NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

English Learning Program For Children

Graduation Project
COM-400

Done by : Yusra Mohamed Abdul Khaliq (20053762)

Supervisor: Asist. Prof. Dr. Ümit Soyer

Nicosia-2008

ACKNOWLEDGMENT

All praise and glory to Allah the almighty who alone made this small objective to be accomplished I feel honored and privileged to glorify his name in the sincerest way through this small accomplishment and ask him to accept my efforts.

I am greatly indebted to my project advisor Dr. Umit Soyer for his constant help, guidance and the countless hours of attention he devoted throughout the course of this work. Also I would like to place on record my great appreciation to Asst. Prof. Dr. Elbrus Imanov for his spent efforts in instructing us the Delphi program.

Also I would like to thank the Computer Engineering Faculty members and my colleagues who helped throughout my project work.

Acknowledgment is due to Near East University for extending facilities, providing recourses and support to this project.

I wish to express my heartfelt gratitude to my friends and my family for their encouragement, constant prayers continuing support and motivating.

ABSTRACT

This program "Children Education Program" is made specially for the children who are just started learning English, also it can be a good program for the children who are in a level ahead or even two levels, it can be a good teacher, specially if the child keeps learning from it and keeps as updating as the program is; because this program is going to be always altered and growing continuously which will increase the children knowledge.

This program contains lessons for teaching the simple English, starting with the alphabets through words ending with some easy sentences that will teach the children some new words and verbs.

This program is colorful and full of pictures that will attract the children and will be the best center of their attention.

TABLE OF CONTENT

INTRODUCTION

English is an international language used every where nowadays in the world. Learning English is very important for every one. This program contains the base of learning English (Alphabets).

I have chosen this program because it deals with the children, with the new generation, who has to have a good education for getting a better future. This program will be the first step on the way of the million miles.

Specially that the English Language is the language of the era as it is mentioned in the beginning, and the best time to learn is the childhood; cause as it is known that the learning at the childhood is like a drawing on the stone.

This program helps the children to learn alphabets using pictures. Also this program supports children to learn some words, and sentences by matching words with pictures.

The purpose of using pictures in this program is to attract children's and motivate them to learn because the children are usually like to deal with this kind of pictures. Thus, this program is suitable to motivate children's to start and continue learning English.

We selected the suitable name for this program (Children learning program). This is because it deals with children's only.

To use the children learning program easily, this booklet is going to be the only option. Also this booklet will contains some information about the program that is used for creating the children learning program, and the data base program that is used for making its data base.

This booklet is going to be divided into four chapters with the following topics, which will be discussed later with the shown sequence: **Chapter One:** Delphi Program, its advantages, and some other information. **Chapter Two:** Microsoft Office Access, its history, features, and some other information. **Chapter Three:** the Delphi Codes. **Chapter Four:** how to use the Children Learning Program.

CHAPTER ONE

1. DELPHI PROGRAMMING LANGUAGE

1.1 DELPHI PROGRAM

1.1.1 Delphi Description

A Delphi Development Package created by Borland it is a powerful application for Delphi development and web development.

Take enterprise application development from concept to production faster with Model Driven Architecture and UML visual design features. Build dynamic Web applications quickly and easily with RAD visual server-side Web application development.

1.1.2 Why Delphi

Borland Delphi is a development tool for Microsoft Windows applications. Delphi is powerful and easy to use tool for generating stand-alone graphical user interface (GUI) programs or 32-bit console applications (programs that have no GUI presence but instead run in what is commonly referred to as a "DOS box.")

When paired with Borland Kylix, Delphi users can build single-source applications for both Windows and Linux, which opens new opportunities and increases the potential return on development investments. Use the Cross-platform CLX component library and visual designers to build high-performance portable applications for Windows that can be easily re-compiled on Linux.

Delphi is the first programming language to shatter the barrier between high-level, easy-to-use rapid application development environments and low-level bits-and-bytes power tools.

When creating GUI applications with Delphi, you have all the power of a true compiled programming language (Object Pascal) wrapped up in a RAD environment.

All the common parts of the Windows graphical user interface, like forms, buttons and lists objects, are included in Delphi as components. This means that you don't have to write any code when adding them to your application. You simply draw them onto your form like in a paint program. You can also drop ActiveX controls on forms to create specialized programs such as Web browsers in a matter of minutes. Delphi allows the developer to design the entire interface visually, and quickly implement an event driven code with the click of the mouse.

Delphi ships in a variety of configurations aimed at both departmental and enterprise needs. With Delphi, you can write Windows programs more quickly and more easily than was possible ever before.

1.1.2.1 Pascal

The best way of describing Delphi is an Object Pascal-based visual development environment. Delphi's environment is based on Object Pascal, a language that is as object oriented as C++, and in some cases, better. For developers with no Pascal experience, its templates for Pascal program structures speed the process of learning the language.

The compiler produces applications packaged in compact executable files, with no need for bulky runtime libraries (DLL's)-a notable benefit.

1.1.2.2 Visual Component Library

Visual Component Library (self-contained binary piece of software that performs some specific predefined function), or VCL, is Delphi's object-oriented framework. In this rich library, classes for Windows objects will be found such as windows, buttons, etc, and also classes for custom controls such as gauge, timer and multimedia player, along with non-visual objects such as string lists, database tables, and streams.

1.1.2.3 Databases

Delphi can access many types of databases. Using forms and reports that the user has create, the BDE (Borland Database Engine) can access local databases, like Paradox and DBase, network SQL server databases, like InterBase, and SysBase, and any data source accessible though ODBC (open database connectivity).

1.1.3 Sample program

At the end let's see one of the smallest Delphi applications: the famous 'Hello World!' program.

This example is not for beginners – there is no main form of application or something like that. This is only a demonstration.

```
program HelloWorld;
uses dialogs;
begin
 ShowMessage('Hello World!');
end.
```

1.1.4 Warning

Delphi 7 shouldn't be installed in a directory that contains an older version of Delphi. Always the new version of Delphi should be installed in a new directory. Different versions of Delphi can coexist on the same system, but each version must be installed in its own directory.

1.1.5 Minimum system requirements

* Intel Pentium 166 MHz or higher (P2 400 MHz recommended).

* Microsoft Windows 98, 2000, and XP.

* 256 Mb.

* Approximate hard disk space required for a full install: 475 Mb (Enterprise edition).

* CD-ROM drive.

* VGA or higher resolution monitor.

* Mouse or other pointing device.

1.1.6 Delphi Advantages

* Suitable for Rapid Application Development (RAD).

* Based on a well-designed language, high-level and strongly typed, but able to use low-level code for hardware access and performance (McConnell 1993:49).

* A large community on Usenet and the web (e.g.news://newsgroups.borland.com/ and Borland's web access to Delphi newsgroups).

* Can compile to a single executable, simplifying distribution and eliminating DLL version issues.

* Many VCL (Visual Component Library) and third-party components (usually available with full source code) and tools (documentation, debug tools, etc.).

* Quick optimizing compiler also able to use assembler code.

* Multiple platform native code from the same source code.

* High level of source compatibility between versions.

* CrossKylix - a now discontinued third-party toolkit which allows native Kylix/Linux applications to be compiled from the Windows Delphi IDE, enabling dual-platform development and deployment.

* CrossFPC - a sister project to CrossKylix, not released and now inactive, which enables Windows Delphi applications to be cross-compiled from the Delphi IDE for platforms supported by the Free Pascal compiler.

* Class helpers to bridge functionality available natively in the Delphi RTL.

* The language's object orientation features only class- and interface-based polymorphism.

* Delphi 2005, Delphi 2006 and Delphi 2007 all support advanced refactoring features such as Method Extraction, etc.

* Metaclasses are first class objects.

* There are dedicated string types (as well as null-terminated strings). Strings can be added by using the '+' sign, rather than using functions.

* Objects are actually references to the objects (like in Java), which Delphi implicitly dereferences.

* Delphi is strongly type-based.

* Delphi's compiler is extremely efficient and fast.

CHAPTER TWO

2. ACCESS DATABASE

2.1 MICROSOFT OFFICE ACCESS

2.1.1 What Is Access

Microsoft Office Access, previously known as Microsoft Access, is a relational database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools. It is a member of the 2007 Microsoft Office system.

Access can use data stored in Access/Jet, Microsoft SQL Server, Oracle, or any ODBC-compliant data container (including MySQL and PostgreSQL). Skilled software developers and data architects use it to develop application software. Relatively unskilled programmers and non-programmer "power users" can use it to build simple applications. It supports some object-oriented techniques but falls short of being a fully object-oriented development tool.

2.1.1.1 Uses

Access is used by small businesses, within departments of large corporations, and by hobby programmers to create ad hoc customized desktop systems for handling the creation and manipulation of data. But, if the user is not a disciplined programmer, many errors can creep in. Access can be used as a database for basic web based applications hosted on Microsoft's Internet Information Services and utilizing Microsoft Active Server Pages ASP. Most typical web applications should use tools like ASP/Microsoft SQL Server or the LAMP stack.

Some professional application developers use Access for rapid application development, especially for the creation of prototypes and standalone applications that serve as tools for on-the-road salesmen. Access does not scale well if data access is

via a network, so applications that are used by more than a handful of people tend to rely on Client-Server based solutions. However, an Access "front end" (the forms, reports, queries and VB code) can be used against a host of database back ends, including JET (file-based database engine, used in Access by default), Microsoft SQL Server, Oracle, and any other ODBC-compliant product.

2.1.1.2 Features

One of the benefits of Access from a programmer's perspective is its relative compatibility with SQL (structured query language) —queries may be viewed and edited as SQL statements, and SQL statements can be used directly in Macros and VBA Modules to manipulate Access tables. Users may mix and use both VBA and "Macros" for programming forms and logic and offers object-oriented possibilities.

MSDE (Microsoft SQL Server Desktop Engine) 2000, a mini-version of Microsoft SQL Server 2000, is included with the developer edition of Office XP and may be used with Access as an alternative to the Jet Database Engine.

Unlike a modern RDBMS, the Access and the Jet Engine implements database triggers and stored procedures in a non-standard way. Stored Procedures are implemented in VBA, and Triggers are only available from embedded Forms. Both Triggers and Stored procedures are only available to applications built completely within the Access database management system. Client applications built with VB or C++ are not able to access these features. Starting in Access 2000 (Jet 4.0), there is a new syntax for creating queries with parameters, in a way that looks like creating stored procedures, but these procedures are still limited to one statement per procedure. Microsoft Access does allow forms to contain code that is triggered as changes are made to the underlying table (as long as the modifications are done only with that form), and it is common to use pass-through queries and other techniques in Access to run stored procedures in RDBMSs that support these.

In ADP files (supported in Access 2000 and later), the database-related features are entirely different, because this type of file connects to a MSDE or Microsoft SQL Server, instead of using the Jet Engine. Thus, it supports the creation of nearly all objects in the underlying server (tables with constraints and triggers, views, stored procedures and UDF-s). However, only forms, reports, macros and modules are stored in the ADP file (the other objects are stored in the back-end database).

2.1.1.3 Protection

If the database design needs to be secured to prevent from changes, Access databases can be locked/protected (and the source code compiled) by converting the database to an .MDE file. All changes to the database structure (tables, forms, macros, etc.) need to be made to the original MDB and then reconverted to MDE.

Some tools are available for unlocking and 'decompiling', although certain elements including original VBA comments and formatting are normally irretrievable.

2.2 WHAT ARE TABLE RELATIONSHIPS

2.2.1 Types of Table Relationships

A relationship works by matching data in key columns, usually columns with the same name in both tables. In most cases, the relationship matches the primary key from one table, which provides a unique identifier for each row, with an entry in the foreign key in the other table.

There are three types of relationships between tables. The type of relationship that is created depends on how the related columns are defined.

2.2.1.1 One-To-Many Relationships

A one-to-many relationship is the most common type of relationship. In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A.

A one-to-many relationship is created if only one of the related columns is a primary key or has a unique constraint.

In Access, the primary key side of a one-to-many relationship is denoted by a key symbol. The foreign key side of a relationship is denoted by an infinity symbol.

2.2.1.2 Many-To-Many Relationships

In a many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa. You create such a relationship by defining a third table, called a junction table, whose primary key consists of the foreign keys from both table A and table B.

2.2.1.3 One-To-One Relationships

In a one-to-one relationship, a row in table A can have no more than one matching row in table B, and vice versa. A one-to-one relationship is created if both of the related columns are primary keys or have unique constraints.
This type of relationship is not common because most information related in this way would be all in one table. You might use a one-to-one relationship to:

* Divide a table with many columns.

* Isolate part of a table for security reasons.

* Store data that is short-lived and could be easily deleted by simply deleting the table.

* Store information that applies only to a subset of the main table.

In Access, the primary key side of a one-to-one relationship is denoted by a key symbol. The foreign key side is also denoted by a key symbol.

2.2.2 Creating Database Relationships

One of the major advantages of databases such as Microsoft Access is their ability to maintain relationships between different data tables. The power of a database makes it possible to correlate data in many ways and ensure the consistency (or referential integrity) of this data, also to prevent the redundant data from table to table. For example, if we are designing a database for the Acme Widget Company, and we want to track both our employees and our customer orders.

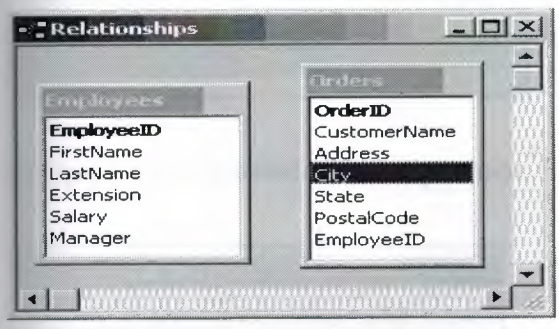
We might have a table called Orders, in this table the EmployeeID would be duplicated for each order has been ordered.

A better solution is to store the employee information only once in a separate table, Employees. We would then put a pointer in the Employees table that references an entry in the Orders table.

To make sure that our data is not out of sync, we can enforce referential integrity between the Employees and Orders tables. Referential integrity relationships help ensures that information in one table matches information in another. For example, each Order in the Orders table must be associated with a specific Employee r in the Employees table. An EmployeeID cannot be added to the database for an Order that does not exist in the database.

In this chapter we'll take a look at the process of creating a simple relationship using a Microsoft Access database. We might use a table structure similar to the one shown

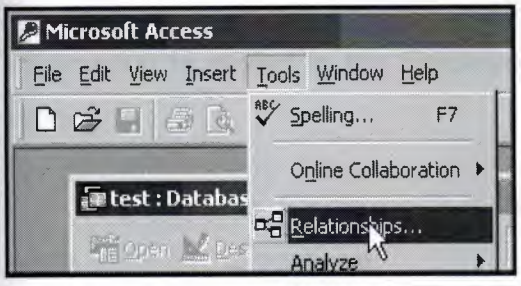
below:



Remember that each order is associated with a specific employee. This information overlap presents the perfect situation for the use of a database relationship. Together we'll create a Foreign Key relationship that instructs the database that the EmployeeID column in the Orders table corresponds to the EmployeeID column in the Employees table.

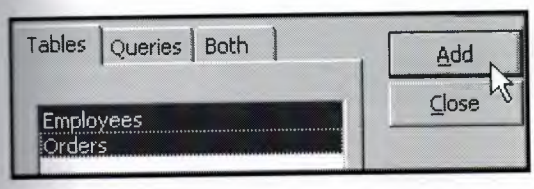
Once the relationship is established, we've unleashed a powerful set of features in Microsoft Access. The database will ensure that only values corresponding to a valid employee (as listed in the Employees table) can be inserted in the Orders table. Additionally, we have the option of instructing the database to remove all orders associated with an employee when the employee is deleted from the Employees table.

1. Open the Relationships Window from the Tools menu.

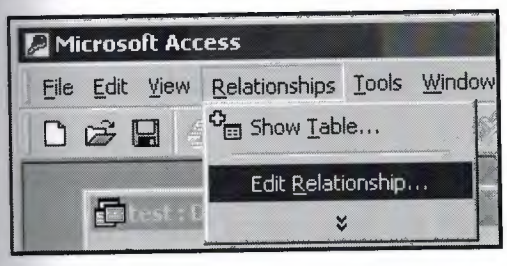


2. Add the appropriate tables. We'll add the Employees and Orders tables you can use the Shift-Click combination to select both at the same time. Once you've highlighted the

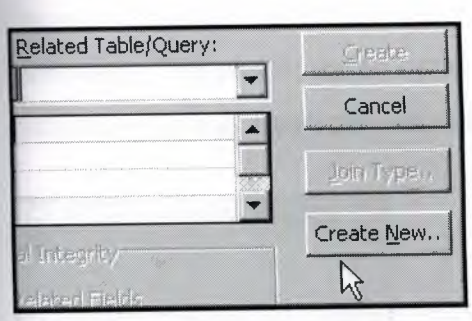
tables click the Add button.



3. **Open the Edit Relationships tool.** We'll find it on the Relationships pull-down menu.

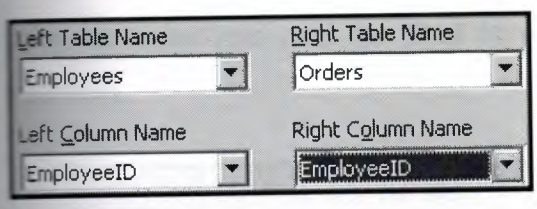


4. **Create a new relationship.**



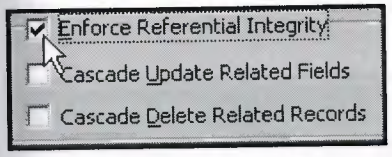
5. Fill in the appropriate details. The left table is the primary source of the data and the right table is the desired location of the foreign key. In this case, our original information is stored in the Employees table and the foreign key will be in the Orders table.

The field name is EmployeeID in both cases. Once you've selected the appropriate items, click OK to continue.



6. Choose whether to enforce Referential Integrity. In most circumstances, you will want to select this option. This is the real power of a relationship -- it ensures that new records in the Orders table only contain the IDs of valid employees from the Employees table.

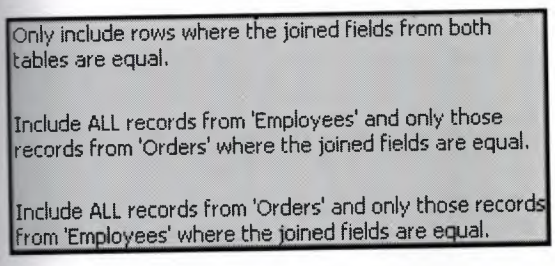
You'll also notice two other options here. The "Cascade Update Related Fields" option ensures that if an EmployeeID changes in the Employees table that change is propagated to all related records in the Orders table. Similarly, the "Cascade Delete Related Records" option removes all related Orders records when an Employee record is removed. The use of these options will depend upon the particular requirements of your database. In this example, we won't utilize either one.



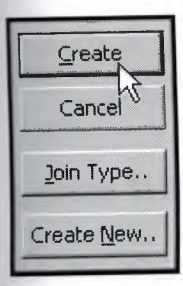
7. Open the Join Type dialog box.

8. Select a join type. The three options are shown in the figure below. If you're familiar with SQL, you might notice that the first option corresponds to an inner join, the second

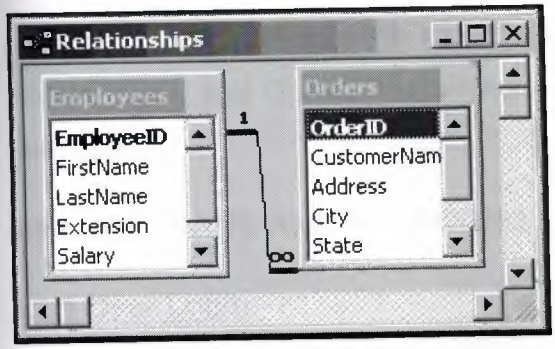
to a left outer join and the final to a right outer join. We'll use an inner join for our example.



9. Create the relationship.



The relationship window now reflects our new relationship!



CHAPTER THREE

3. HOW THE PROGRAM WORKS

3.1 INTRODUCTION

In this chapter I am going to explain how to use the Children Learning Program. All the program's components will be shown and explained in details, so after reading this chapter program will be a piece of cake.

Children Learning Program is based on the idea of dividing the users into three groups: Admin, User, and Guest. The pages and the permissions will appear according to the user group.

3.2 LOGIN NAME AND PASSWORD

3.2.1 User Login

When any of the three users opens the program the first page will appear is the password page which is seen in the picture 3.1, the user should have a login name and a password to start with the program. If the user doesn't enter them, he / she will not be able to login.

Picture 3.1

Otherwise the second page will appear, in the second page -which is the main page- at the upper part, in the left side the user can find the main menu as it obvious in the picture 3.2, the main menu is there in the all pages so the user can press any option at any time. The main menu contains the following option:

Picture 3.2

3.3 MAIN MENU COMPONENTS

3.3.1 Users

This page the will not appear to the Guest, it will be appear to the Admin and the User only. In this page the user can add new users to the program by adding all the needed information for example: user name, user group, user tell, user login name, user password and some other information.

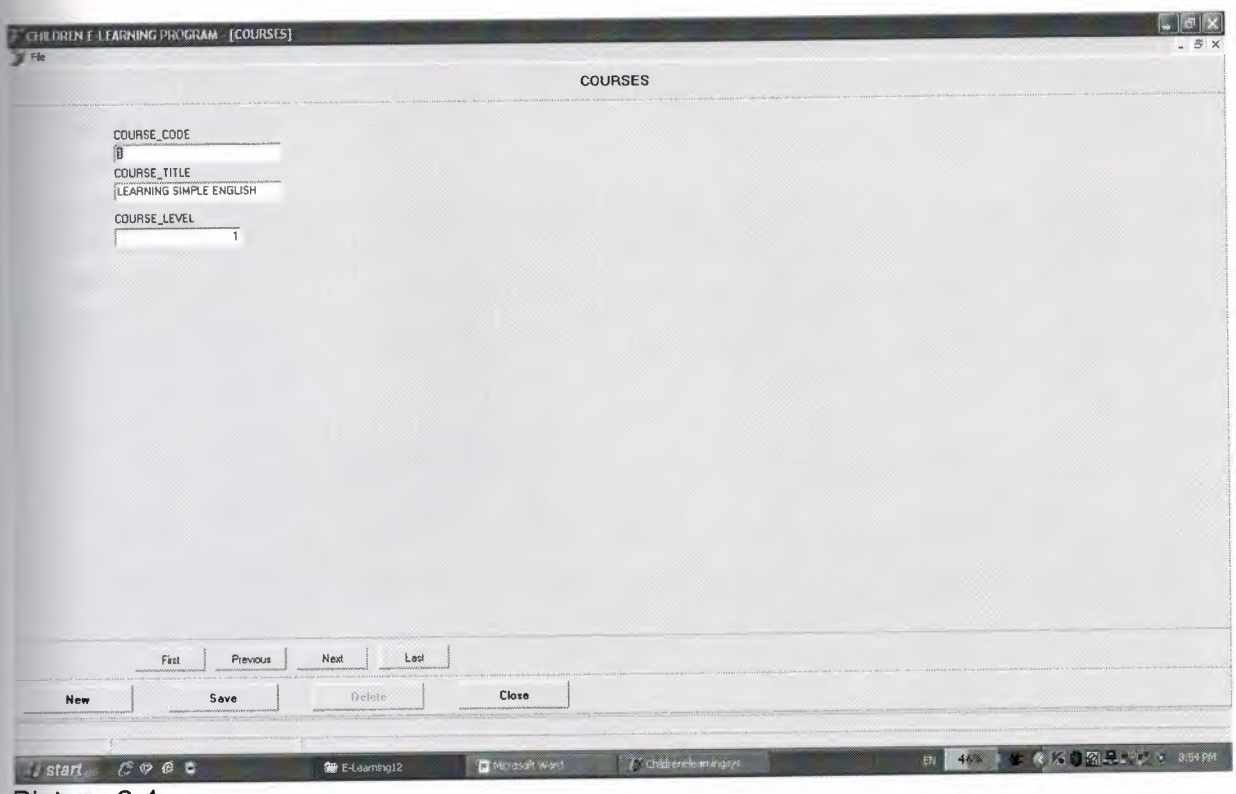
In this page only the Admin can delete users as it is shown in picture 3.3. In the middle part there are some buttons fro the sequence purpose.

Picture 3.3

3.3.2 Courses

This page will appear to the three groups of users. In this page the Guest can only search and the User can only add new courses or update the current courses, he / she can't delete anything from there as it clear in the picture 3.4, only the Admin who can delete from the program.

In this page some information about the courses are found, like course title and course level. Also there are some buttons which are based in the middle part of the page. These buttons are used for sequencing the records.



Picture 3.4

3.3.3 Lessons

This page is also is going to appear to the all users no matter to what group he / she belongs. Again in this page the Guest can only control the sequencing buttons that are placed in the middle check picture 3.5, which means for the searching purpose only. In this page the Admin and the User can add new lessons in the needed courses or update them, but again the Admin has a more powerful permissions so he /she is the only group who is capable to delete undesired lessons.

The needed information to be filled by the Admin or the User are as follows: lesson code, lesson title, lesson type: what the shape of the lesson looks like according to some characteristics known to the Admin and the User, and some other important information.

ture 3.5

.4 Lessons Data

s page is again will appear to the three groups according to the user group eviously mentioned permissions. Here is the page where the images and the images' es are added in this page see picture 3.6.

Picture 3.6

3.3.5 Exit

This is the last option of the main menu as it was shown before in picture 3.2, and it is used for the quitting from the whole program.

Back to the main page, under the main menu there is the tree of the courses. More courses will be added later but for now there is only one course which is the Learning Simple English see the first line in the tree in the red bar.

Picture 3.7

3.4 THE COURSE AND ITS LESSONS

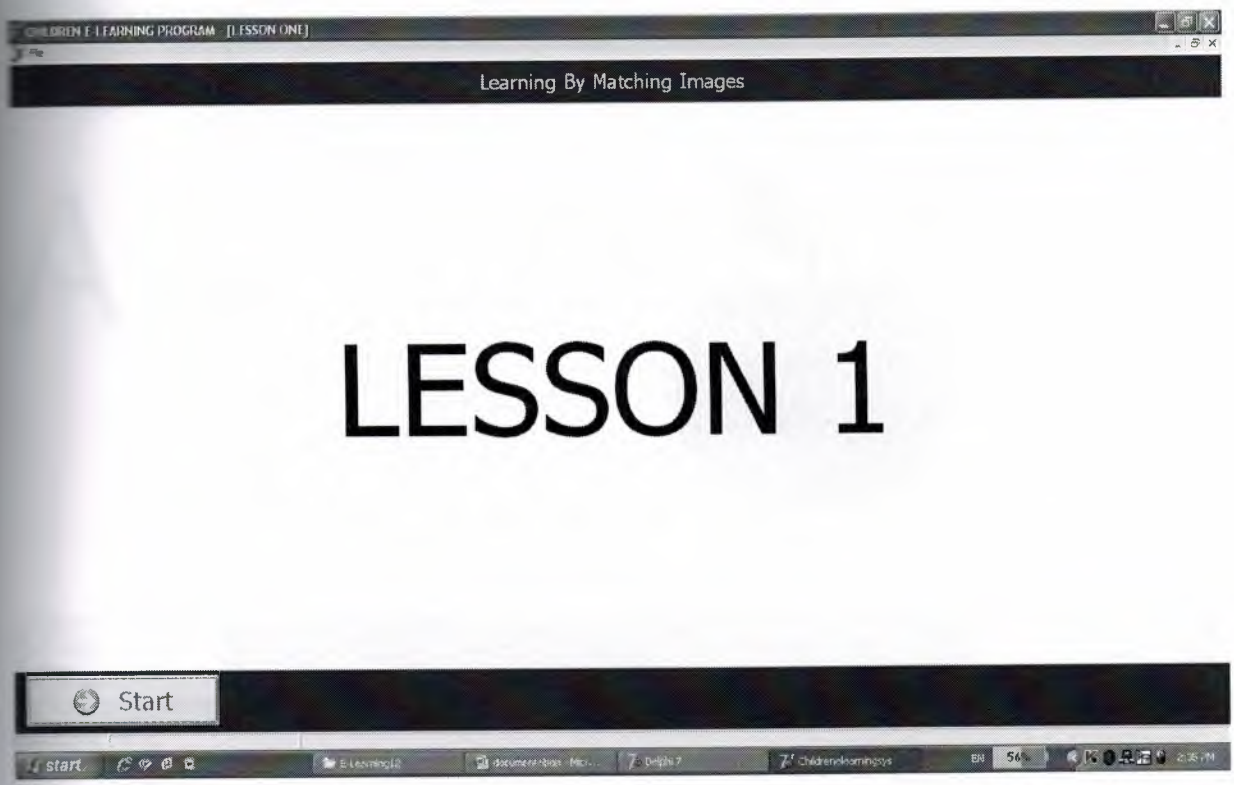
3.4.1 Learning Simple English

It is made specially for the kids who are just started learning English, also it can be a good program for the kids who are in a level ahead or even two levels, it can be a good teacher specially if the kid keeps learning from it and keeps as updating as the program is, because this program is going to be always altered and growing continuously.

In the tree under the course title there are the four following lessons:

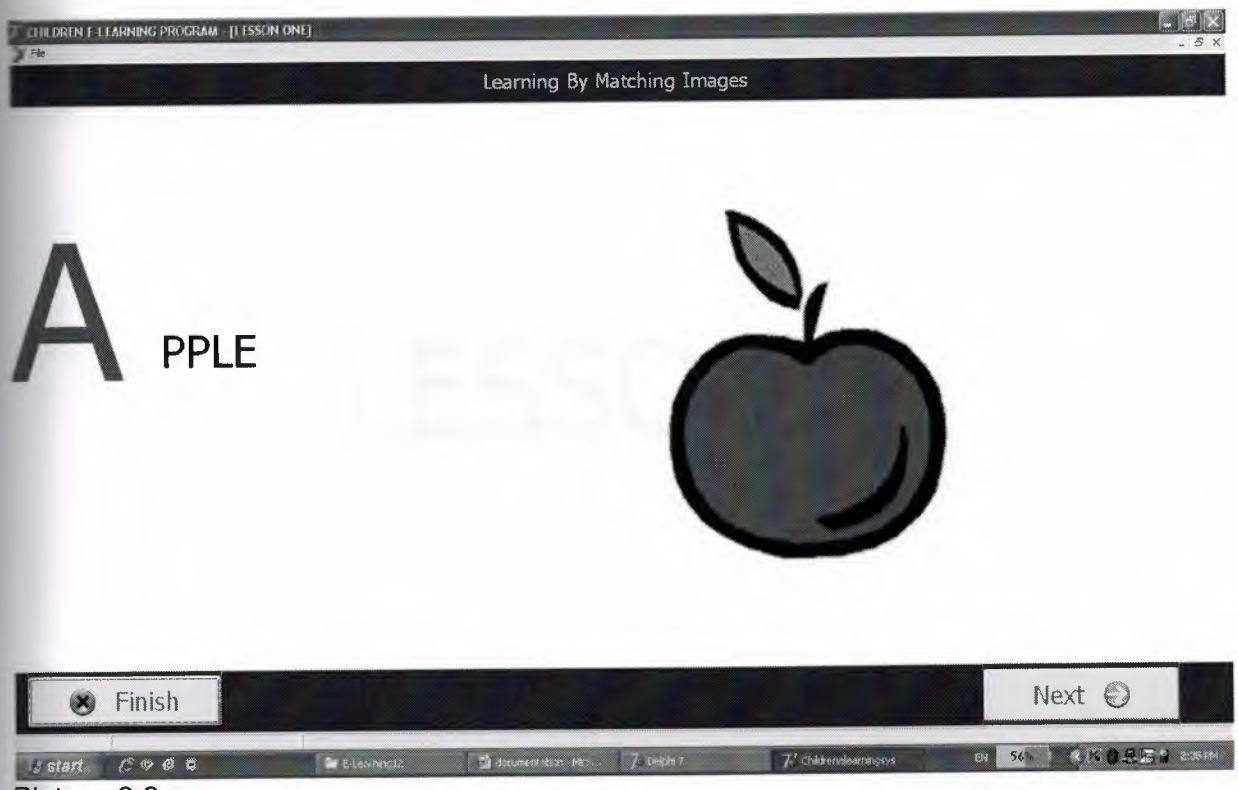
3.4.2 Learning By Matching Images

As soon as the user press on the first lesson option twice or click once then the bottun Go that is placed in the down part of the main page in the red bar as it was clear in the last picture 3.7, the following picture will appear 3.8. To start the lesson the user should press the Start bottun that is in the bottom of the page.



Picture 3.8

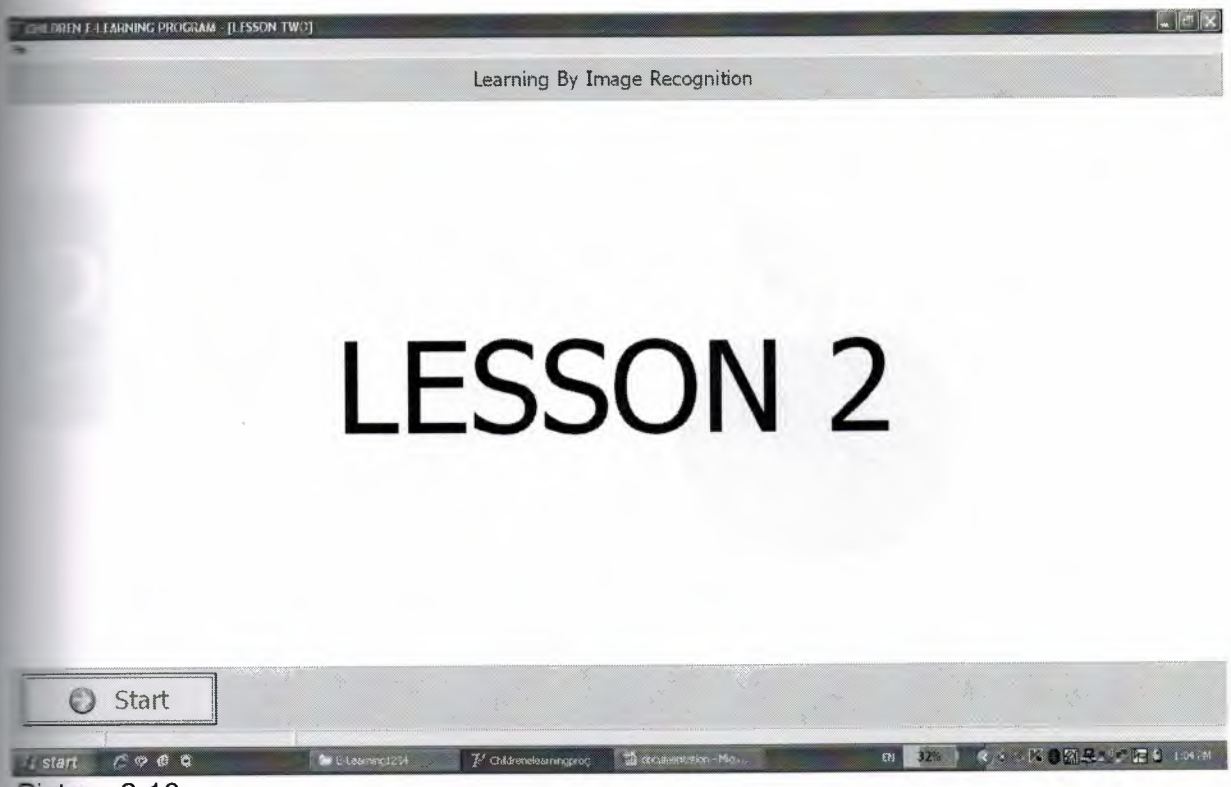
After pressing the Start botton the following picture will appear 3.9, to countinue press the botton Next. Also there is an option to go back, by clicking the button Back that will be provided after cliking on the bottun Next in the following picture. These words and pictures will be ordered sequentially from A to Z and they will be changed as long as the button Next is pressed till the letter Z, for example in the coming picture 3.9 there is a image of an apple and the word Apple, the second page will illustrate a picture of a bird and the word Bird then it will goes in this way till the end.



Picture 3.9

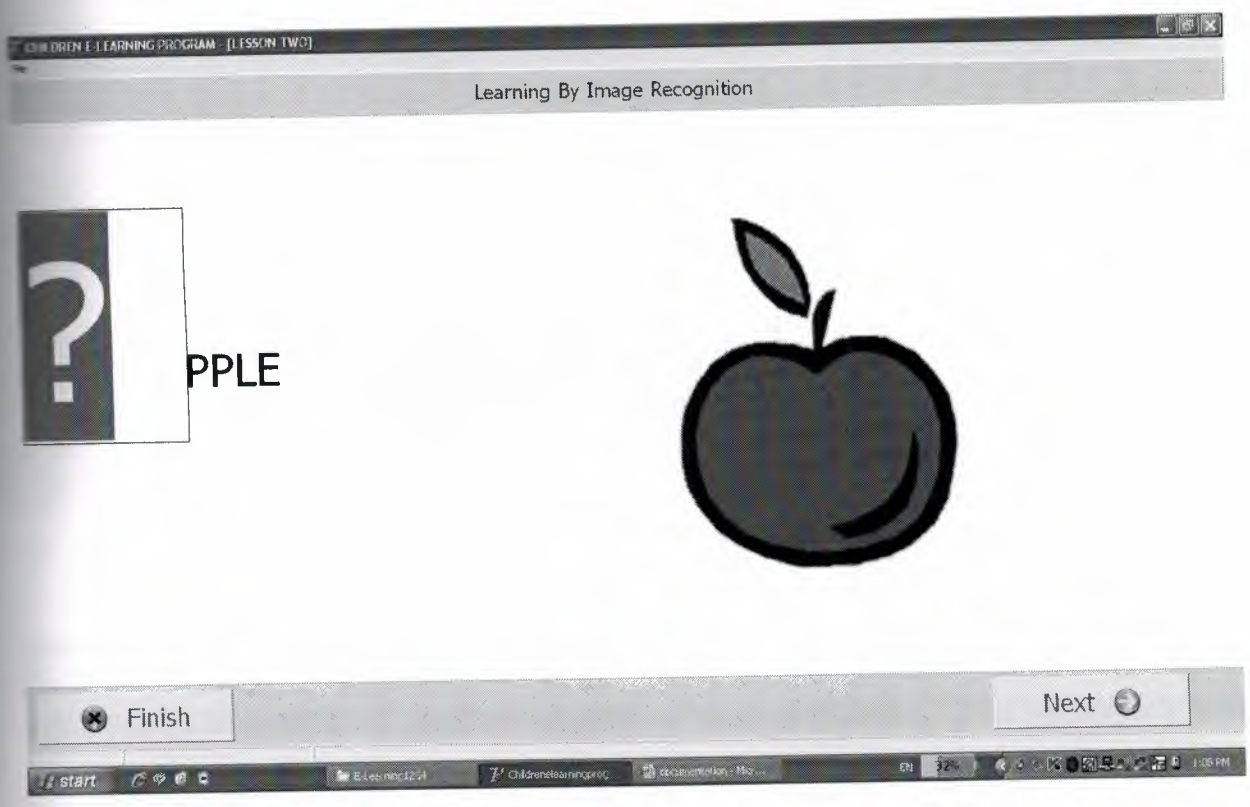
3.4.3 Learning By Image Recognition

When the user double click on the second lesson or click on the second lesson then the button Go the second lesson will appear and it will be as shown down In the picture 3.9.



Picture 3.10

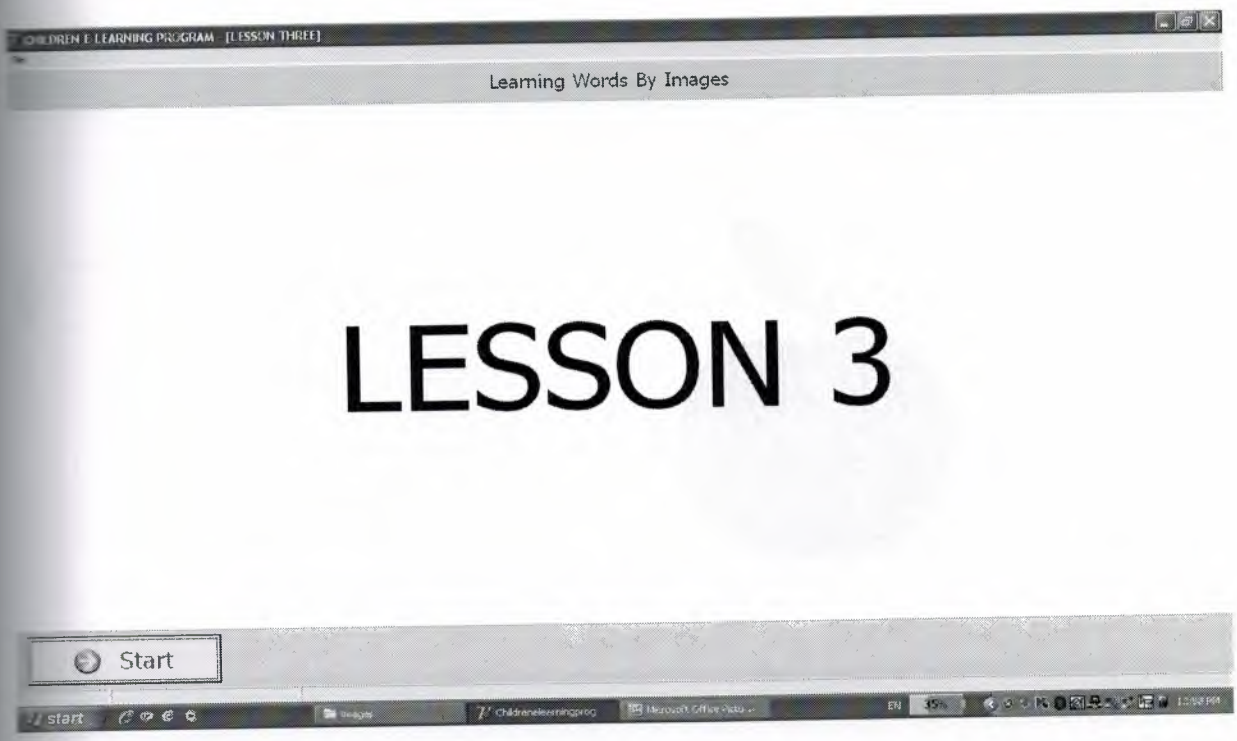
To start the lesson, again press the Start bottun that is in the bottom of the page, then the following picture 3.11 will appear. As it is illstrating in the next picture there is again a picture of an apple and in this the first letter is not typed so the user should type the missing character, if it is corect a small tick sighn will accure otherwise a small wrong sighn will be shown.



Picture 3.11

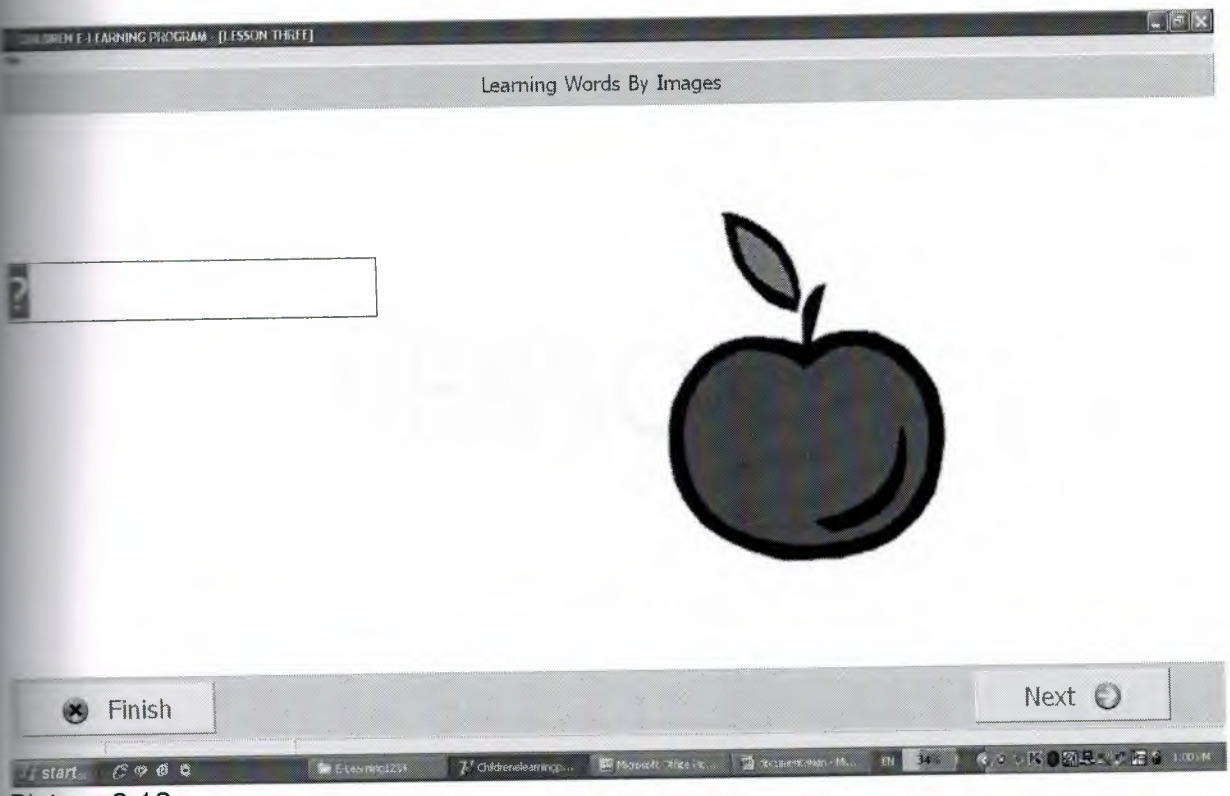
3.4.4 Learning Words By Images

When the user presses on the third lesson, the phrase lesson three will appear like the two previouse lessons and as shown down In the picture 3.12.



Picture 3.12

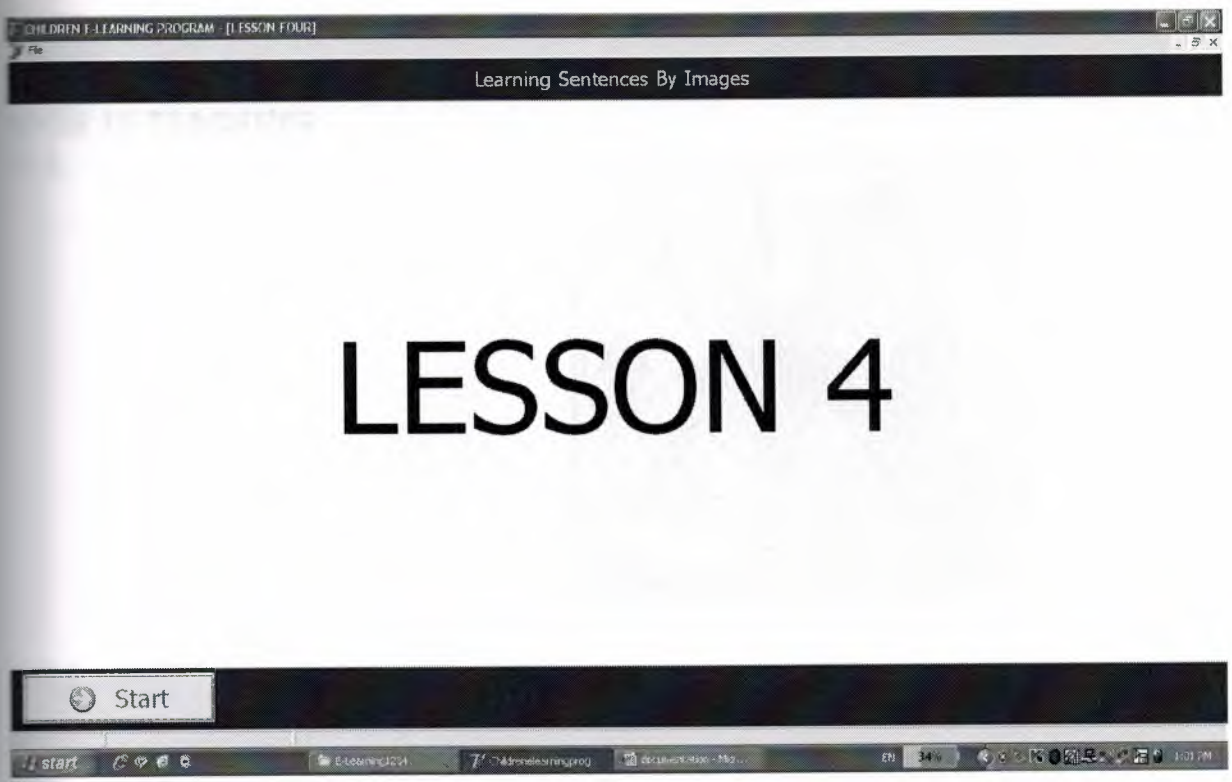
After clicling on the the Start bottun, the following picture 3.13 will appear. In this time the user will be asked to type the name of the picture that is shown on the right side if it is corect, a small tick sighn will accure otherwise a small wrong sighn will be show



Picture 3.13

3.4.5 Learning Sentences By Images

Like the prior lessons after clicking on the fourth lesson option, the phrase lesson four will appear like the as it is clear down In the picture 3.14.



Picture 3.14

When the lesson will be started the following picture 3.15 will appear. This lesson tends to teach the kids some sentences using some pictures as it is obvious in the same picture 3.15.


E-LEARNING PROGRAM - [LESSON FOUR]
Learning Sentences By Images
AMIR IS TEACHING
ALI.
Finish
Next
start
Picture 3.15

CONCLUSION

Children Learning Program is a program which has been created for children who wants to learn English. It isn't made to serve the children who have just started learning English only; it also can help children who are in a step ahead or even two.

This program is colorful and full of pictures to attract children, also it going to be changed and updated to keep its users satisfied.

This program is consisting of one course with four lessons. The first one is used for learning the alphabets by recognizing them, the second one is for learning the alphabets by typing them, and in these two lessons the child can read the words and learn them. In the third lesson the child will be asked to type the full word which has been written in the last tow lessons. The last lesson will show some sentences which will teach the children some new words and verbs. All these lesson will be supported with pictures, as mentioned before.

REFERENCES

1. http://en.wikipedia.org/wiki/Borland_Delphi
2. http://www.mnabialek.pl/en/programs-applications-programmer-applications-programming.html
3. http://databases.about.com/od/tutorials/l/aarelationship1.htm
4. http://databases.about.com/od/tutorials/l/aarelationship2.htm
5. http://support.microsoft.com/kb/304466/en-us
6. http://wareseeker.com/
7. http://en.wikipedia.org/wiki/Microsoft_Access
8. http://delphi.about.com/
9. The (install) document from the Delphi 7.0 Program CD.

APPENDIX

1.1 THE MAIN UNIT CODES

1.1.1 ChildreneLearningProg UNIT

```
program ChildreneLearningProg;

uses
 Forms,
 MainUnit in 'MainUnit.pas' {MainForm},
 Lesson3Unit in 'Lesson3Unit.pas' {Lesson3Form},
 Lesson4Unit in 'Lesson4Unit.pas' {Lesson4Form},
 LoginUnit in 'LoginUnit.pas' {LoginForm},
 UsersUnit in 'UsersUnit.pas' {UsersForm},
 CoursesUnit in 'CoursesUnit.pas' {CoursesForm},
 LessonsDataUnit in 'LessonsDataUnit.pas' {LessonsDataForm},
 LessonsUnit in 'LessonsUnit.pas' {LessonsForm},
 Lesson1Unit in 'Lesson1Unit.pas' {Lesson1Form},
 Lesson2Unit in 'Lesson2Unit.pas' {Lesson2Form};
 {$R *.res}
 begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TLoginForm, LoginForm);
```

```
Application.Run;

end.
```

1.2 THE LOGIN NAME AND PASSWORD UNIT CODES

1.2.1 Login UNIT:

```
unit LoginUnit;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, ComCtrls, StdCtrls, Buttons, DB, ADODB;

type

TLoginForm = class(TForm)

StatusBar1: TStatusBar;

Panel1: TPanel;

Panel2: TPanel;

Panel3: TPanel;

Edit1: TEdit;

Edit2: TEdit;

Label1: TLabel;

Label2: TLabel;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;
```

```
ADOConnection1: TADOConnection;
ADODataSet1: TADODataSet;
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
private
{ Private declarations }
public
{ Public declarations }
end;
var
LoginForm: TLoginForm;
implementation
uses MainUnit;
{$R *.dfm}
procedure TLoginForm.BitBtn2Click(Sender: TObject);
begin
Application.Terminate; //close bitbb
end;
procedure TLoginForm.BitBtn1Click(Sender: TObject);
begin
```

```
ADOConnection1.Close;

ADOConnection1.Properties['Data
Source'].Value:=ExtractFilePath(Application.ExeName)+'CHILDEDUCATION.mdb';

ADOConnection1.Open;

ADODataSet1.Close;

ADODataSet1.CommandText:='SELECT * FROM USERS WHERE
LOGIN_NAME=:LOGIN_NAME AND LOGIN_PASSWORD=:LOGIN_PASSWORD';

ADODataSet1.Parameters.ParamByName('LOGIN_NAME').Value:=Edit1.Text;

ADODataSet1.Parameters.ParamByName('LOGIN_PASSWORD').Value:=Edit2.Text;

ADODataSet1.Open;

if ADODataSet1.IsEmpty then //if empty or wrong

begin

ShowMessage('Incorrect Username Or Password ! Please Try Again !!!');

end

else

begin

MainForm.Start.Enabled:=true;

MainForm.File1.Enabled:=true;

MainForm.UserGroup:=ADODataSet1.FieldByName('USER_GROUP').Value;

MainForm.Panel1.Visible:=true; //show the panel with the tree

Close;
```

```
end;
end;
procedure TLoginForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Action:=caFree;
end;
end.
```

1.3 THE MAIN FORM UNIT CODES

1.3.1 MainForm UNIT

```
unit MainUnit;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Menus, ComCtrls, ExtCtrls, jpeg, DB, ADODB, StdCtrls, Buttons;
type
TMainForm = class(TForm)
StatusBar1: TStatusBar;
MainMenu1: TMainMenu;
File1: TMenuItem;
Exit1: TMenuItem;
Image1: TImage;
```

```
ADOConnection1: TADOConnection;
Panel1: TPanel;
Panel2: TPanel;
TreeView1: TTreeView;
ADODataSet1: TADODataSet;
ADODataSet2: TADODataSet;
Start: TSpeedButton;
Courses1: TMenuItem;
Lessons1: TMenuItem;
Users1: TMenuItem;
LessonsData1: TMenuItem;
procedure Exit1Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure TreeView1DblClick(Sender: TObject);
procedure StartClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Courses1Click(Sender: TObject);
procedure Lessons1Click(Sender: TObject);
procedure Users1Click(Sender: TObject);
procedure LessonsData1Click(Sender: TObject);
```

```
private
{ Private declarations }
public
UserGroup : Integer;
{ Public declarations }
end;
var
MainForm: TMainForm;
implementation
uses Lesson1Unit, Lesson2Unit, CoursesUnit, LessonsUnit, UsersUnit,
LessonsDataUnit, Lesson3Unit, Lesson4Unit;
{$R *.dfm}
procedure TMainForm.Exit1Click(Sender: TObject);
begin
Close;
end;
procedure TMainForm.FormShow(Sender: TObject);
Var
Node : TTreeNode;
begin
TreeView1.Items.Clear;
```

```
ADODataSet1.Close;

ADODataSet1.CommandText:='SELECT * FROM COURSES ORDER BY
COURSE_LEVEL ASC';

ADODataSet1.Open;

ADODataSet1.First;

While not ADODataSet1.Eof do

begin

Node:=TreeView1.Items.Add(nil,ADODataSet1.FieldByName('COURSE_TITLE').AsStri
ng);

ADODataSet2.Close;

ADODataSet2.CommandText:='SELECT * FROM LESSONS WHERE
LESSON_COURSE='+ADODataSet1.FieldByName('COURSE_CODE').AsString;

ADODataSet2.Open;

ADODataSet2.First;

while not ADODataSet2.Eof do

begin

TreeView1.Items.AddChild(Node,ADODataSet2.FieldByName('LESSON_TITLE').AsStri
ng);

ADODataSet2.Next;

end;

ADODataSet1.Next;

end;
```

```
TreeView1.FullExpand;
end;
procedure TMainForm.FormCreate(Sender: TObject);
begin
ADOConnection1.Close;
ADOConnection1.Properties['Data
Source'].Value:=ExtractFilePath(Application.ExeName)+'CHILDEDUCATION.mdb'; // to
open whereever it is
ADOConnection1.Open;
end;
procedure TMainForm.TreeView1DblClick(Sender: TObject);
begin
StartClick(Sender);
end;
procedure TMainForm.StartClick(Sender: TObject);
begin
if (TreeView1.SelectionCount>0) AND (TreeView1.Selections[0].Parent<>nil) then
begin
ADODataSet1.First;
ADODataSet1.MoveBy(TreeView1.Selections[0].Parent.Index);  //Parent.Index
ADODataSet2.Close;
```

```
ADODataSet2.CommandText:='SELECT * FROM LESSONS WHERE
LESSON_COURSE='+ADODataSet1.FieldByName('COURSE_CODE').AsString+'
ORDER BY LESSON_INDEX ASC';

ADODataSet2.Open;

ADODataSet2.First;

ADODataSet2.MoveBy(TreeView1.Selections[0].Index);  //Selections.Index

case ADODataSet2.fieldbyname('LESSON_TYPE').AsInteger of      //this is used for
lesson type

1 :

begin

Application.CreateForm(TLesson1Form, Lesson1Form);

Lesson1Form.Show;     //to show lesson with type 1

Panel1.Visible:=false; //to make the panel go when the lesson opens

end;

2 :

begin

Application.CreateForm(TLesson2Form, Lesson2Form);

Lesson2Form.Show;

Panel1.Visible:=false;

end;

3 :

begin
```

```
Application.CreateForm(TLesson4Form, Lesson4Form);
Lesson4Form.Show;
Panel1.Visible:=false;
end;
4 :
begin
Application.CreateForm(TLesson3Form, Lesson3Form);
Lesson3Form.Show;
Panel1.Visible:=false;
end;
end;
end;
end;
procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Application.Terminate;
end;
procedure TMainForm.Courses1Click(Sender: TObject);  //here course1 will be created
and for more understanding go dowm
begin
Panel1.Hide;
```

```
Application.CreateForm(TCoursesForm, CoursesForm);
CoursesForm.Show;
case UserGroup of
2 : CoursesForm.BitBtn6.Enabled:=false;
3 :
begin
CoursesForm.BitBtn4.Visible:=false;
CoursesForm.BitBtn5.Visible:=false;
CoursesForm.BitBtn6.Visible:=false;
end;
end;
end;
procedure TMainForm.Lessons1Click(Sender: TObject);
begin
Panel1.Hide;
Application.CreateForm(TLessonsForm, LessonsForm);
LessonsForm.Show;
case UserGroup of
2 : LessonsForm.BitBtn6.Enabled:=false;
3 :
begin
```

```
LessonsForm.BitBtn4.Visible:=false;
LessonsForm.BitBtn5.Visible:=false;
LessonsForm.BitBtn6.Visible:=false;
end;
end;
end;
procedure TMainForm.Users1Click(Sender: TObject);
begin
case UserGroup of
1 :
begin
Panel1.Hide;
Application.CreateForm(TUsersForm, UsersForm); //creatr the users form
UsersForm.Show;
end;
2 :
begin
Panel1.Hide;
Application.CreateForm(TUsersForm, UsersForm);
UsersForm.Show;
UsersForm.BitBtn6.Enabled:=false; //case2 user cant see the delet button
```

```
end;
end;
end;
procedure TMainForm.LessonsData1Click(Sender: TObject);
begin
Panel1.Hide;
Application.CreateForm(TLessonsDataForm, LessonsDataForm);
LessonsDataForm.Show;
case UserGroup of
2 : LessonsDataForm.BitBtn6.Enabled:=false; //cant delet, again
3 :
begin
LessonsDataForm.BitBtn4.Visible:=false;// cant do anythin but search, again
LessonsDataForm.BitBtn5.Visible:=false;
LessonsDataForm.BitBtn6.Visible:=false;
end;
end;
end;
end.
```

1.4 THE MAIN MENU UNITS CODES

1.4.1 Users UNIT

```
unit UsersUnit;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, ComCtrls, StdCtrls, Buttons, DB, ADODB, Mask, DBCtrls,

Grids, DBGrids;

type

TUsersForm = class(TForm)

StatusBar1: TStatusBar;

Panel1: TPanel;

Panel2: TPanel;

Panel3: TPanel;

BitBtn1: TBitBtn;

ADODataSet1: TADODataSet;

DataSource1: TDataSource;

BitBtn4: TBitBtn;

BitBtn5: TBitBtn;

BitBtn6: TBitBtn;

Panel4: TPanel;
```

```
BitBtn2: TBitBtn;
BitBtn3: TBitBtn;
BitBtn7: TBitBtn;
BitBtn8: TBitBtn;
ADODataSet1USER_ID: TAutoIncField;
ADODataSet1USER_NAME: TWideStringField;
ADODataSet1USER_SURNAME: TWideStringField;
ADODataSet1USER_ADDRESS: TWideStringField;
ADODataSet1USER_TEL: TWideStringField;
ADODataSet1USER_EMAIL: TWideStringField;
ADODataSet1USER_GROUP: TIntegerField;
ADODataSet1LOGIN_NAME: TWideStringField;
ADODataSet1LOGIN_PASSWORD: TWideStringField;
Label1: TLabel;
DBEdit1: TDBEdit;
Label2: TLabel;
DBEdit2: TDBEdit;
Label3: TLabel;
DBEdit3: TDBEdit;
Label4: TLabel;
DBEdit4: TDBEdit;
```

```
Label5: TLabel;
DBEdit5: TDBEdit;
Label6: TLabel;
DBEdit6: TDBEdit;
Label7: TLabel;
DBEdit7: TDBEdit;
Label8: TLabel;
DBEdit8: TDBEdit;
Label9: TLabel;
DBEdit9: TDBEdit;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn3Click(Sender: TObject);
procedure BitBtn4Click(Sender: TObject);
procedure BitBtn5Click(Sender: TObject);
procedure BitBtn6Click(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn8Click(Sender: TObject);
procedure BitBtn7Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
```

```
{ Private declarations }
public
{ Public declarations }
end;
var
UsersForm: TUsersForm;
implementation
uses MainUnit;
{$R *.dfm}
procedure TUsersForm.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;
procedure TUsersForm.BitBtn2Click(Sender: TObject);
begin
ADODataSet1.Next;
end;
procedure TUsersForm.BitBtn3Click(Sender: TObject);
begin
ADODataSet1.Prior;
```

```
end;
procedure TUsersForm.BitBtn4Click(Sender: TObject);
begin
ADODataSet1.Insert;
end;
procedure TUsersForm.BitBtn5Click(Sender: TObject);
begin
ADODataSet1.Post;
end;
procedure TUsersForm.BitBtn6Click(Sender: TObject);
begin
ADODataSet1.Delete;
end;
procedure TUsersForm.BitBtn1Click(Sender: TObject);
begin
Close;
MainForm.Panel1.Show;
MainForm.FormShow(Sender);
end;
procedure TUsersForm.BitBtn8Click(Sender: TObject);
begin
```

```
ADODataSet1.First;
end;
procedure TUsersForm.BitBtn7Click(Sender: TObject);
begin
ADODataSet1.Last;
end;
procedure TUsersForm.FormCreate(Sender: TObject);
begin
ADODataSet1.Open
end;
end.
```

1.4.2 Courses UNIT:

```
unit CoursesUnit;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, ComCtrls, StdCtrls, Buttons, DB, ADODB, Mask, DBCtrls,
Grids, DBGrids;
type
TCoursesForm = class(TForm)
StatusBar1: TStatusBar;
```

Panel1: TPanel;

Panel2: TPanel;

Panel3: TPanel;

BitBtn1: TBitBtn;

ADODataSet1: TADODataSet;

Label1: TLabel;

DBEdit1: TDBEdit;

DataSource1: TDataSource;

Label2: TLabel;

DBEdit2: TDBEdit;

BitBtn4: TBitBtn;

BitBtn5: TBitBtn;

BitBtn6: TBitBtn;

Panel4: TPanel;

BitBtn2: TBitBtn;

BitBtn3: TBitBtn;

BitBtn7: TBitBtn;

BitBtn8: TBitBtn;

Label3: TLabel;

DBEdit3: TDBEdit;

ADODataSet1COURSE_CODE: TAutoIncField;

```
ADODataSet1COURSE_TITLE: TWideStringField;
ADODataSet1COURSE_LEVEL: TIntegerField;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn3Click(Sender: TObject);
procedure BitBtn4Click(Sender: TObject);
procedure BitBtn5Click(Sender: TObject);
procedure BitBtn6Click(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn8Click(Sender: TObject);
procedure BitBtn7Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
CoursesForm: TCoursesForm;
implementation
uses MainUnit;
{$R *.dfm}
```

```
procedure TCoursesForm.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;
procedure TCoursesForm.BitBtn2Click(Sender: TObject);
begin
ADODataSet1.Next;
end;
procedure TCoursesForm.BitBtn3Click(Sender: TObject);
begin
ADODataSet1.Prior;
end;
procedure TCoursesForm.BitBtn4Click(Sender: TObject);
begin
ADODataSet1.Insert;
end;
procedure TCoursesForm.BitBtn5Click(Sender: TObject);
begin
ADODataSet1.Post;
end;
```

```
procedure TCoursesForm.BitBtn6Click(Sender: TObject);
begin
ADODataSet1.Delete;
end;
procedure TCoursesForm.BitBtn1Click(Sender: TObject);
begin
Close;
MainForm.Panel1.Show;
MainForm.FormShow(Sender);
end;
procedure TCoursesForm.BitBtn8Click(Sender: TObject);
begin
ADODataSet1.First;
end;
procedure TCoursesForm.BitBtn7Click(Sender: TObject);
begin
ADODataSet1.Last;
end;
end.
```

1.4.3 Lessons UNIT:

```
unit LessonsUnit;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, ComCtrls, StdCtrls, Buttons, DB, ADODB, Mask, DBCtrls,

Grids, DBGrids;

type

TLessonsForm = class(TForm)

StatusBar1: TStatusBar;

Panel1: TPanel;

Panel2: TPanel;

Panel3: TPanel;

BitBtn1: TBitBtn;

ADODataSet1: TADODataSet;

DataSource1: TDataSource;

BitBtn4: TBitBtn;

BitBtn5: TBitBtn;

BitBtn6: TBitBtn;

Panel4: TPanel;

BitBtn2: TBitBtn;
```

```
BitBtn3: TBitBtn;
BitBtn7: TBitBtn;
BitBtn8: TBitBtn;
ADODataSet1LESSON_CODE: TAutoIncField;
ADODataSet1LESSON_COURSE: TIntegerField;
ADODataSet1LESSON_TITLE: TWideStringField;
ADODataSet1LESSON_INDEX: TIntegerField;
ADODataSet1LESSON_TYPE: TIntegerField;
Label1: TLabel;
DBEdit1: TDBEdit;
Label2: TLabel;
DBEdit2: TDBEdit;
Label3: TLabel;
DBEdit3: TDBEdit;
Label4: TLabel;
DBEdit4: TDBEdit;
Label5: TLabel;
DBEdit5: TDBEdit;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn3Click(Sender: TObject);
```

```
procedure BitBtn4Click(Sender: TObject);
procedure BitBtn5Click(Sender: TObject);
procedure BitBtn6Click(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn8Click(Sender: TObject);
procedure BitBtn7Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
LessonsForm: TLessonsForm;
implementation
uses MainUnit;
{$R *.dfm}
procedure TLessonsForm.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;
```

```
procedure TLessonsForm.BitBtn2Click(Sender: TObject);
begin
ADODataSet1.Next;
end;
procedure TLessonsForm.BitBtn3Click(Sender: TObject);
begin
ADODataSet1.Prior;
end;
procedure TLessonsForm.BitBtn4Click(Sender: TObject);
begin
ADODataSet1.Insert;
end;
procedure TLessonsForm.BitBtn5Click(Sender: TObject);
begin
ADODataSet1.Post;
end;
procedure TLessonsForm.BitBtn6Click(Sender: TObject);
begin
ADODataSet1.Delete;
end;
procedure TLessonsForm.BitBtn1Click(Sender: TObject);
```

```
begin
Close;
MainForm.Panel1.Show;
MainForm.FormShow(Sender);
end;
procedure TLessonsForm.BitBtn8Click(Sender: TObject);
begin
ADODataSet1.First;
end;
procedure TLessonsForm.BitBtn7Click(Sender: TObject);
begin
ADODataSet1.Last;
end;
end.
```

1.4.4 LessonsData UNIT:

```
unit LessonsDataUnit;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, ComCtrls, StdCtrls, Buttons, DB, ADODB, Mask, DBCtrls,
Grids, DBGrids;
```

```
type
TLessonsDataForm = class(TForm)
StatusBar1: TStatusBar;
Panel1: TPanel;
Panel2: TPanel;
Panel3: TPanel;
BitBtn1: TBitBtn;
ADODataSet1: TADODataSet;
DataSource1: TDataSource;
BitBtn4: TBitBtn;
BitBtn5: TBitBtn;
BitBtn6: TBitBtn;
Panel4: TPanel;
BitBtn2: TBitBtn;
BitBtn3: TBitBtn;
BitBtn7: TBitBtn;
BitBtn8: TBitBtn;
ADODataSet1DATA_LESSON: TIntegerField;
ADODataSet1DATA_TITLE: TWideStringField;
ADODataSet1DATA_IMAGE: TWideStringField;
ADODataSet1DATA_INDEX: TIntegerField;
```

```
Label1: TLabel;
DBEdit1: TDBEdit;
Label2: TLabel;
DBEdit2: TDBEdit;
Label3: TLabel;
DBEdit3: TDBEdit;
Label4: TLabel;
DBEdit4: TDBEdit;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn3Click(Sender: TObject);
procedure BitBtn4Click(Sender: TObject);
procedure BitBtn5Click(Sender: TObject);
procedure BitBtn6Click(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn8Click(Sender: TObject);
procedure BitBtn7Click(Sender: TObject);
private
{ Private declarations }
Public
{ Public declarations }
```

```
end;
var
LessonsDataForm: TLessonsDataForm;
implementation
uses MainUnit;
{$R *.dfm}
procedure TLessonsDataForm.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;
procedure TLessonsDataForm.BitBtn2Click(Sender: TObject);
begin
ADODataSet1.Next;
end;
procedure TLessonsDataForm.BitBtn3Click(Sender: TObject);
begin
ADODataSet1.Prior;
end;
procedure TLessonsDataForm.BitBtn4Click(Sender: TObject);
begin
```

```
ADODataSet1.Insert;
end;
procedure TLessonsDataForm.BitBtn5Click(Sender: TObject);
begin
ADODataSet1.Post;
end;
procedure TLessonsDataForm.BitBtn6Click(Sender: TObject);
begin
ADODataSet1.Delete;
end;
procedure TLessonsDataForm.BitBtn1Click(Sender: TObject);
begin
Close;
MainForm.Panel1.Show;
MainForm.FormShow(Sender);
end;
procedure TLessonsDataForm.BitBtn8Click(Sender: TObject);
begin
ADODataSet1.First;
end;
procedure TLessonsDataForm.BitBtn7Click(Sender: TObject);
```

```
begin
ADODataSet1.Last;
end;
end.
```

1.5 LESSONS' UNITS CODES

1.5.1 Lesson1 UNIT:

```
unit Lesson1Unit;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, StdCtrls, jpeg, Buttons, DB, ADODB;
type
TLesson1Form = class(TForm)
Panel1: TPanel;
Panel3: TPanel;
Image1: TImage;
Panel2: TPanel;
Label1: TLabel;
Label2: TLabel;
StartFinish: TBitBtn;
Panel4: TPanel;
```

```
Panel5: TPanel;
Back: TBitBtn;
Next: TBitBtn;
Bevel1: TBevel;
ADODataSetData: TADODataSet;
Panel6: TPanel;
Timer1: TTimer;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure StartFinishClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure NextClick(Sender: TObject);
procedure BackClick(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Lesson1Form: TLesson1Form;
implementation
```

```
uses MainUnit;

{$R *.dfm}

procedure TLesson1Form.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

Action:=caFree;

end;

procedure TLesson1Form.StartFinishClick(Sender: TObject);

begin

if StartFinish.Caption='Finish' then

begin

MainForm.Panel1.Visible:=true;

Close;

end

else

begin

Back.Visible:=false;

Timer1.Enabled:=true;

Panel6.Visible:=FALSE;

Panel2.Visible:=true;

Panel3.Visible:=true;
```

```
Next.Visible:=True;

StartFinish.Caption:='Finish';

StartFinish.Glyph.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\button
s\finish.bmp');    Application.ProcessMessages;

end;

end;

procedure TLesson1Form.FormCreate(Sender: TObject);

begin

StartFinish.Caption:='Start';

StartFinish.Glyph.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\button
s\forward.bmp');
Panel1.Caption:=MainForm.ADODataSet2.FieldByName('LESSON_TITLE').AsString;
ADODataSetData.Close;

ADODataSetData.CommandText:='SELECT * FROM LESSONS_DATA WHERE
DATA_LESSON='+MainForm.ADODataSet2.FieldByName('LESSON_CODE').AsString
+' ORDER BY DATA_INDEX ASC';

ADODataSetData.Open;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);//definition of the db to the adod
set ,bring it from the near place

Label1.Caption:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,1,1);

Label2.Caption:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,2,lengt
h(ADODataSetData.FieldByName('DATA_TITLE').AsString));
```

```
Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

Panel6.Align:=alClient;

end;

procedure TLesson1Form.NextClick(Sender: TObject);      //in the NEXT BUTTON
situation

begin

ADODataSetData.Next;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString); //bring the picture from data image
FROM THE DB

Label1.Caption:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,1,1);

Label2.Caption:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,2,lengt
h(ADODataSetData.FieldByName('DATA_TITLE').AsString));

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

IF ADODataSetData.RecNo=ADODataSetData.RecordCount then

Next.Visible:=false

else

Next.Visible:=true;

IF ADODataSetData.RecNo=1 then

Back.Visible:=false

else
```

```
Back.Visible:=true;

end;

procedure TLesson1Form.BackClick(Sender: TObject);

begin

ADODataSetData.Prior;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);

Label1.Caption:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,1,1);

Label2.Caption:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,2,lengt
h(ADODataSetData.FieldByName('DATA_TITLE').AsString));

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

IF ADODataSetData.RecNo=1 then

Back.Visible:=false

else

Back.Visible:=true;

IF ADODataSetData.Eof then

Next.Visible:=false

else

Next.Visible:=true;

end;

procedure TLesson1Form.Timer1Timer(Sender: TObject);
```

```
var
clr : TColor;
begin
if Label1.Font.Color=clGreen then
Label1.Font.Color:=clRed
else
if Label1.Font.Color=clRed then
Label1.Font.Color:=clGreen
//else
//if Label1.Font.Color=clGreen then
//  Label1.Font.Color:=clBlack
else
Label1.Font.Color:=clGreen;
//Application.ProcessMessages;
end;
end.
```

1.5.2 Lesson2 UNIT:

```
unit Lesson2Unit;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
```

```
Dialogs, ExtCtrls, StdCtrls, jpeg, Buttons, DB, ADODB;

type

TLesson2Form = class(TForm)

Panel1: TPanel;

Panel3: TPanel;

Image1: TImage;

Panel2: TPanel;

Label2: TLabel;

StartFinish: TBitBtn;

Panel4: TPanel;

Panel5: TPanel;

Back: TBitBtn;

Next: TBitBtn;

Bevel1: TBevel;

ADODataSetData: TADODataSet;

Panel6: TPanel;

Edit1: TEdit;

Timer1: TTimer;

Label1: TLabel;

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure StartFinishClick(Sender: TObject);
```

```
procedure FormCreate(Sender: TObject);
procedure NextClick(Sender: TObject);
procedure BackClick(Sender: TObject);
procedure Edit1Change(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Lesson2Form: TLesson2Form;
implementation
uses MainUnit;
{$R *.dfm}
procedure TLesson2Form.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;
procedure TLesson2Form.StartFinishClick(Sender: TObject);
begin
```

```
if StartFinish.Caption='Finish' then

begin

MainForm.Panel1.Visible:=true;

Close;

end

else

begin

Back.Visible:=false;

Timer1.Enabled:=true;

Panel6.Visible:=FALSE;

Panel2.Visible:=true;

Panel3.Visible:=true;

Next.Visible:=True;

StartFinish.Caption:='Finish';

StartFinish.Glyph.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\button
s\finish.bmp');

Application.ProcessMessages;

Label1.Visible:=false;

Edit1.SetFocus;

end;

end;
```

```
procedure TLesson2Form.FormCreate(Sender: TObject);

begin

StartFinish.Caption:='Start';

StartFinish.Glyph.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\button
s\forward.bmp');

Panel1.Caption:=MainForm.ADODataSet2.FieldByName('LESSON_TITLE').AsString;

ADODataSetData.Close;

ADODataSetData.CommandText:='SELECT * FROM LESSONS_DATA WHERE
DATA_LESSON='+MainForm.ADODataSet2.FieldByName('LESSON_CODE').AsString
+' ORDER BY DATA_INDEX ASC';

ADODataSetData.Open;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);

Edit1.Hint:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,1,1);

Label2.Caption:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,2,lengt
h(ADODataSetData.FieldByName('DATA_TITLE').AsString));

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

Panel6.Align:=alClient;

end;

procedure TLesson2Form.NextClick(Sender: TObject);

begin

ADODataSetData.Next;
```

```
Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);

Edit1.Hint:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,1,1);

Label2.Caption:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,2,lengt
h(ADODataSetData.FieldByName('DATA_TITLE').AsString));

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

IF ADODataSetData.RecNo=ADODataSetData.RecordCount then

Next.Visible:=false

else

Next.Visible:=true;

IF ADODataSetData.RecNo=1 then

Back.Visible:=false

else

Back.Visible:=true;

Edit1.Text:='?';

Label1.Visible:=false;

Edit1.SetFocus;

end;

procedure TLesson2Form.BackClick(Sender: TObject);

begin

ADODataSetData.Prior;
```

```
Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);

Edit1.Hint:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,1,1);

Label2.Caption:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,2,lengt
h(ADODataSetData.FieldByName('DATA_TITLE').AsString));

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

IF ADODataSetData.RecNo=1 then

Back.Visible:=false

else

Back.Visible:=true;

IF ADODataSetData.Eof then

Next.Visible:=false

else

Next.Visible:=true;

Edit1.Text:='?';

Label1.Visible:=false;

Edit1.SetFocus;

end;

procedure TLesson2Form.Edit1Change(Sender: TObject);

begin

Label1.Visible:=true;
```

```
if Edit1.Text=Edit1.Hint then
BEGIN
Edit1.Color:=clGreen;
Label1.Caption:='P';
Label1.Font.Color:=clLime;
end
else
BEGIN
Edit1.Color:=clRed;
Label1.Caption:='O';
Label1.Font.Color:=clRed;
end
end;
end.
```

1.5.3 Lesson3 UNIT:

```
unit Lesson3Unit;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, StdCtrls, jpeg, Buttons, DB, ADODB;
type
```

```
TLesson3Form = class(TForm)
Panel1: TPanel;
Panel3: TPanel;
Image1: TImage;
Panel2: TPanel;
StartFinish: TBitBtn;
Panel4: TPanel;
Panel5: TPanel;
Back: TBitBtn;
Next: TBitBtn;
Bevel1: TBevel;
ADODataSetData: TADODataSet;
Panel6: TPanel;
Timer1: TTimer;
Memo1: TMemo;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure StartFinishClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure NextClick(Sender: TObject);
procedure BackClick(Sender: TObject);
private
```

```
{ Private declarations }
public
{ Public declarations }
end;
var
Lesson3Form: TLesson3Form;
implementation
uses MainUnit;
{$R *.dfm}
procedure TLesson3Form.FormClose(Sender: TObject;
var Action: TCloseAction);
begin
Action:=caFree;
end;
procedure TLesson3Form.StartFinishClick(Sender: TObject);
begin
if StartFinish.Caption='Finish' then
begin
MainForm.Panel1.Visible:=true;
Close;
end
```

```
else
begin
Back.Visible:=false;
Timer1.Enabled:=true;
Panel6.Visible:=FALSE;
Panel2.Visible:=true;
Panel3.Visible:=true;
Next.Visible:=True;
StartFinish.Caption:='Finish';
StartFinish.Glyph.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\button
s\finish.bmp');
Application.ProcessMessages;
end;
end;
procedure TLesson3Form.FormCreate(Sender: TObject);
begin
StartFinish.Caption:='Start';
StartFinish.Glyph.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\button
s\forward.bmp'); //bring the pic
Panel1.Caption:=MainForm.ADODataSet2.FieldByName('LESSON_TITLE').AsString;
ADODataSetData.Close;          ADODataSetData.CommandText:='SELECT * FROM
LESSONS_DATA                                                    WHERE
```

```
DATA_LESSON='+MainForm.ADODataSet2.FieldByName('LESSON_CODE').AsString
+' ORDER BY DATA_INDEX ASC';

ADODataSetData.Open;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);

MEMO1.Text:=ADODataSetData.FieldByName('DATA_TITLE').AsString;

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

Panel6.Align:=alClient;

end;

procedure TLesson3Form.NextClick(Sender: TObject);

begin

ADODataSetData.Next;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);
MEMO1.Text:=ADODataSetData.FieldByName('DATA_TITLE').AsString;// bring it from
the data tit

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

IF ADODataSetData.RecNo=ADODataSetData.RecordCount then

Next.Visible:=false

else

Next.Visible:=true;
```

```
IF ADODataSetData.RecNo=1 then

Back.Visible:=false

else

Back.Visible:=true;

end;

procedure TLesson3Form.BackClick(Sender: TObject);

begin

ADODataSetData.Prior;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);

MEMO1.Text:=Copy(ADODataSetData.FieldByName('DATA_TITLE').AsString,2,length(
ADODataSetData.FieldByName('DATA_TITLE').AsString));

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

IF ADODataSetData.RecNo=1 then

Back.Visible:=false

else

Back.Visible:=true;

IF ADODataSetData.Eof then

Next.Visible:=false

else

Next.Visible:=true;
```

```
end;

end.
```

1.5.4 Lesson4 UNIT:

```
unit Lesson4Unit;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, StdCtrls, jpeg, Buttons, DB, ADODB;

type

TLesson4Form = class(TForm)

Panel1: TPanel;

Panel3: TPanel;

Image1: TImage;

Panel2: TPanel;

StartFinish: TBitBtn;

Panel4: TPanel;

Panel5: TPanel;

Back: TBitBtn;

Next: TBitBtn;

Bevel1: TBevel;

ADODataSetData: TADODataSet;
```

```
Panel6: TPanel;
Edit1: TEdit;
Timer1: TTimer;
Label1: TLabel;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure StartFinishClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure NextClick(Sender: TObject);
procedure BackClick(Sender: TObject);
procedure Edit1Change(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Lesson4Form: TLesson4Form;
implementation
uses MainUnit;
{$R *.dfm}
procedure TLesson4Form.FormClose(Sender: TObject;
```

```
var Action: TCloseAction);
begin
Action:=caFree;
end;
procedure TLesson4Form.StartFinishClick(Sender: TObject);
begin
if StartFinish.Caption='Finish' then
begin
MainForm.Panel1.Visible:=true;
Close;
end
else
begin
Back.Visible:=false;
Timer1.Enabled:=true;
Panel6.Visible:=FALSE;
Panel2.Visible:=true;
Panel3.Visible:=true;
Next.Visible:=True;
StartFinish.Caption:='Finish';
```

```
StartFinish.Glyph.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\button
s\finish.bmp');

Application.ProcessMessages;

Label1.Visible:=false;

Edit1.SetFocus;

end;

end;

procedure TLesson4Form.FormCreate(Sender: TObject);

begin

StartFinish.Caption:='Start';

StartFinish.Glyph.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\button
s\forward.bmp');

Panel1.Caption:=MainForm.ADODataSet2.FieldByName('LESSON_TITLE').AsString;

ADODataSetData.Close;

ADODataSetData.CommandText:='SELECT * FROM LESSONS_DATA WHERE
DATA_LESSON='+MainForm.ADODataSet2.FieldByName('LESSON_CODE').AsString
+' ORDER BY DATA_INDEX ASC'; //bring the data according tothe d index order

ADODataSetData.Open;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);

Edit1.Hint:=ADODataSetData.FieldByName('DATA_TITLE').AsString;

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;
```

```
Panel6.Align:=alClient;

end;

procedure TLesson4Form.NextClick(Sender: TObject);

begin

ADODataSetData.Next;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);

Edit1.Hint:=ADODataSetData.FieldByName('DATA_TITLE').AsString;

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

IF ADODataSetData.RecNo=ADODataSetData.RecordCount then

Next.Visible:=false

else

Next.Visible:=true;

IF ADODataSetData.RecNo=1 then

Back.Visible:=false

else

Back.Visible:=true;

Edit1.Text:='?';

Label1.Visible:=false;

Edit1.SetFocus;
```

```
end;

procedure TLesson4Form.BackClick(Sender: TObject);

begin

ADODataSetData.Prior;

Image1.Picture.LoadFromFile(ExtractFilePath(Application.ExeName)+'images\'+ADOD
ataSetData.FieldByName('DATA_IMAGE').AsString);

Edit1.Hint:=ADODataSetData.FieldByName('DATA_TITLE').AsString;

Panel6.Caption:='LESSON
'+MainForm.ADODataSet2.FieldByName('LESSON_INDEX').AsString;

IF ADODataSetData.RecNo=1 then

Back.Visible:=false

else

Back.Visible:=true;

IF ADODataSetData.Eof then

Next.Visible:=false

else

Next.Visible:=true;

Edit1.Text:='?';

Label1.Visible:=false;

Edit1.SetFocus;

end;
```

```
procedure TLesson4Form.Edit1Change(Sender: TObject);
begin
Label1.Visible:=true;
if Edit1.Text=Edit1.Hint then
BEGIN
Edit1.Color:=clGreen;
Label1.Caption:='P';
Label1.Font.Color:=clLime;
end
else
BEGIN
Edit1.Color:=clRed;
Label1.Caption:='O';
Label1.Font.Color:=clRed;
end
end;
end.
```