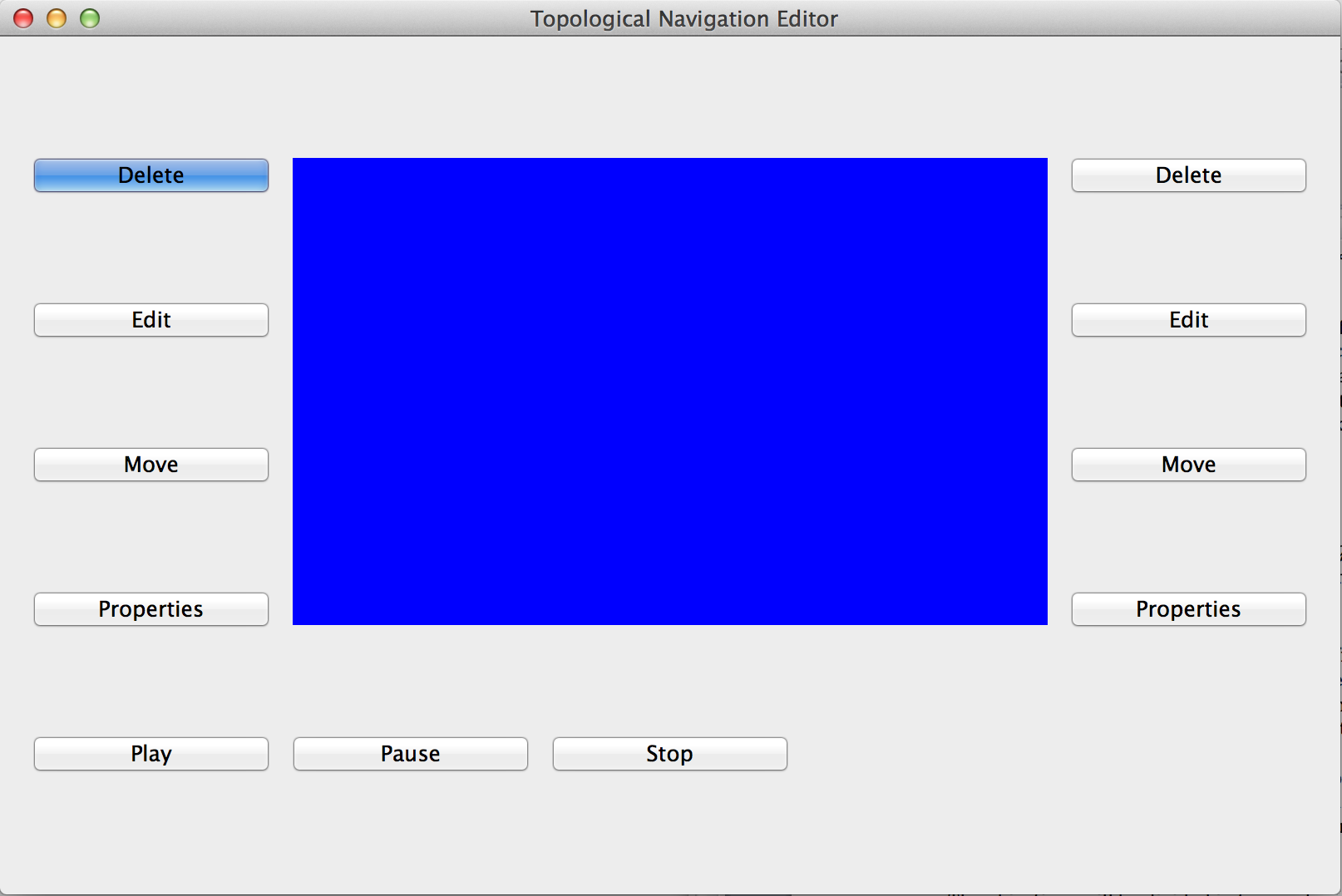
self.setWindowTitle("Topological Navigation Editor")

mainWindow = MainWindow()

mainWindow.setGeometry(100, 100, 1000, 600)

mainWindow.show()

The mainWindow class will represent the topological navigation editor’s user interface. At this stage only button widgets are used as an illustration for the layout as shown by the figure below:



A grid layout has been used to position the toolbox, scene and the property window. This was achieved by adding widgets in each layout. We are representing the topological navigation editor as a Python class because we want to maintain the state of the interface. Here, we maintain the state of the interface e.g. every time there is user input, as well as check to see the latest interface state.

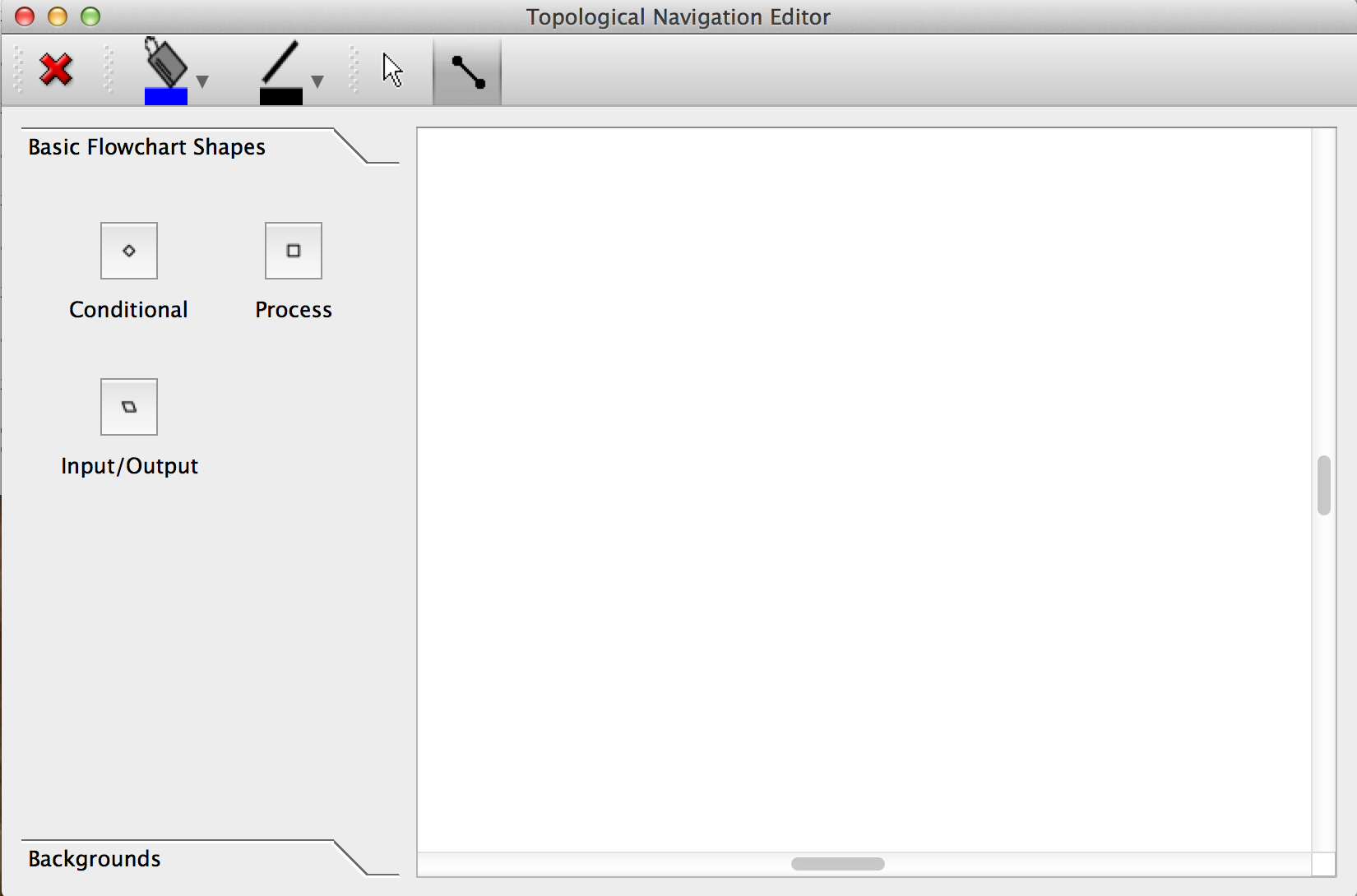
* + - 1. **Evaluation (1st iteration)**

The first iteration went as expected, as seen in the figure above, the button widgets are used only to show the layout of the main window. The evaluation at this stage was done with the client. It was discussed that the toolbox will occupy the left side of the scene and the property window on the right. The other topic of discussion was centred the menu bar. However, not yet implemented, the menu bar was deemed not necessary so it will not implemented. For the ability to save the changes made to the maps, a mouse right click would bring up the options for saving files along with some editing options like copy, paste just to mention a few.

* + 1. **Second Iteration**

Toolbar and toolbox

The second phase involved the implementation of the toolbar and toolbox. The toolbox includes the nodes, background and map loading buttons. The toolbox will save as a container for widgets that the user will interact with. The node tab in the toolbox will contain node shapes, while the map and background tabs will contain buttons for loading and changing maps and backgrounds respectively as shown by the figure below:



The toolbar was also implemented at this stage, as shown on the main window below the window title in the above figure.

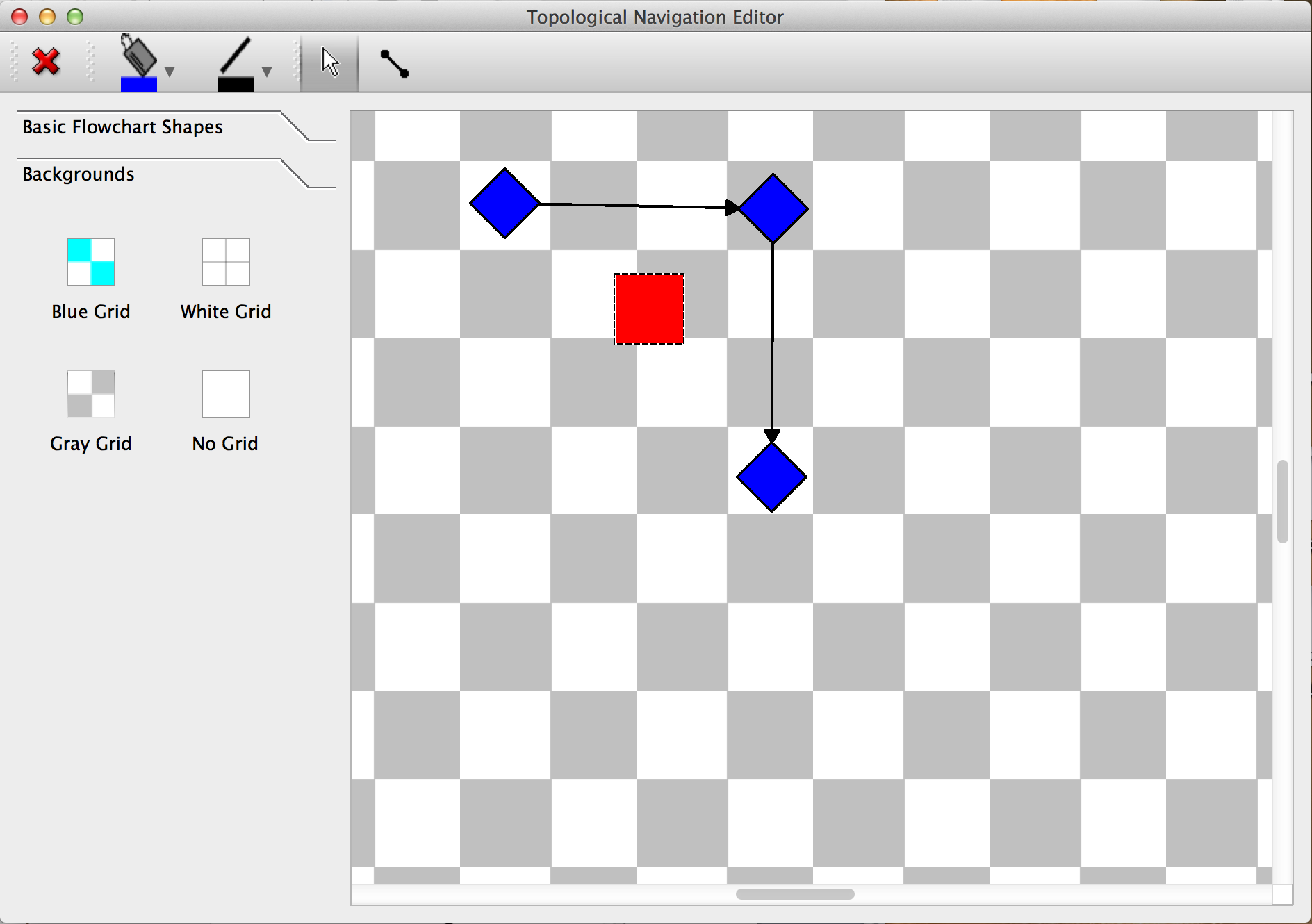
* + - 1. **Evaluation**

In these iteration evaluations, only the client took part.

* + 1. **Third Iteration**

Nodes, Edges and backgrounds

At this stage the artefact has taken form and shape



**Fourth Iteration**

Property windows



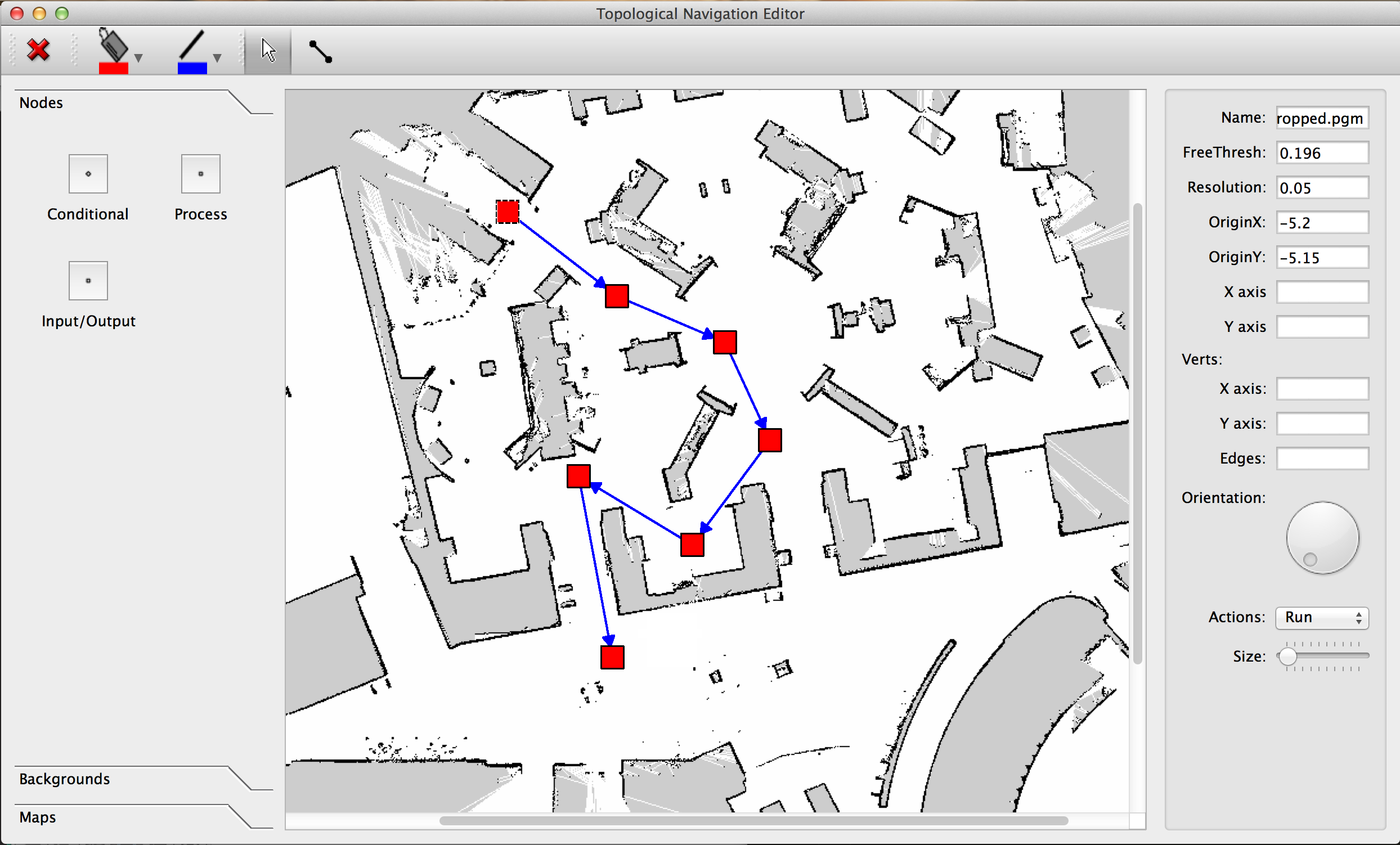
**Fifth Iteration**

Loading maps and map display



**Sixth Iteration**

Updating and Saving



1. **Evaluation** 
   1. **Overview**

While it rings true that every software artefact in a project in the field of computer science is developed for a reason and a specific purpose, the need to make sure that the artefact meets requirements is paramount. Evaluation of the artefact will be conducted using a focus group and the client.

Surveys, Questionnaires and interviews are common methods for gathering information useful for evaluating programs. Recently, however, a group method for gathering such information has become increasingly popular. This method is the focus group interview. In a paper, *Using Focus Groups in Program Development and Evaluation* published by Roger A. Rennekamp *et al,* they say “focus groups are a special type of group used to gather information from members of a clearly defined target audience. Rather, a focus group is...

* Composed of six to twelve people,
* Who are similar in one or more ways, and
* Are guided through a facilitated discussion,
* On a clearly defined topic,
* To gather information about the opinions of the group members.

The goal of a focus group is to promote self-disclosure among participants. Because a group, rather than an individual, is asked to respond to questions, dialogue tends to take on a life of its own. Participants “piggy-back” on the comments of others and add a richness to the dialogue that could not be achieved through a one-on-one interview. The focus group used in the evaluation comprised of people with the following attributes:

* Students
* Age range 18 - 25
* Studying level 3 Autonomous mobile robotics
* 1. **Systems Evaluation**
  2. **Methodology Evaluation**
     1. **Overview**

This section will discuss in depth the reason for using the extreme programming framework among all the other agile models and some traditional models. A methodology is a system of broad principles or rules from which specific methods or procedures may be derived to interpret or solve different problems within the scope of a particular discipline. However during the course of the project, some practises where borrowed from other models.

**Evaluation**

Project duration is a factor used for choosing a methodology. Heavyweight methodologies involve a lot of “time waste” outputs such as documentation, design documents, writing analysis etc. Concluding that when time is limited, using an agile methodology would be better. This is the one of the two main reasons I believe Extreme programming was the best methodology to use for the project of this size within the time constraints. The other reason for selecting this model was due to the fact that it offers flexibility and adaptability, which were the fundamental requirements of the model as aforementioned in the above sections. This is type of agile software development involves developing and delivering functionality in fragments or in incremental stages. The XP methodology allows for iterations and changes are encouraged and are seen as natural and desirable happenings that occur during the course of the projects development cycle. This approach is beneficial and appropriate for this particular project where the exact requirements and potential difficulties are not always foreseeable due to the experimental and dynamic nature of the project. The methodology involves splitting up the different aspects of the projects functionality steps giving the developer a greater degree of control over the projects development process.

“XP builds on best practises such as unit testing, pair programming, and refactoring” (Erich Gamma, 1999). However, since this is an individual project, pair programming will not be practised, as this requires two people to undertake. “Extreme programming uses an object oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities: planning, design, coding and testing.” (Roger S. Pressman, 2010). The extreme model approach carries far less risk than most traditional methodologies like Waterfall including some agile models approaches. The focus is on delivering fully tested, independent, valuable, small features. As such, risks are diversified i.e. if one feature goes wrong, it should not impact another feature.

Extensive communication is also another core principle of agile development methodologies, however the level of communication differs in different approaches. In Scrum and Extreme Programming (XP) informal daily meetings happen which keeps all team members up-to-date with the process, while in Feature Driven Development (FDD) and Dynamic Systems Development Methodology (DSDM) information is shared through documents which puts a lot of work on the developers to write the documentation, and in Crystal and ASD face to face meetings take place. While extreme programming puts more emphasis on programmer productivity than the documentation, this leaves it vulnerable to easy abuse and poorly documented as compared to other approaches like FDD and DSDM.

One of the disadvantages of Extreme programming is the way it allows for changes. This in one hand is an advantage, while on the other hand is a disadvantage since changing requirements allow clients to get tempted to keep demanding more functionalities. The agility of this methodology makes it suitable for a project of any size, however it is recommended that this model be applied to small projects due to its other limitations.

The language used is Python, which complements this methodology and because the advantages it offers. Python is a general-purpose, dynamic, object-oriented programming language. The design purpose of the Python language emphasizes programmer productivity and code readability.

1. **Resources /Toolset Analysis**
   1. **Strands Project**
   2. **Ubuntu 14.04.3LTS (Trusty)**
   3. **Ros Indigo**

“The goal of ROS is *not* to be a framework with the most features. Instead, the primary goal of ROS is to support code *reuse* in robotics research and development. ROS is a distributed framework of processes (aka *Nodes*) that enables executables to be individually designed and loosely coupled at runtime.” (ROS.org)

* 1. **PyQt4**

This section is specifically about PyQt4, the Python bindings for the Qt 4 C++ application development framework. “PyQt4 is provided in the form of 10 Python modules which between them contain around 400 classes and about 6 000 methods and functions. PyQt is the python binding for Qt Library. With the help of PyQt, we can develop GUI applications. PyQt4 contains various modules and classes. Some of the important modules used for this project are

• QtCore  
– Contains details about non-gui functionalities

* e.g. Time, strings, directories, threads etc.

• QtGui

– Details about graphical components  
– e.g. button, windows, tool bar, dialog box etc.

The GUI toolkit provides the graphical user interface (GUI) elements, or widgets, like buttons, scrollbars, text boxes, label fields, tree view lists, file-open dialogues and so on. These widgets are an integral and essential part of the user interface.

* 1. **Python**

Python is probably the easiest to learn and nicest scripting language in widespread use, and Qt is probably the best library for developing GUI applications. No compilation is required thanks to Python being interpreted, and no source code changes to adapt to different operating systems are required thanks to Qt abstracting away the platform- specific details.

Python is a very expressive language, which means that we can usually write far fewer lines of Python code than would be required for an equivalent application written in, say, C++ or Java. This makes PyQt an ideal tool for rapidly and easily developing GUI applications, whether for prototyping or for production use.

* 1. **PyMongo**
  2. **MongoDB**

MongoDB is a “scalable, high performance, open source NoSQL Database” (MongoDB.org). NoSQL refers to a movement that began in 2009, “which is best described as Not Only SQL” (nosql-database.org). According to Harrison (2010), NoSQL databases are highly scalable, can store a large amount of data types, are easier to maintain, cheaper and more flexible than regular Relational Database Management Systems (RDBMS’s).

“In MongoDB you store JSON-like documents with dynamic schemas. The goal of MongoDB is to bridge the gap between key-value stores (which are fast and scalable) and relational databases (which have rich functionality). This new data model simplifies coding significantly, and also improves performance by grouping relevant data together internally.” (MongoDB.org).

1. **Conclusion**
   1. **Overview**

Generally, research and innovation in tools trail innovation in user interface design, since it only makes sense to develop tools when you know for what kinds of interfaces you are building tools. Given the consolidation of the user interface on the desktop metaphor in the last 15 years, it is not surprising that tools have matured to the point where commercial tools have fairly successfully covered the important aspects of user interface construction. It is clear that the research on user interface software tools has had enormous impact on the process of software development. However, I believe that user interface design is poised for a radical change in the near future, primarily brought on by the rise of ubiquitous computing, recognition-based user interfaces, 3D, and other technologies. Therefore, I expect to see a resurgence of interest and research on user interface software tools in order to support the new user interface styles.

* 1. **Objective Analysis (all)**
  2. **Critical Reflection**
  3. **Further Development**

Whereas most of today’s tools provide good support for widgets such as menus and dialog boxes, which use a keyboard and a mouse, these will be a much smaller proportion of the interfaces of the future. We expect to see substantially more use of techniques such as gestures, handwriting, and speech input and output. These are called *recognition-based* because they require software to interpret the input stream from the user to identify the content.

**Reference:**

ISO 9241-11. (1998) *Ergonomics of Human System Interaction - Part 11: Usability: Definitions and concepts*.

Kent Beck and Cynthia Andres (2005) *Extreme Programming Explained, Embrace Change.* Pearson Education, Inc.

Jenny Preece, Yvonne Rogers, Helen Sharp (2011) *Interaction Design beyond human-computer interaction 3rd Edition*. Wiley.

SOMMERVILLE, I. 2007. *Software Engineering*. Eight Edition. England, Pearson Education Limited.

Sommerville, Ian. 2011. *Software Engineering*. Ninth Edition. England, Pearson Education Limited.

SUMMERFIELD, M. 2007. *Rapid GUI Programming with Python and Qt*. Michigan, Edwards Brothers.

Boudewijn Rempt. 2001. *GUI Programming with Python: QT Edition.* Commandprompt, Inc.

Vliet, Hans Van. 2008. Software Engineering: Principles and Practices. Third Edition. John Wiley & Sons Ltd

Ben Shneiderman and Catherine Plaisant, (2005) *Designing the user interface, Strategies for effective human-computer interaction 4th Edition.* Pearson Education Inc. USA

Julie A. Jacko, Andrew Sears (2003) *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications.* Lawrence Erlbaum Associates. London.

Roger S. Pressman, (2010) *Software Engineering: A Practitioner’s Approach Seventh Edition.* McGraw-Hill.

G. Lindgaard (1994) *Usability Testing And System Evaluation: A guide for designing useful computer systems.* Chapman & Hall. London

D, Stone., C. Jarret., M. Woodroffe., S. Minocha (2005). *User Interface and Evaluation*. Morgan Kaufmann Publishers.

Brad Myers, Scott E. Hudson, and Randy Pausch. (2000) *Past, Present, and Future of User Interface Software Tools*. Carnegie Mellon University

Cliff, D., Husbands, P. and Harvey, I. (1993) *Explorations in evolutionary robotics, Adaptive Behavior* 2(1), 73–110.

Ehn, P. and Kyng, M (1991) *Cardboard Computers: Mocking it up or hands-on the future*. New jersey.

Map ref

IEEE Robotics and Automation Society (2015). *IEEE Standard for Robot Map Data Representation for Navigation* (IEEE Std 1873TM-2015)

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7300355> [Accessed on 16 Mar.16]

A. Elfes. 1987. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3 (3):249–265.

A. Elfes. 1989. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. Department of Electrical and Computer Engineering, Carnegie Mellon University.

Sebastian Thrun (2002) Robotic Mapping: A Survey. CMU-CS-02-111

Carnegie Mellon University Pittsburgh, PA 15213

Dudek, G., Jenkin, M., Milios, E., and Wilkes, D. 1991. *Robotic exploration as graph construction*. IEEE Transactions on robotics and Automation 7 (6): 859 – 865.

Choi, Jinwoo, et al (2011). *Autonomous topological modelling of a home environment and topological localization using a sonar grid map*,” *Autonomous Robots,* 30(4): 351–368.

Kuipers, B and Y. T. Byun, (1991) *A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations*, *Journal of Robot and Autonomous Syst.*, vol. 8, pp. 47–63.

Thrun, S. and Bucken, A. 1996. Integrating grid-based and topological maps and mobile robot navigation. In proceedings of the Thirteenth National Conference oon Artificial Intelligence. Menlo Park.

Thrun, S. 1998. *Learning maps for indoor mobile robot navigation*. Artificial Intelligence.

Chatila, R and Laumond, JP. 1985. *Position referencing and consistent world modelling for mobile robotics*. In Proceedings of the 1985 IEEE International Conference on Robotics and Automation.

Borenstein, J. and Koren, Y. 1991. *The vector field histogram – fast obstacle avoidance for mobile robots*. IEEE Journal of Robotics and Automation 7 (3): 278 – 288.

Kortenkemp, D. and Weymouth, T. 1994. *Topological mapping for mobile robots using a combination of sonar and vision sensing*. In Proceedings of the Twelfth national Conference on Artificial Intelligence, 979 – 894.

Nicola T, Illah N and Roland Siegwart (2001). *Simultaneous Localization and Map Building: A Global Topological Model with Local Metric Maps*

Mataric’ M.J. 1990. *A distributed model for mobile robot environment-learning and navigation*. MIT

Moravec, H.P, 1998. *Sensor fusion in certainty grids for mobile robots.* AI Magazine 61 – 74

Mark Summerfield (2007). Rapid GUI Programming with Python and Qt: *The Definitive Guide to PyQt Programming.* Publisher: Prentice Hall

<https://www.cs.washington.edu/research/projects/urbansim/books/pyqt-book.pdf> [Accessed on 14 Mar.15]

Roger A. Rennekamp and Martha A. Nall. *Using Focus Groups  
in Program Development and Evaluation.* University of Kentucky

<http://www2.ca.uky.edu/agpsd/Focus.pdf> [Accessed on 09 Mar.16]

William B, Fox. (2005) *Entrepreneur, About an industry virtually guaranteed to become a Next New Thing.*

<http://www.amfir.com/AmFir/Commentary%20Pages/EditorialsArticles/I,%20Robot%20Investor/I,%20RobotEntrepPart-1-FutureGrowthTrend.html> [Accessed on 16 Jan. 16]

The Qt Company Ltd, (2015). *What is PyQt?*

<https://riverbankcomputing.com/software/pyqt/intro> [Accessed on 11 Jan. 15]

<https://books.google.co.uk/books?id=bt3oAgAAQBAJ&pg=PT228&lpg=PT228&dq=edges+in+pyqt+python&source=bl&ots=gYgs3TFZmw&sig=yZ0l8gRKPB80veQtItVLGho9gpA&hl=en&sa=X&ved=0ahUKEwj5t6796tfJAhWIVhoKHfS4AAM4ChDoAQgpMAI#v=onepage&q=edges%20in%20pyqt%20python&f=false> [Accessed on 13 Dec. 15]

Elastic nodes (custom made)

<http://ftp.ics.uci.edu/pub/centos0/ics-custom-build/BUILD/PyQt-x11-gpl-4.7.2/examples/graphicsview/elasticnodes.py> [Accessed on 12 Dec. 15]

Elastic nodes

<https://github.com/Werkov/PyQt4/blob/master/examples/graphicsview/elasticnodes.py> [Accessed on 12 Dec. 15]

Widgets

Jan Bodnar

<http://zetcode.com/gui/pyqt4/widgets/> [Accessed on 02 Dec. 15]

<http://stackoverflow.com/questions/18428095/qt4-qmenu-addaction-connect-function-with-arguments> [Accessed on 11 Jan. 16]