# A Menu Interface Development Environment Based on Lean Cuisine

A thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Computer Science at Massey University

**Jianpeng Gu**

**1995**

# Abstract

The integrated user interface development environment based on the Lean Cuisine graphical notation [Apperley & Spence, 89] is a combination of software tools used to support user interface development from initial design, rapid prototyping through to direct implementation. This thesis describes the development of three software tools used in the integrated user interface development environment.

The Lean Cuisine graphical editor (Elc) provides an interactive design environment for graphical specifications of menu-based interfaces and shows that the Lean Cuisine notation described in [Apperley & Spence, 89] has been implemented in a practical computer environment as an interactive interface design tool.

The user interface simulator (Slc) is a very effective and reliable interface simulating and testing tool which supports quick and convenient user interface simulation. Using Slc, a menu interface can be quickly simulated in its design environment, where a menu-based interface can be partially or wholly simulated and invalid menu structures can be dynamically modified, or in its application environment, where evaluators are given a real feel of how this menu-based user interface works.

The user interface generator (Glc) is used to generate basic interface source code files for a user interface from its Lean Cuisine graphical specification file, and a working model of a user interface can be easily and quickly implemented without programming.

The integrated user interface development environment based on the Lean Cuisine graphical notation [Apperley & Spence, 89] successfully integrates a graphical notation, the visual programming technique, with an existing programming toolkit and offers advantages over other User Interface Programming Toolkits, language-based UIMSs and current Visual Programming Tools. It supports three main phases (design, prototyping and implementation) of the graphical user interface development lifecycle. This approach has not been found in previous user interface development tools and user interface management systems.

# Acknowledgments

I would like to thank my supervisor Professor Mark Apperley for his guidance, encouragement and help during my study.

Thanks also extend to all staff and postgraduate students in the Department of Computer Science for their assistance and friendship.

On the personal side, I would especially like to thank my wife, Bei Zhong, and my family for their continued support and encouragement.

# Table of Contents

# Chapter 1

# Introduction

## 1.1. User Interface Development

Creating a good user interface for a system is a difficult task, and the software to support user interfaces is often large, complex, and difficult to debug and modify [Myers, 88]. Current techniques for the specification and development of user interfaces can be divided into three basic groups: User Interface Programming Toolkits, User Interface Management Systems (UIMSs) or User Interface Development Systems (UIDSs), and Visual Programming Tools. Although many tools have been created to make user interfaces easier to design and implement, their development is still a hard work. It is generally the case that as user interfaces become easier to use for the end user, they become more complex and harder to create for the user interface creator. Direct-manipulation user interfaces are particularly difficult to create because they often provide elaborate graphics, many ways to give the same command, a mode free interface, and rapid semantic feedback [Myers, 89].

## 1.2. The Limitation of Current Approaches

Many solutions for finding better methods of creating good user interface have been proposed in the areas of User Interface Programming Toolkits, User Interface Management Systems(UIMSs) and Visual Programming Tools. Many tools have been created to make user interfaces easier to design and implement but each method has its own attendant problems.

### 1.2.1. User Interface Programming Toolkits

User Interface Programming Toolkits such as Motif [Young, 89], Macintosh Toolbox [Apple, 85], OpenLook [Sun, 90A], and XView [Heller, 90], provide a conventional programming interface and these programming toolkits are widely used for user interface construction. With all toolkits, the user interface designer writes programs in a conventional programming environment to create the user interface. The disadvantages

of using toolkits are that they provide limited interaction styles, lack of support for easy prototyping, difficulties in ease of use at least as great as the base language, no abstraction mechanisms, no path of communication between the user interface designer and the end user [England, 90], and they cannot be used by non-programmers.

### 1.2.2. UIMSs

The problems with User Interface Programming Toolkits have led to the creation of UIMSs (User Interface Management Systems). UIMSs attempt to provide solutions to some of the problems of conventional programming toolkits. In most UIMSs, the designer specifies the user interface with a special purpose language and provides abstract user-system dialogue specification mechanisms. Their descriptions are generally interpreted, thus aiding prototyping to some degree. But they still require programming, and with an ordinary text editor it is not easy to create, debug, and modify a textual specification file for a complex user interface, and textual specifications of user interfaces are too complicated to be understand and used. So language-based UIMSs are not suitable for the non-programming interface designer.

### 1.2.3. Visual Programming Tools

Visual Programming Tools offer advantages over interface toolkits and language-based UIMSs inherent in their direct manipulation approach. No programming language has to be learnt, and immediate feedback is provided to the designer. Visual user interface tools fall into two categories, those built on existing programming toolkits and those aimed at more general user interface construction. An example of former category is OpenWindows Developer's Guide (Devguide) [SUN, 90B], it is a graphical user interface design tool built on XView user interface toolkits and that allows the user interface designer to interactively construct the user interface by drawing the desired interface primitives on the screen. The interface is automatically prototyped. But Devguide cannot offer this drawing function to all interface primitives; for example, Devguide uses the menu editor window to construct menus. Also Devguide cannot organize a complex user interface because it lacks specification of the user interface and cannot support the dynamical creation of hierarchical menus from their specification at run-time. An example of later category is Peridot [Myers, 88] which offers programming by demonstration for component construction and component behaviour specification. The information used to construct an interface is implicit and therefore hidden in the system. Also the interventions of the inference mechanism can become tedious for an experienced user [England, 90].

## 1.3. A New Menu Interface Development Environment

In order to make user interface development much easier and quicker, a new menu user interface development environment based on the Lean Cuisine [Apperley & Spence, 89] diagrammatic design tool has been built. The main goals for developing this menu user interface development environment are :

- To provide an interactive menu interface design environment and allow user interface designers who are not programmers to create menu interfaces by using Lean Cuisine graphical notation.

- To provide an automatic prototype environment and allow user interface designer to simulate the design partially or wholly at any design stage.

- To provide a tool used to generate source code for a user interface from its Lean Cuisine specification file.

- To support user interface development from design, simulation to direct implementation based on the Lean Cuisine graphical description.

- To ensure a true separation of concerns between the application and the user interface.

The new menu user interface development environment based on the Lean Cuisine graphical notation [Apperley & Spence, 89] aims to fulfil some of the deficiencies outlined in User Interface Programming Toolkits, UIMSs and Visual Programming Tools above, and to support the graphical user interface development lifecycle. The details of how these goals were achieved will be discussed in later chapters.

## 1.4. Overview of the Thesis

**Chapter 2** presents a review of several existing user interface development tools in chronological order, and briefly discusses the limitation of these tools.

**Chapter 3** discusses models of software development lifecycle and graphical user interface development lifecycle, main graphical environments for user interface

development, notations for dialogue specification, and the integrated interface development environment based on the Lean Cuisine graphical notation.

**Chapter 4** briefly reviews menu structures in the Lean Cuisine graphical notation, then describes in detail the basic structure and functions of Elc, the Lean Cuisine graphical editor, and discusses how a user interface designer interactively design the graphical dialogue specification by using the Elc tool. The user interface simulation at the design stage, and the implementation environment of Elc are also discussed.

**Chapter 5** briefly reviews some existing interface testing tools and describes in detail Slc, an interface simulating tool used in the Lean Cuisine menu interface development environment to support quick and convenient user interface simulation based on the Lean Cuisine graphical notation.

**Chapter 6** describes in detail Glc, a software tool used to generate basic user interface source files for a user interface from its Lean Cuisine graphical specification file.

**Chapter 7** gives two detailed examples of software development in the menu interface development environment based on the Lean Cuisine graphical notation, and shows that a menu-based user interface can be rapidly simulated, modified and implemented based on its Lean Cuisine graphical specification.

**Chapter 8** reviews the integrated user interface development environment based on the Lean Cuisine graphical notation and provides conclusions.