An E-guide to Appalachian Forest Trees

Vamsi Komarneni

Problem Report submitted to the College of Engineering and Mineral Resources
At West Virginia University
In partial fulfillment of the requirements
for the degree of

Master of Science
In
Electrical Engineering

Lane Department of Computer Science and Electrical Engineering

Morgantown West Virginia
2007

Keywords: Field Guides, Computer Aided Approach

ABSTRACT

An E-guide to Appalachian Forest Trees

Vamsi Komarneni

There is a growing demand for computer applications to identify plants and trees in different fields like agriculture, horticulture and forestry. Though there are some studies carried out to bring some automation to the plant identification process, none of them are portable and user friendly. In this context this study will address the challenge of providing an application that will help a forest enthusiast to identify specific trees.

ACKNOWLEDGEMENTS

I am mostly in debt to Dr. V. Jagannathan, committee chair, for his caring guidance, active involvement, and constant assistance throughout my study as well as my major advisor for my problem report. I would also like to thank my committee members, Dr. James Mooney and Dr.Tantiana Borisova for their insightful suggestions, constructive criticism, and support.

I am most appreciative of the opportunity to further my education at West Virginia University. I wish to thank my parents for their encouragement and support. I would also extend my special thanks to Yoganand Budumuru for his constant support and encouragement throughout my studies at Morgantown and in the completion of my problem report.

I would also like to express my gratitude to all my friends, fellow students, the staff and faculty for their friendship and help and moral support and encouragement

TABLE OF CONTENTS

Disclaimer: - For better understanding some of the technical description in Appendix is reproduced as it is from http://www.floridagardener.com/misc/Leaves.htm

Chapter 1

1.1 Introduction

All living things are classified in to groups called species. A **species** can be defined as one of the basic units of biological classification that has similarities in appearance, anatomy, physiology and genetics due to having common ancestors. Traditionally, multiple examples of a proposed species must be studied for unifying characteristics before it can be regarded as a species.

Identifying and classification of trees is an important part of Forestry Judging. Every year there are over a millions who visit forests. Though millions visit forest every year, only few know about the trees that are common. Though for most of the people who have curiosity about nature such as forests and different trees grown in forests, they are constrained by the fact there is no proper tools that are easy to use and are readily available. Recognizing this problem and potential application in this field, this study is an attempt to develop an application where we can integrate existing technology in to smart devices to overcome this constraint.

1.2 Objectives:

Though the classification and identification trees are not a novel idea, previously there were few researchers who made an attempt to develop an application for identifying trees. However none of them are portable. The main advantage of being portable is that user can easily carry such a device where ever he goes and no matter how remote is the location is. So, keeping in view its advantages of being easy to carry, simplicity and to make use of the PDA technology in the market I started to design a PDA application. This study will address the problem of portability in identifying the trees with overall objective of developing a fully fledged application on a portable device with the following specific objectives:

1) Developing a Fat Client Application
2) Developing a business logic code that will help in identifying trees
3) Developing an back end XML file that is used for identifying the trees in various forests

Chapter 2

2.1 Application Review

There are quite a few number of web applications in the present market. Few applications made a significant contribution are

1) Tree identification application by Arborday.org
2) A forestry educational site for Virginia schools by Dr. Jeff Kirwan and James Ward

Tree identification application by Arborday.org

This application offers comprehensive information on generally planted trees that grow throughout the United States. Information on height spread soil and sun requirements, leaves and fruit, history, wildlife habitat. This type of information is helpful for both professional though would be adequate to non professionals. A detailed information about trees is presented which we can recognize using common name, scientific name or family. It will also allow you to search for a tree by common name or by scientific name by typing into the search tool

A forestry educational site for Virginia schools

This application is confined to only tree that grow in Virginia forests. This application tells about the about the different characteristics that are key elements that are used in the identification of the trees. Choosing any one of the key factor we can recognize the trees. In this application leaf is taken as the key element. Most of the applications choose leaf because it is easier to recognize the characteristics of the leaf.

Chapter 3

3.1 Conceptual Framework

3.1.1 Fat Client/Thick Client

A fat client will use most of his local resources in order to perform a task rather than being distributed over a network as in the case of the thick client. Most PCs (personal computers), for example, are fat clients because they have their own hard drives, CD/DVD drives, software applications and so on.

Fat clients are most preferred ones because they are very customizable and the user has more control over the programs that are being installed for a specific system configuration where as thin clients are more easily managed and low maintenance costs and could be easily secured.

3.1.2 Developing the Business logic

Business logic can be stated as a functional algorithm which handles information exchange between a back-end application usually a database (in this case it is an XML file) and user interface. Usually the term is used with most of the web application where both front-end and back-end of an application does not reside at one place.

3.2 Choosing Right Framework

J2ME and .Net CF are frameworks that are being developed to support developing applications for smart devices. After these technologies came in to existence designing applications for smart devices has become lot more easily. Let's look at a brief comparison between these technologies.

3.2.1 Platform Support

.Net CF runs only on windows compact edition of operating system. However as .Net CF uses common language runtime, its applications are portable many devices that support Windows CE and Pocket PC based OS. Other implementations of CLI exist such as Potable .NET and Mono where we can develop application on to UNIX based operating systems.

J2ME excels over .Net CF over here. It runs on non-windows operating systems like Symbian, iDen, Brew except on Palm OS. All these Operating Systems mentioned above have built in support for Java. There are third party run times which support for windows OS also.

3.2.2 Language Support

Though J2ME supports various platforms due to different standardizations followed by various platforms in reality it's really difficult to develop applications using J2ME. Different vendors have their own proprietary extension packages which makes it difficult to the developer in developing applications.

With .NET CF this aspect is totally different. Developers who are familiar with any language supported by Visual studio can easily develop an application using .NET CF.

3.2.3 The specification process

Except for a few classes which are unavailable for use within .NET CF library for portability reasons, .Net CF is closely associated with .Net Frame work. Because Microsoft holds proprietary hold on this technology there are no lengthy debates on the features that need to be incorporated in to the compact frame work.

In contrast to .NET compact frame work J2ME specification implementation is very difficult because of various vendors who are involved in the development of J2ME

3.2.4 Development Tools

There is no other vendor who can compete with Microsoft when it comes to Integrated Development Environment**.** Visual Studio is the IDE used for developing application in .NET CF where as there are several tools for J2ME application development.

Both the J2ME and .NET CF are excellent platforms for developing mobile applications, though. However, each platform has its own strengths and weaknesses. J2ME beats .NET CF when it comes to portability and cross platforms. However developing applications on J2ME is quite challenging and tedious. In respect to developing applications .NET CF is mostly desired because of its ease in development and also as its supports different languages. For High end PDA's .Net CF is easier to work with.

Keeping in view the above advantages .NET CF is chosen for application development.

3.3 .NET Compact Frame work

The .NET Compact Framework makes life easier for developers in developing application to smart devices such as Pocket PC's (Pocket PC 2002, Pocket PC Phone Edition) and other devices running Windows CE.NET 4.1 or later.

The .NET Compact Framework has two main components

1) the common language runtime
2) The .NET Compact Framework class library.

3.3.1 Common Language Runtime

The common language runtime (CLR) is similar to Java Virtual Machine (JVM). It is a virtual machine component of Microsoft .NET frame work. It is a common language infrastructure (CLI) that manages code which targets .NET Compact Framework. The

CLR runs a byte code called Microsoft Intermediate Language. This is stated as Common language infrastructure because ever thing will be converted in to common byte code which will be independent of language used. Microsoft implementation of CLR addressed many serious problems that are present in low level programming languages such as C. Tasks such as memory management, thread management; exception handling, garbage collection and security are few of them that need special attention in programming languages like C, where as in .NET they are managed by .NET runtime. Though mobile applications are written using VB or C# external libraries can still be accessed using Windows CE APIs. Windows CE API's can be accessed through applications as it provides data types and support for structures.

3.3.2 .NET Compact Framework Class Library

The .NET Compact Framework class library is a collection with reusable objects or classes that function within the limits of common language runtime. The .NET Compact Framework supports wide variety of tasks such as programming user interface design, access to XML, database, thread management and file I/O management.

Following is a list of functionality available through the .NET Compact Framework that is utilized in developing this Application.

3.3.2.1 Form-related Classes

The .NET Compact Framework form classes implement a subset functionality of Windows forms and windows drawing classes. These form classes and drawing classes' helps in constructing a rich Windows CE based user interface. The form designer manages much of the interaction with the classes. Due to constraints in its size and performance some control properties and methods and events are not included in the .net compact framework, although it implements on most of the controls. The implementation of the functionalities that was not included in the .net compact framework can be achieved with little bit of coding. This feature is supported by .net compact framework which allows inheritance from the base control class.

3.3.2.2 Data and XML Classes

.NET Compact Framework allows incorporating data, XML content, into your mobile applications using System.Data and System.XML classes. The .NET compact framework implements a subset of both data and XML classes of the .NET framework. Functionality in the .net compact framework does not support XmlDataDocument, Extensible Style sheet Language Transformation, Validating entity references on an applied DTD. Apart from the above mentioned functionalities .NET Compact framework supports most of the other functionalities.

3.3.2.3 Web Services

Much of the System.Web namespace in .NET Compact framework is scaled down version of its original capabilities and functionalities of .NET framework. The major difference between these two namespace is that both can create web services clients but .NET Compact framework would not be able to host services. Clients can be either synchronous or asynchronous.

3.3.2.4 Input/output (I/O)

As there are differences in devices there are constraints and restrictions on the I/O model. File change notifications are not notified by .net compact framework. File and Directory attributes cannot be set or accessed because devices I/O occur in RAM.

3.3.2.5 Networking

Few classes are like Infrared Data Association (IrDA) classes for making infrared connections and Web listening classes for servicing HTTP requests are only specific to .NET Compact framework.

3.4 Advantages of Visual Studio IDE and .NET technologies

.Net is able to support multiple programming languages. Hence, .Net CF can reach a variety of developers and reuse existing libraries. Although it supports multiple languages because of its CLR and object oriented in nature developers must me familiar with the OOP concepts. The .Net CF development tool, VS.Net, currently supports only two major .Net languages but also considerable effort to include J# is also done.

Chapter 4

4.1 System overview

The Appalachian Guide for trees application allows the user to make a selection between two different types of selection

1) Tree Identification using Image.
2) Tree Index (Help)

Tree Identification using Image

The system walks through a series of questionnaire to the user such as how the leaves look like, how the leaves are arranged, length of the leaf. Depending upon the selection of the user the system tells at what tree he is looking at. Thus using this option user can identify the trees with the image. An appropriate text will also be displayed below the image so that user can easily identify using text if needed.

For example an image is displayed for the leaves being oppositely arranged. The user after looking at the tree makes the appropriate selection in the application. Depending upon the user's selection of the key factor another set of images will be displayed to the user. After a new set of images pop up the user makes a selection depending upon what the image looks like. After a series of such selections a final image will be displayed giving out the full information about that image. Thus at the end user can learn about the tree and a brief information on the selections he made in order to obtain this tree. If the user wants to know more about the image that he is looking at he needs to click on it. After the user clicks on the image a new window pops up with a brief explanation about that image.

Tree Index

This one is completely different from the previous selection. In the previous one the user learns from the key selection he makes. But in this selection the user learns about specific trees using the name of the tree.

For example if the user wants to know about "Butternut - *Juglans cinerea",* User selects this tree by clicking on the name of the tree. After clicking on the name a new window pop up giving a brief introduction of the tree such as [1]"*The Butternut, also called white* *walnut, is found throughout Iowa mostly on bottom lands and lower slopes on moist, rich soils. The pinnately compound leaves are similar to black walnut leaves, being 15 to 30 inches long with 11 to 17 sharp pointed, oblong, finely toothed leaflets 2 to 3 inched long. The leaf stems and leaflets are velvety. The large, lemon-shaped fruit has a shell which is very rough and sharply ridged, and the kernel is edible. The nut is enclosed in yellow-green husk with short, rusty, clammy, sticky hairs. On young branches, the bark is smooth and light gray. On older branches and trunk it breaks into shallow, flat gray ridges which form a diamond-shaped pattern. The under bark is chocolate brown".* An image is also displayed with this information. Thus if a user wants to learn about a specific tree he then selects this option and information is displayed.

4.2 Requirements

As the development and deployment of the project is on two different systems here is the list of the hardware requirements that are applicable for both development and deployment.

4.2.1 Hardware and Software Requirements for development of the Application

Component	Minimum Requirements
Processor	600 MHz processor

RAM	192 MB
Available Disk space	Without MSDN: • 1 GB of available space required on system drive[2] • 2 GB of available space required on installation drive With MSDN: • 1 GB of available space required on system drive • 3.8 GB of available space required on installation drive with a full installation of MSDN • 2.8 GB of available space required on installation drive with a default installation of MSDN

Operating System	Windows 2000 Service Pack 4, Windows XP Service Pack 2, Windows Server 2003 Service Pack 1, or Windows Vista[3,4] For a 64-bit computer, the requirements are as follows: • Windows Server 2003 Service Pack 1 x64 editions • Windows XP Professional x64 Edition
CD-ROM Drive or DVD-ROM Drive	Required
Video	800 X 600, 256 colors Recommended: 1024 X 768, High Color 16-bit
Software	Visual Studio 2005, .net framework 2.0, Active Sync 3.5 or higher

4.2.2 Hardware Requirements for Deployment of the Application

Component	Minimum Requirements
Processor	200 MHz Processor

RAM	64 MB
Available Disk space	2MB of RAM required for .net compact framework 3 MB of RAM required for installing application
Operating System	Window mobile or windows compact Edition
Display	Should support 65,535 colors
Software	.NET compact framework

4.3 System Design

4.3.1 Design Overview

The design of the application starts from identifying the type of the project we need to create. As we are developing a project which has to be a rich set of forms we choose native method of developing the projects rather than developing using the managed code. Here is the list of table how we can identify different projects in .net environment in native development.

As we are developing a windows forms application we choose a Device Application type project in order to accomplish our goals. After a choosing the type of the project we need to choose the programming language that has to be used for developing this application.

[2]*C# is a modern, object-orientated language. Its garbage collection features and support for the .NET Compact Framework classes make it an ideal language for developing reliable and secure mobile applications. Visual C# for Smart Devices includes a large number of controls for quickly creating a graphical user interface (GUI), and the Compact Framework classes support features such as GDI+, XML, and Web Services. Visual C# can also call native Windows CE functions for situations not supported by the .NET Compact Framework.*

After all the stage being set for development of the application

The design of this application is divided into two different projects.

1) **Main Application** This project contains all the forms that are present in the whole application.

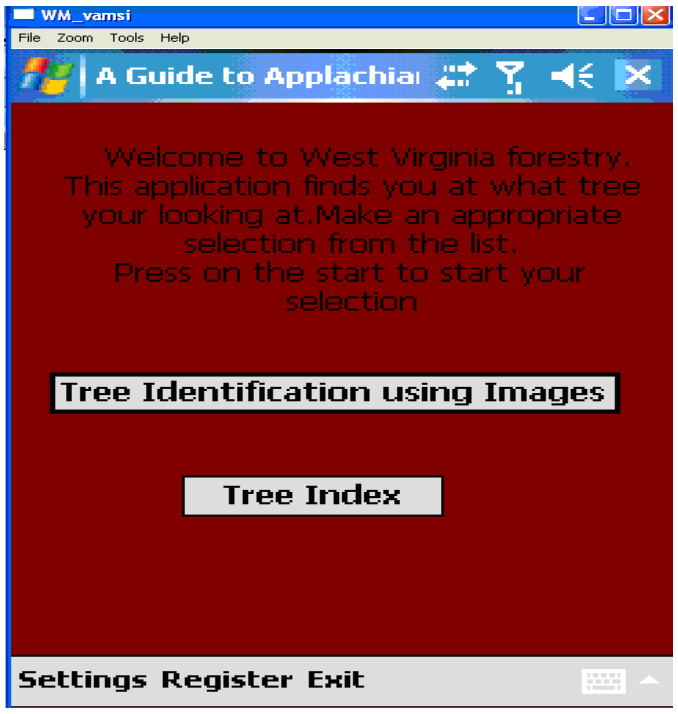
2) **Setup Project:** This project is about the setup project for the smart devices.

3) **XML file :** constitute the hierarchical arrangement of the keywords

Main Application

The main application provides with two options

1) Tree Identification using Image
2) Tree Index



Welcome to West Virginia forestry. This application finds you at what tree your looking at.Make an appropriate selection from the list. Press on the start to start your selection

FIG 1 Initial screen during the startup which displays two option Tree identification using images and Tree index

These two options are developed in a similar fashion. The whole project contains five forms.

- Start
- First Form
- Second Form

- Third Form
- Fourth Form

Start: This form in loaded whenever applications start. This form provides with the option having to select between either of the two options mentioned previously. This form is designed in such a way that it contains two button one for each option and the other button to close the application. All these buttons are included in a panel. Whenever a button is clicked respective event handlers take necessary action and load different forms accordingly.

First Form: This form will be the major back bone for this application. This is loaded in three different occasions.

1) when the application displays the final image of the tree

2) When the user click on the image for help

3) When certain tree in tree index option is selected

This form is designed with a picture box, a button, a label control. All these controls are added in to a panel control.

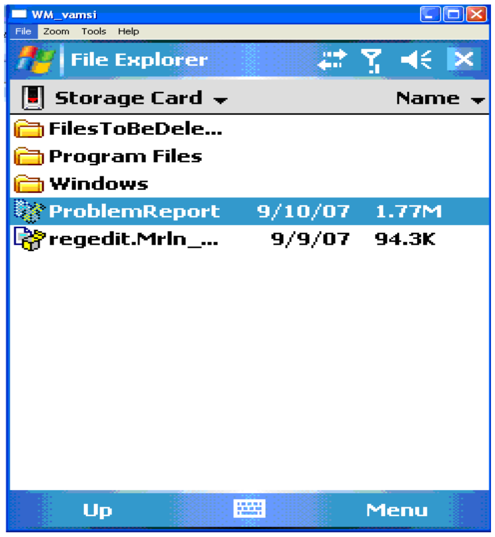
When the application reaches the end of the node i.e. if the application can decide after which tree the user is looking at then the application loads this form and displays the tree image and brief description about the image. The situation is completely different when the user clicks on the image in order to obtain some information about the image. Whenever the user clicks on the image a new form of this type will be loaded with zoomed version of the image displayed in the picture box and also with a little bit of information displayed. Below are the screenshots for three different options for this form.

Second Form, Third Form and Fourth Form: All these three forms resemble the same except that each form has different number of picture boxes. For example, second form has two picture boxes, third form has three picture boxes and fourth form has four picture boxes. Apart from the difference in the picture boxes everything resembles the same in these forms. These forms are designed in such a way that respective forms having respective picture boxes and a button control and a panel control. All these controls are put in to panel controls such that all update will be synchronous. These forms loading depend on either the number of child nodes while traversing in forward direction or on the number of parent nodes while traversing in the backward direction. Below are few screen shoots for each of the form type.

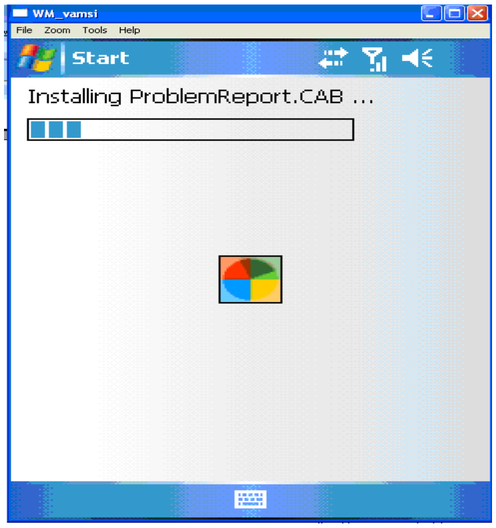
Setup Project: In order to ensure the portability of the software and allow easy distribution of the software an installation file or setup file for smart devices (CAB) is created. Thus this CAB file can be distributed to users so that the users can install on their own systems. When the user starts installing the application, windows installer checks for the existence of the .NET CF version 2.0 is installed. If either this version of this framework is not installed or if .NET CF is not installed then the application prompts with an appropriate message to the user. The user needs to install the proper version of the .NET CF before he proceeds with the next step. The application prompts where the path where the application should be installed. Depending upon the provided path the application gets installed.

While developing this setup project we need to make sure that necessary care should be take while creating the folders. We should make sure that the image folder is write-protected i.e. except a properly signed application can only write to this folder.

All the necessary shortcuts are created in the program files folder, Application folder and startup folder. With the above mentioned options when the whole project is build for release version of the project a CAB file is generated which can handed to all the users. Before releasing the application a necessary care should be taken to sign the application with a string name so that the assembly of the project doesn't get corrupted. After signing the application with the strong name the CAB file can be distributed.



FilesToBeDele...
Program Files
Windows
ProblemReport 9/10/07 1.77M
regedit.MrIn_... 9/9/07 94.3K



Installing ProblemReport.CAB ...

FIG 2 and FIG 3 these two figures show about the installation cab file being created and how setup in being done.

4.3.2 Functions used in the Application

Keepupdate(arguments) The key functionality of this function is to load a proper form depending upon the number of the nodes the current loading node has during its runtime. For example if the function is called as "keepupdate(3)" it means that the now loading node has three nodes i.e. a form with three image boxes has to be loaded in to the memory. Once the corresponding form is loaded in to the memory, now using the XML class the path of each key is retrieved and the reference of each path to is pointed to their corresponding image section. Thus upon loading this form is displayed to the user. This code for the following function is given in the appendix.

RadioButton_CheckedChanged() This event is raised whenever a radio button is either checked or unchecked. If the radio button is checked this event is automatically executed. This function in turn **keepupdate()** function depending upon the selection made. The number of child nodes for the specific selection is passed as the argument to the keepupdate function. Using this number the keepupdate function loads the required form in to the memory.

Button_Click: Back() This event is raised if the user wants to change his previous selections. A button named back is provided on the user interface. When the user clicks on this Button Back() is called and then internally keepupdate() function is called. The arguments for the keepupdate() will be the number of child nodes to this parent node and the present node is pointed back to the present parent node. Thus the parent nodes and its sibling nodes are loaded on to the form and displayed.

onClick() This event is raised whenever the user click on the image displayed on any form. As the tooltip is not provided in the .NET Compact framework this functionality is implemented as tool for help. Whenever the user wants some help on the image, the user clicks on the image. A new window pop up displaying the zoomed version of this image and a little help explaining about the image.

Button Close() This event is raised whenever user clicks on the close button. This button is present only one form. If the form is loaded from the another form i.e. if the window pop up due to the click on image then this button closes the present form and loads previous form in to the memory. If the form is loaded in order to display the last child node of the form i.e. the tree the button close opens upon the first page of the application.

4.4 Technical Constraints

In the design and implementation of our application there are many factors that affected the capabilities of this application.

4.4.1 Processing and memory limitations

PDA's are typically limited in memory and processing capabilities. The target platform used in our experiments was a Sprint Pocket PC running Windows mobile 5.0. It has 64MB RAM and 128MB of flash memory for binaries and data. The processor runs on a 416MHz clock. Due to this limitation on the processing capability and the memory we should make sure that there should not be much load on the device.

4.4.2 Limited input and screen size

The input for the device is limited i.e., due to the small screen size the text input which is typically done through a digital keyboard is difficult and so is limited. There might be few cases where a separate keyboard will be attached to the device. Though a separate device is attached as a keyboard or is present in the device itself due to limitation in the size of the keyboard it makes it hard to enter the data. The designer has to take care of such problems. Screen resolution, which is QVGA (Quarter VGA), with a useful area of 230x320 pixels, is small when compared to the normal PC.

4.4.3 Programming language, Framework and Runtime Constraints

Dot NET compact framework is a subset of .NET framework. Due to some constraints on the windows CE Some serious trimming was made to the .NET framework.

The most important functionalities that were missing are

1) Print Related Controls
2) A few XML functionalities such as XML Data Document, XML Path, and Extensible Style sheet Language Transformation, Validating entity references on an applied DTD and XSD.
3) A part of web services namespaces is trimmed i.e. does not support device based XML web services.
4) Lack of security to managed code.
5) Non existence of BinaryFormatter and SoapFormatter.
6) Access to Windows registry was not included in .NET compact framework where as access to the Windows CE registry by invoking the relevant windows API.
7) The other most notable exclusion from the .NET Compact framework is the ability to use cookies.
8) The .NET Compact Framework offers limited cryptographic abilities with respect to Web services when compared to .NET Framework

9) Windows CE does not natively support GDI+, so GDI+ related functionality was removed from .NET Compact Framework.

4.5 Policies that were made to overcome these constraints.

Processor Limitation There is nothing much we could do in this constraint. We need to make sure that there should not be much load on the process in terms of computing. Memory Limitation: This limitation can be met either by buying a memory cards that are readily available in the market. The amount on the memory card that would be required depends on the user how much he needs for other application. This application does not need much of it. A 32MB memory card which will cost less than $4 would also be sufficient.

Limitation on the framework

The limitation on the framework can either be obtained by creating a new object or else finding different alternative approach in the design. There is no tool tip available in the .net compact framework. In order to overcome this problem either a new class for tool tip has to be created or else a new method has to be used. In this application we used a total different approach i.e. whenever the user clicks on the image a new window pop up with a little bit of information regarding the image. This is how the problem of tool tip for the .net application is solved.

Due to absence of the XML.XPath class in .net XML.Xload class is used instead of XML.Xpath to retrieve the information from the XML file. Though this XML.Xload is a complicated one but due to limitation there is no other way of doing this.

4.6 Design Issues

I tried to fix all the issues in the are present in the project, but there are few situations where we need to choose between the performance of the system and memory constraints. We need to choose a tradeoff between the memory and the performance of the system

1) As we previously discussed that PDA has a memory a memory constraint on it. Due to this memory constraint we need to make sure that the design in the application should be in such a way that the application should occupy less space. One such solution would be to compress all the images in to a zip archive and try to retrieve a particular image when a request is made. This kind of design though address the memory constraint would hurt the performance of the system. Though there is a significant amount of decrease in the image storage memory but in this case the performance of the system will be reduced. Consider the lengthiest traversal of the node in XML file, it will take considerable amount of time in order to unzip the archive and then load them in to the application. Thus giving importance to the performance of the system we need to trade the memory of the system.

2) In order to keep track of all images with respect to their key values we need to arrange all the information in hierarchical order. In order to arrange the data in either

hierarchical order we need to either store information in a database or in either in XML file. The better way is to choose an XML file because it supports many other operating systems and being open source in nature. The real advantage would be if we plan to port this application to other compact operating systems then it would be easier to handle than the database.

3) The whole application can be build using one form. Depending upon the number of keys different number of picture boxes will be initialized. As we discussed earlier the limitation on the processor with this type of structure a heavy load on the processor would kill lot of RAM and also the processing clock cycles. So In order to overcome this four different forms are created and only the proper images load at proper time in to the picture boxes. This will reduce lot of burden on the processor.

Chapter 5

Conclusions

- The Guide to Appalachian forests meets all the requirements necessary to evaluate as the field guide to forests tree.
- Non Professionals or who are relatively new to this field can also use this application.
- Multiple ways of learning is implemented like learning from the characteristics of the leaves such as leaf margins, twigs and also learning about specific tree is also implemented to target different types of users.

Future Work

- There will always be a scope for improvement in the Presentation layer or the user interface of the application which depends on the feedback from the users
- With the upcoming release of .NET CF few design implementations can be changed in order to improve the performance of the system.
- As the current application is limited to only forest trees of Appalachian region there is a scope to improve its capabilities for the whole plant species in USA
- This application can also extend its capabilities to other species such as birds and wild animals, sea animals such as fish.

References

1 http://www.extension.iastate.edu/pages/tree/butternu.html

2 http://msdn2.microsoft.com/en-us/library/ms228747(VS.80).aspx

3 http://www.cnr.vt.edu/dendro/forsite/Idtree.htm

4 http://www.fw.vt.edu/dendro/forsite/key/intro.htm

5 http://www.aacps.org/aacps/boe/instr/CURR/comed/ES/webquest/ClassifyingTrees_files/ClassifyingTrees.htm#Introduction

6 http://www.must.edu.my/~dwong/resources/mobile_commerce_web/j2mevsnetcf.html#virtual

Appendix

Leaf is an aerial and lateral outgrowth of the stem of a usually flat and dorsiventral anatomy. It functions mainly to manufacture food by photosynthesis. A structurally complete leaf typically consists of a stalk also called petiole, a flattened blade, the lamina, the leaf base and strands of conducting and strengthening tissues, the veins, the pattern of the veins is called venation.

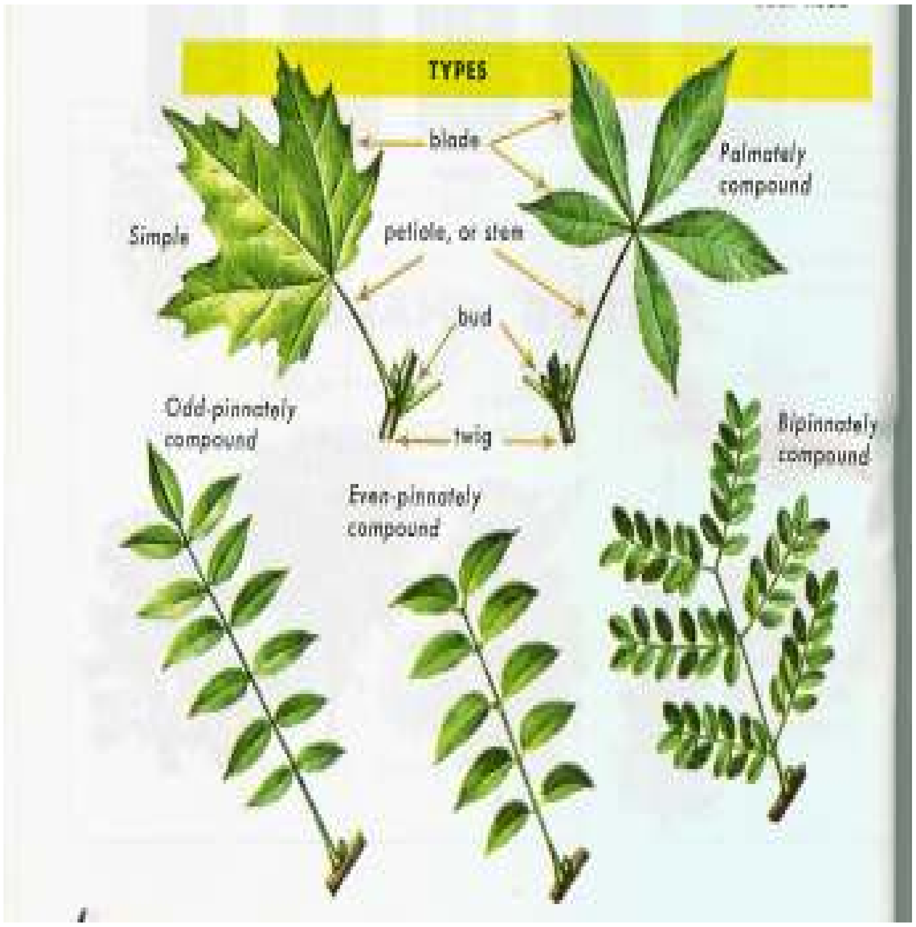


Fig 1. Leaf types

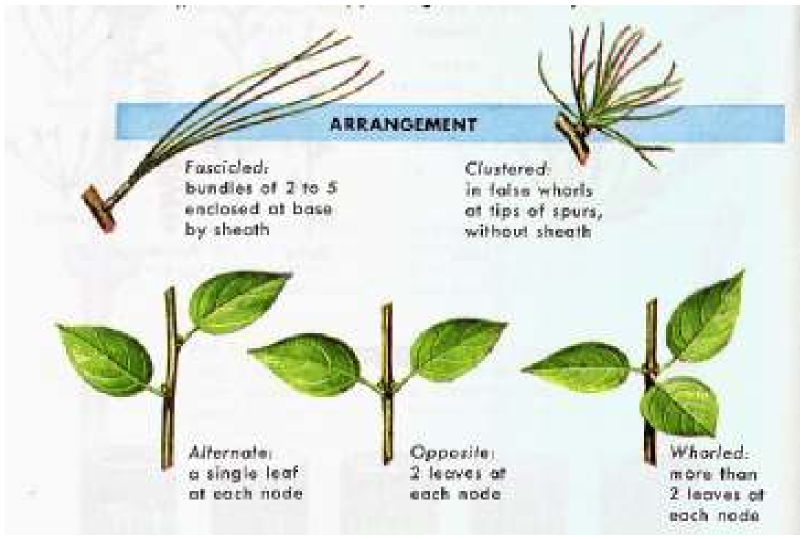
Identification of the plant usually starts in determining the leaf forms, leaf base, leaf texture, leaf tip, leaf divisions, and arrangement on the stem, venation, leaf margins, stipule type, and leaf attachments.

<u>Leaf forms</u>:

Linear - narrow, several times longer than wide, and essentially of the same width throughout.

Lanceolate - much longer than wide and tapering towards the apex from a broader base.

Oblanceolate - much longer than wide, tapering towards the base instead of the apex (the opposite of lanceolate).


ARRANGEMENT
Fascicled: bundles of 2 to 5 enclosed at base by sheath
Clustered: in false whorls at tips of spurs, without sheath
Alternate: a single leaf at each node
Opposite: 2 leaves at each node
Whorled: more than 2 leaves at each node

Oblong - nearly twice as long as broad, with the sides nearly or parallel most of their length.

Elliptic - oblong, broadest in the middle with the two ends narrowing.

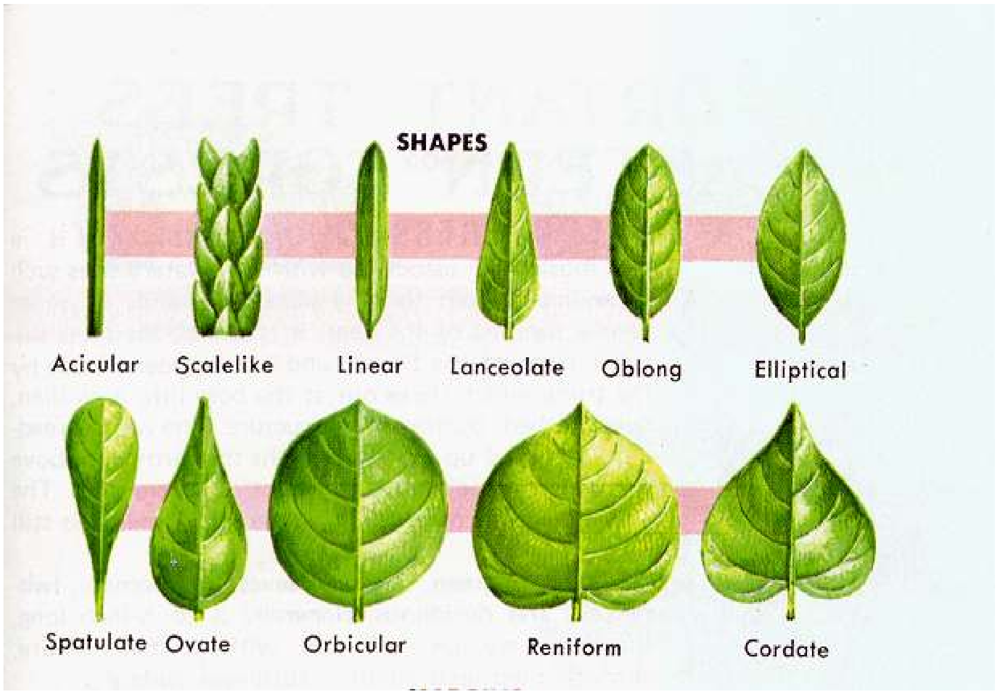
Ovate - egg-shaped, with the broadest part near the base.

Obovate - opposite of ovate, with the narrower part near the base.

Cuneate - wedge-shaped, broad at the tip and tapering by nearly straight lines to an acute angle at the base.

Spatulate - oblong but tapering to a narrow base; spoon-shaped.

Sagittate - arrow-shaped; lobes at base acute and pointing downward, while the main body tapers upward to a point.



<u>Leaf Base</u>:

Cordate - heart-shaped.

Reniform - kidney-shaped; like cordate but rounder and broader than long.

Auriculate - a small pair of projections, or ears, usually at the base.

Hastate - halberd-shaped; lobes at base pointed and narrow and nearly at right angles to petiole.

Oblique - slanting, unequal-sided.

<u>Leaf Textures</u>

Succulent - juicy, fleshy, soft, and thickened in texture.

Scabrous - rough to the touch; texture of sandpaper.

Coriaceous - leather-like, tough.

Smooth (glabrous) - surface is not hairy, rough, pubescent, or scabrous.

Downy - covered with very short, weak, and soft hairs.

Pubescent - hairy.

Canescent - covered with gray or white soft hairs as in Leucophyllum frutescens (Texas sage).

Tomentose - covered with matted, woolly hairs.

Hirsute - pubescent with coarse, stiff hairs.

Hispid - rough with bristles, stiff hairs, or minute spines.

<u>Leaf Tips</u>

Acuminate - prolonged into a narrowed or tapering point.

Acute - ending in an acute angle, but not a prolonged point.

Obtuse - blunt or rounded apex.

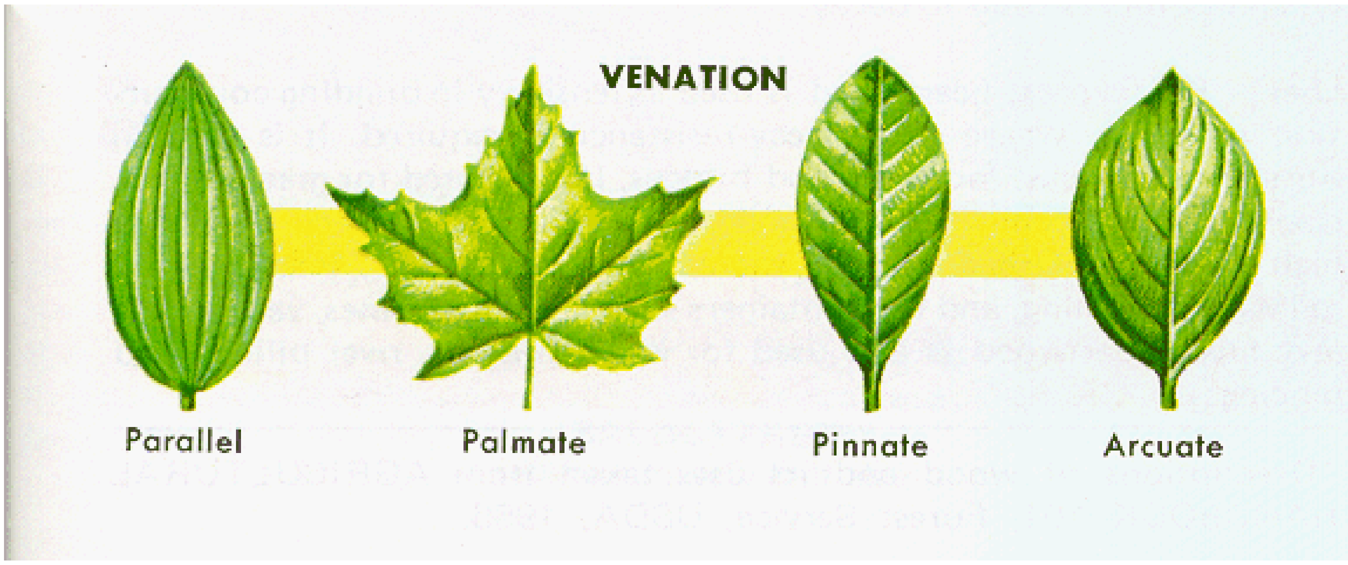
Truncate - square end that looks cut off.

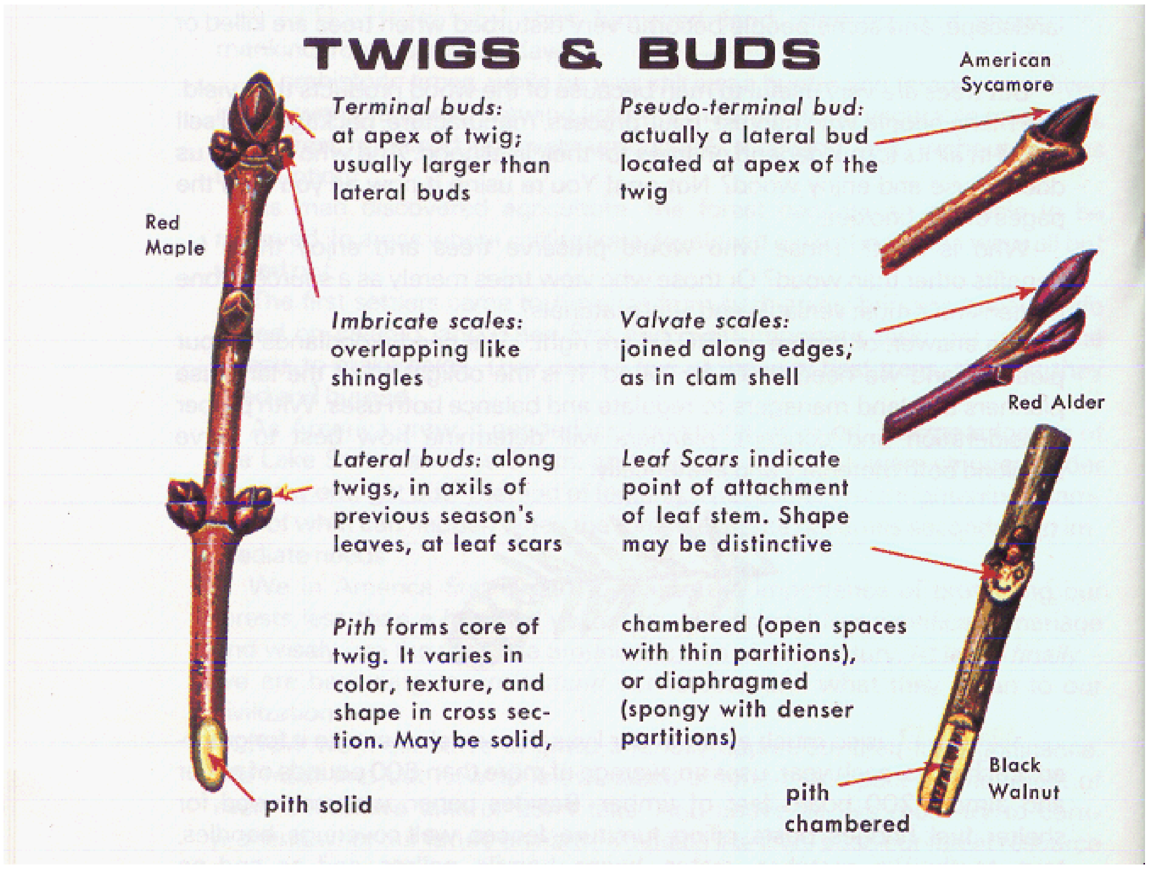
Emarginate - indented or notched.

Obcordate - inversely heart-shaped; an obovate leaf which is much more deeply notched at the tip.

Cuspidate - tipped with an elongated sharp or rigid point.

Mucronate - abruptly tipped with a small, short point; like a mere projection of the midrib.




TWIGS & BUDS
Terminal buds: at apex of twig; usually larger than lateral buds
Pseudo-terminal bud: actually a lateral bud located at apex of the twig
American Sycamore
Red Maple
Imbricate scales: overlapping like shingles
Valvate scales: joined along edges, as in clam shell
Red Alder
Lateral buds: along twigs, in axils of previous season's leaves, at leaf scars
Leaf Scars indicate point of attachment of leaf stem. Shape may be distinctive
Pith forms core of twig. It varies in color, texture, and shape in cross section. May be solid,
chambered (open spaces with thin partitions), or diaphragmed (spongy with denser partitions)
pith solid
pith chambered
Black Walnut

Leaf Division

Simple - blade is of one piece, as in Camellia japonica . It may still be simple and be lobed or cleft, as in Hibiscus rosa-sinensis (hibiscus), Quercus shumardii (Shumard oak), and Acer rubrum (red maple).

Compound - blade is made up of a number of separate leaflets. The two principal types of compound leaves are pinnate and palmate:

Pinnate - leaflets or pinnae are arranged on the sides of the main leaf stalk. Examples are Nephrolepsis exaltata 'Bostoniensis' (Boston sword fern), Roystonea regia (Cuban royal palm), and Zamia floridana (coontie).

Palmate - the leaflets are attached directly to the end of the petiole and extend outward much like fingers in a palm. Examples are Schefflera actinophylla (Australian umbrella tree) and Parthenocissus quinquefolia (Virginia creeper).

Leaf Arrangements on Stem

Alternate - one leaf at each node, as in Hibiscus rosa-sinensis (hibiscus), Brunfelsia australis (yesterday-today-and-tomorrow), and Citrus spp. (citrus).

Opposite - two leaves at each node, always on opposite sides of the stem. Examples are Catharanthus roseus (periwinkle), Ixora coccinea (ixora), and Viburnum odoratissimum (sweet viburnum).

Whorled - more than two leaves at a node spaced around the stem, as in Nerium oleander (oleander) and Macadamia spp. (macadamia).

Venation

Parallel - veined leaves, the veins run parallel to each other. This condition is characteristic of the monocotyledoneae. Parallel veins may run lengthwise on the leaf, as in Eucharis grandiflora (Amazon lily), or they may be parallel, but directed outward from the midrib to the margin (penniparallel).

Pinnately - veined leaves have a single primary vein or midrib, from which smaller veins branch off, like the divisions of a feather. Examples are Eriobotrya japonica (loquat) and Camellia japonica (camellia).

Palmately - veined leaves have several principal veins radiating from the base of the leaf blade, as in Acer rubrum (red maple) and Carica papaya (papaya).

<u>Leaf Edges/Margins</u>

Entire - even line, without teeth, notches, or lobes.

Serrate - cut into sharp, saw-like teeth pointing forward.

Dentate - toothed, teeth point outward instead of forward and are large.

Crenate - teeth are short and rounded; also called scalloped.

Undulate - margin of the leaf forms a wavy line, bending slightly inward and outward in succession.

Sinuate - like undulate, margin is very wavy (sinuous).

Incised - cut into sharp, deep, and irregular teeth or incisions.

Lobed - incisions do not extend deeper than halfway between the margin and the center of the blade and are rounded.

Cleft - incisions extend more than halfway between the margin and the center of the blade, and are sharper.

Deeply Lobed - incisions are even deeper, but not quite to the midrib or base of the blade.

<u>Stipule Types</u>

Simple - stipules located on the sides of the petiole, as in Hibiscus rosa-sinensis (hibiscus).

Adnate - stipules which adhere to the sides of the petiole, as in Trifolium spp. (clover) and Rosa spp. (rose).

Leafy - green, leaf-like stipules which serve as foliage, as in Pisum sativum (pea) and Delonix regia (royal poinciana).

<u>Leaf Attachments</u>

Petiolate - petiole (leaf stalk) is present, examples are Hibiscus rosa-sinensis (hibiscus) and Quercus spp. (oaks).

Sessile - attached directly to the main stem or branch without a petiole, as in Podocarpus macrophyllus (Japanese yew) and Gloriosa superba 'Rothschildiana' (gloriosa lily).

Peltate - petiole attached to the lower surface of the leaf instead of at the base or margin, as in Tropaeolum majus (garden nasturtium).

Clasping - leaf partially encircles the stem, as in Calendula officinalis (calendula).

Sheathing - base of the leaf is wrapped around the stem like a grass leaf, as in Zea mays (corn) and Zingiber spp. (ginger).

Decurrent - leaf base extends downward to form a wing or ridge along the stem, as in Psidium guajava (guava).

Winged petiole - petiole has a leaf-like or membrane-like extension along its length, as in Citrus x paradisi (grapefruit).

Winged rachis - compound leaf stem with a membrane-like extension on both sides of the rachis, as in Rhus copallinum (winged sumac).