Coursework – Speech

|  |  |
| --- | --- |
| Slide # |  |

1. The topic of my coursework is graph visual editing. So I had to develop an application which is in fact a visual graph editor.
2. There’s the list of requirements on this slide (#2).  
   The application should have the very basic features such as add nodes, remove nodes, move nodes, specify caption for the node, draw links between nodes and so on. The possibility of storing documents in files on of the important requirements.
3. The application is written in C# programming language. So it’s based on the Microsoft .NET framework. I used Microsoft Visual Studio Express 2012 integrated development environment. Since it’s a visual editor, user interface is an important part of this application, so to build it I used Windows Presentation Foundation framework.
4. I also used git source code management system with a graphical frontend and I stored my git repositories online on GitHub hosting service. And I used NLog logging library to debug my application.
5. Here’s how the application window looks. You see some document displayed. It contains several nodes and links. All operations are accessible through the main menu and context menus of the nodes.
6. One of the important features of the application is the ability to store documents in XML format so they can be easily accessed, for example, from other applications. These documents have the following format: in the root “content” element there are NodeControl and LinkControl elements. Each represents a node or a link respectively. NodeControl elements have text and position attributes while each LinkControl element has two attributes which store unique IDs of the nodes which are interconnected by particular link.
7. While working on this project I got to use Windows Presentation Foundation framework. So I wanted to mention some of its features. It supports hardware acceleration and it is resolution independent. It also supports styles and it’s actually possible to create really good-looking interfaces with it.
8. One of the key features of the Windows Presentation Foundation is separating user interface design from the application source code responsible for application logic. The user interface should be specified in XAML, an xml-like markup language. Here’s an example form my application. It’s the part of the main window.
9. Windows presentation foundation introduces dependency properties and provides means for connecting properties of different objects like, for example, some data and a window control which displays this data. This concept is called Data binding and updating properties is not its only feature. Using dependency properties makes the application more complex to some point.
10. There are also two notable features of the C# language which are still relatively new but I believe they are well-known for C# developers. Linq is a set of special features for searching lists and processing various data. For example, this expression searches for links connected to the node. And another feature is lambda expression. Which is, in fact, an anonymous function which can be constructed inside another function and passed to other functions as a parameter so they may call later it when necessary. This code is also taken from my project and it sets dependency property value for every element of the list which matches specified type.
11. This brings me to the end of my presentation

<Window x:Class="Coursework\_2.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

>

<!-- ... -->

<Window.Resources>

<Style x:Key="NodeBorderStyle" TargetType="Border">

<Setter Property="BorderThickness" Value="1.5" />

<Setter Property="BorderBrush" Value="DarkGray" />

</Style>

<Image x:Key="Image\_NewItem" Source="Images/NewItemBlack16.png" />

<!-- ... -->

<Image x:Key="Image\_DrawLink" Source="Images/LinkBlack16.png" />

</Window.Resources>

<DockPanel>

<Menu DockPanel.Dock="Top" UseLayoutRounding="True" SnapsToDevicePixels="True">

<MenuItem Header="File" Icon="{StaticResource Image\_File}">

<MenuItem Header="Open file..." x:Name="OpenFileMenuItem" Icon="{StaticResource Image\_OpenFile}" />

<!-- ... -->