в MS Mathematics схожою на обчислення з калькулятором, лише інтелектуальним. Тому її використання не вимагатиме суттєво змінювати зміст і методику викладання. Цю програму можна успішно застосовувати для навчання лінійній алгебрі, диференціальному та інтегральному численню функцій однієї та кількох змінних.

ЛІТЕРАТУРА

1. Microsoft Download Center. Microsoft Mathematics 4.0 [Електронний ресурс]. – Режим доступу: http://www.microsoft.com/ru-ru/download/details.aspx?id=15702. – Назва з екрану.
2. Зюков М. Е. Обучение высшей математике с использованием Microsoft Mathematics / Зюков М. Е. // Вісник Луганського національного університету імені Тараса Шевченко (педагогічні науки) – 2013, № 20 (279). – С. 67-72. – ISSN 2227-2844.
3. Беклемишев Д. В. Курс аналитической геометрии и линейной алгебры: Учеб. для вузов / Д. В. Беклемишев. – 11-е изд., испр. – М.: ФИЗМАТЛИТ, 2006. – 312 с. – ISBN 5-9221-0691-0.
4. Ильин В. А. Линейная алгебра и аналитическая геометрия: Учеб. для вузов / В. А. Ильин, Г. Д. Ким. – М.: Изд-во Моск. ун-та, 1998. – 320 с. – ISBN 5-211-03814-2.
5. Wolfram Language & Sysem. Documantation Center. RowReduce \ Properties & Relations [Електронний ресурс]. – Режим доступу: http://reference.wolfram.com/language/ref/RowReduce.html. – Назва з екрану.
6. Александров П. С. Курс аналитической геометрии и линейной алгебры: Учеб. для вузов / П. С. Александров. – М.: Наука, Гл. ред. физ.-мат. лит., 1979. – 512 с.
7. Бронштейн И. Н. Справочник по математике для инженеров и учащихся втузов / И. Н. Бронштейн, К. А. Семендяев. – 13-е изд., исправленное. – М.: Наука, Гл. ред. физ.-мат. лит., 1986. – 544 с.

УДК 004.5

СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ЭЛЕМЕНТА В ТЕХНОЛОГИИ WPF

Н.В. Карпенко[1], А.А. Доброгорский[2]

[1]кандидат физико-математических наук, доцент кафедры электронных вычислительных машин, Днепропетровский национальный университет имени Олеся Гончара, г. Днепропетровск, Украина, e-mail: karpenko_nadija@mail.ru

[2]студент группы КІ-12У-1, кафедра электронных вычислительных машин, Днепропетровский национальный университет имени Олеся Гончара, г. Днепропетровск, Украина, e-mail: zore3x@gmail.com

Аннотация. В статье рассмотрены различные варианты создания пользовательского элемента управления (кнопки-образа) с использованием технологии Windows Presentation Foundation.

Ключевые слова: WPF, интерфейс, пользовательский элемент, styles, templates.

CREATING CUSTOM ELEMENTS IN WPF TECHNOLOGY

Nadiia Karpenko[1], Anton Dobrogorskyi[2]
[1]Ph.D. in Physics and Mathematical Sciences, Associate Professor, Department of Computer, Dnipropetrovs'k National University named by Oles Honchar, Dnipropetrovs'k, Ukraine, e-mail: karpenko_nadija@mail.ru
[2]Student, Department of Computer, Dnipropetrovs'k National University named by Oles Honchar, Dnipropetrovs'k, Ukraine, e-mail: zore3x@gmail.com

Abstract. The article describes the various options for creating a custom control (buttons image) using the technology Windows Presentation Foundation.

Keywords: WPF, interface, custom control, styles, templates.

Введение. Любое взаимодействие пользователя с системой происходит через интерфейс, и чем удобнее и понятнее он будет, тем работа с приложением будет эффективнее. При создании приложения обычно рекомендуют не только придерживаться стандартов разработки интерфейсов [1], но и учитывать психологические особенности целевой аудитории, т.е. тех людей, которые будут взаимодействовать с системой. Если в качестве целевой аудитории выступают дети дошкольного возраста, то интерфейс, чаще всего, оформляют с использованием цвета, форм, звуков и движений, с минимальным содержанием текста или вообще без него. В данном случае, наиболее приемлемым интерфейсом является тот, в котором в качестве элементов управления используются знакомые понятия, образы и ассоциации, что делает интерфейс интуитивно понятным и легким для освоения.

В настоящее время наиболее перспективной технологией для создания как автономных, так и запускаемых в браузере Windows-приложений является Windows Presentation Foundation (WPF). До создания данной технологии разработчикам приходилось делать выбор между гибкостью элементов управления и их удобством. Базовые элементы управления были удобны в использовании, но они имели стандартную визуализацию и практически не давали возможности настраивать их под потребности пользователя. Нестандартный элемент управления приходилось «создавать с нуля», т.е. разработчикам необходимо было не только вручную рисовать элемент, но и реализовать все его функциональные возможности, такие как, например, обработка нажатия клавиш, выделение текста и т.д.

В технологии WPF, благодаря применению стилей и шаблонов, решена проблема настройки элементов управления. Каждый элемент создается в коде .NET, а не упаковывается базовый элемент из API-интерфейса Win32,

как было в Windows Forms. Таким образом, Windows Presentation Foundation предоставляет механизмы, позволяющие настраивать или полностью изменять элементы [2].

Цель работы. При разработке компьютерной игры обучающего характера для детей дошкольного возраста, возникла необходимость в создании кнопок, которые имели бы образ соответствующих объектов, таких как звездочка, зайчик и др. Поэтому целью данной работы является создание такого пользовательского элемента управления, как кнопка-образ.

Материалы и результаты исследования. Технология Windows Presentation Foundation хороша тем, что поставленную задачу можно решить несколькими способами:

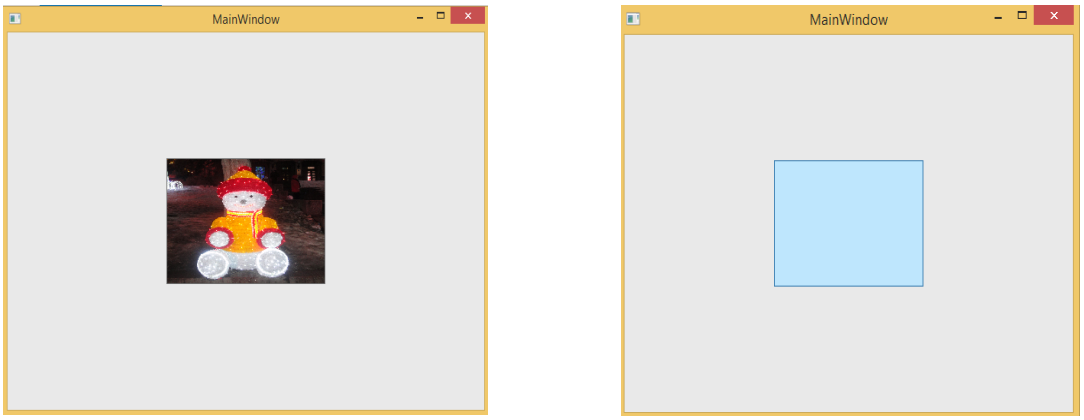
1. вставить в кнопку изображение-фон;
2. вставить в кнопку контейнер компоновки и в нем нарисовать векторное изображение с помощью классов из пространства имен System.Windows.Shapes;
3. создать кнопку соответствующей формы с использованием стилей и шаблонов (наследование от класса System.Windows.Controls.Control) [3].

Рассмотрим каждый из этих способов более подробно.

1) Для заполнения фона окна или элемента управления растровым изображением используется кисть ImageBrush, которая работает с наиболее распространенными типами файлов, включая BMP, PNG, GIF и JPEG. Путь к рисунку определяет свойство ImageSource. Если рисунок находится в папке сборки, то достаточно указать название файла и его тип, т.е. ImageSource="DSC06432.jpg". Например, кнопку, фон которой нарисован с помощью кисти ImageBrush (рис. 1,а), можно получить следующим образом:

```
<Button >
  <Button.Background>
    <ImageBrush ImageSource="E:\2015\DSC06432.jpg"/>
  </Button.Background>
</Button>
```

Недостатком данного способа является то, что при наведении курсора мыши на такую кнопку, ее стандартные настройки приводят к изменению свойства Background (рис. 1, б).



а) б)

Рисунок 1 – Заполнение фона кнопки растровым рисунком

2) Поскольку класс Button является элементом управления содержимым, есть возможность помещать в него не только текст, но и, элементы, и контейнер компоновки, в котором, в свою очередь, также могут находиться элементы или контейнеры. На рис. 2 представлены три кнопки, заполнение которых сделано с использованием вложенных в кнопку элементов и контейнера компоновки. Например, в первую кнопку вложили простую форму Ellipse с градиентной заливкой:

```
<Button Height="200" Width="200" Background="Moccasin"      Mar-
gin="66,50,538,70" Click="Button_Click">
          <Ellipse Width="150" Height="150" >
            <Ellipse.Fill>
              <RadialGradientBrush GradientOrigin="0.25,0.25"  >
                <GradientStop Color="White" Offset="0" />
                <GradientStop Color="Blue"  Offset="1" />
              </RadialGradientBrush>
            </Ellipse.Fill>
          </Ellipse>
     </Button>
```

Вторая кнопка содержит контейнер компоновки Grid, внутри которого находится Polyline:

```
<Button Height="200" Width="200" Background="Moccasin" Mar-
gin="312,50,292,70" Click="Button_Click_1">
    <Grid>
      <Polyline Points="20,10 180,10 20, 90 180,90" Fill="lightpink"
Stroke="red"
StrokeThickness="3"/>
    </Grid>
</Button>
```

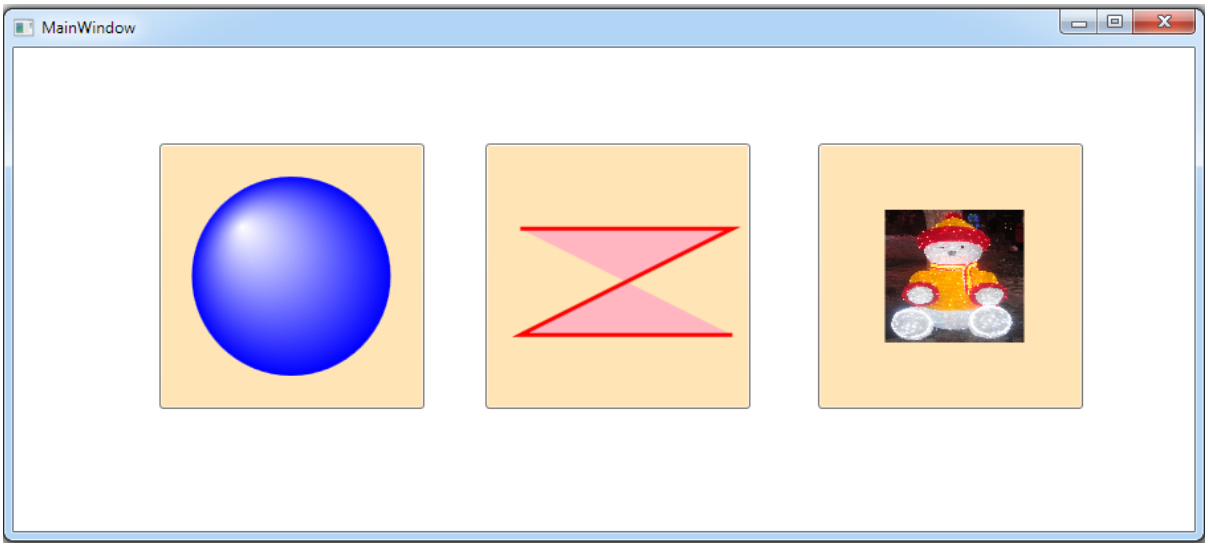


Рисунок 2 – Кнопки с вложенными элементами

В третью кнопку, представленную на рис. 2, вставлен рисунок [4]:

```
<Button Height="200" Width="200" Background="Moccasin" Mar-
gin="563,50,41,70" Click="Button_Click_2">
    <Image Height="100" Source="Resources/DSC064321.jpg" Stretch="Fill"
Margin="48,0,41,0"/>
</Button>
```

Все три кнопки при наведении на них курсора сохраняют изображение.

3) Наследование от класса System.Windows.Controls.Control.

Класс Control определяет элементы управления, которые могут взаимодействовать с пользователем, например, кнопки, списки, текстовые элементы (Button, ListBox, TextBox). Использование шаблонов позволяет вместо стандартного представления элемента управления реализовать пользовательский стиль. Например, создать кнопку в виде звездочки (рис. 3) позволит следующий код:

```
<Window.Resources>
    <Style x:Key="Star" TargetType="{x:Type Button}">
      <Setter Property="Template">
        <Setter.Value>
          <ControlTemplate TargetType="{x:Type Button}">
            <Grid x:Name="Main">
        <Path x:Name="path" Stretch="Fill" Stroke="Black" Data="F1 M 50,10L
35,40L 0,50L 30,60L 20,90L 50,70L 80,90L 70,60L 100,50L 65,40L 50,10 Z ">
              <Path.Fill>
```

```
                    <LinearGradientBrush EndPoint="0.5,1" StartPoint="1,0.5">
                        <GradientStop Color="Gold" Offset="0.3"/>
                        <GradientStop Color="White" Offset="0.8"/>
                      </LinearGradientBrush>
                    </Path.Fill>
                  </Path>
                <ContentPresenter RecognizesAccessKey="True"
                  HorizontalAlignment="{TemplateBinding HorizontalCon-
tentAlignment}"
                  VerticalAlignment="{TemplateBinding VerticalContentAlign-
ment}"/>
              </Grid>
            </ControlTemplate>
          </Setter.Value>
        </Setter>
      </Style>
    </Window.Resources>
```

Для того чтобы воспользоваться данным стилем, необходимо в элементе управления указать ключ: Style="{DynamicResource Star}" или Style="{StaticResource Star}". Аналогичным образом стили можно применять к любому элементу.

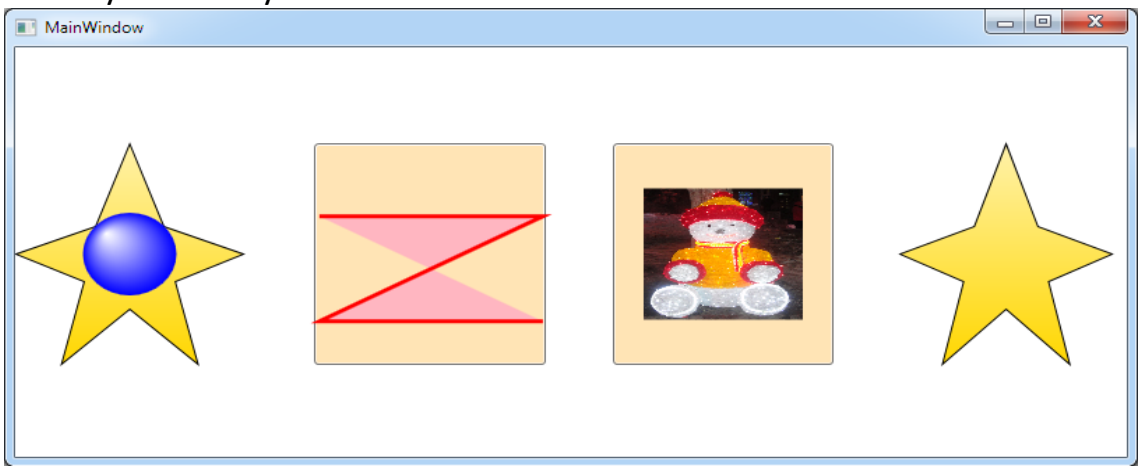


Рисунок 3 – Использование Styles и Template для визуального представления кнопки

Форму кнопки можно изменить с помощью Setter Property="Template". Шаблоны используются только теми элементами, которые наследуются от класса Control, например, кнопка:

ControlTemplate TargetType="{x:Type Button}"

В этом случае свойством Template перекрывается стандартное визуальное представление Control'а. Таким образом, другие свойства созданного пользовательского элемента также необходимо явно прописывать в стиле. Например, ContentPresenter предоставляет возможность для вставки текста или любого другого содержимого.

Использование Template позволяет изменить не только визуальное представление элемента управления, но и задать его поведение при возникновении какого-либо события. Например, на рис. 4 представлена кнопка-звезда, которая реагирует на событие IsMouseOver не изменением Background, а появлением ореола вокруг данной кнопки в виде семи маленьких звездочек:

```
<ControlTemplate.Triggers>
    <Trigger Property="IsMouseOver" Value="True">
        <Setter Property="Opacity" TargetName="miniStar1" Value="1"/>
        <Setter Property="Opacity" TargetName="miniStar2" Value="1"/>
        <Setter Property="Opacity" TargetName="miniStar3" Value="1"/>
        <Setter Property="Opacity" TargetName="miniStar4" Value="1"/>
        <Setter Property="Opacity" TargetName="miniStar5" Value="1"/>
        <Setter Property="Opacity" TargetName="miniStar6" Value="1"/>
        <Setter Property="Opacity" TargetName="miniStar7" Value="1"/>
        <Setter Property="Opacity" TargetName="miniStar8" Value="1"/>
    </Trigger>
</ControlTemplate.Triggers>
```

Рисунок 4 – Использование Template для реализации поведения

Следует отметить, что для исполнения вышеприведенного листинга необходимо предварительно описать все формы с соответствующими названиями (TargetName="miniStar1" и т.д.).

Выводы. Использование стилей и шаблонов технологии WPF значительно упрощает создание элементов управления, ориентированных на конкретного пользователя.

ЛИТЕРАТУРА

1. Методическое пособие по использованию стандартов, обеспечивающих разработку интерфейсов пользователей с операционной средой. [Электронный ресурс]. – Режим доступа: window.edu.ru›resource/919/23919…standards.pdf.— Загл. с экрана.
2. Элементы управления WPF [Электронный ресурс]. – Режим доступа: http://professorweb.ru/my/WPF/UI_WPF/level6/UI_WPF_index.php. — Загл. с экрана.
3. Натан А. WPF 4. Подробное руководство. - Пер. с англ. - СПб.: Символ-Плюс, 2011. - 880 с.
4. Как добавить изображение на кнопку WPF. [Электронный ресурс]. – Режим доступа: http://howtowpf.ru/image-wpf-button/.— Загл. с экрана.

УДК 378.147.111

О ПРОЕКТИРОВАНИИ МЕТОДИЧЕСКОЙ СИСТЕМЫ ОБУЧЕНИЯ В УСЛОВИЯХ ИНФОРМАТИЗАЦИИ ОБУЧЕНИЯ

А.Ю. Лагошный[1], Е.А. Лагошная[2]
[1]старший преподаватель кафедры прикладной математики, Государственное высшее учебное заведение «Приднепровская государственная академия строительства и архитектуры», г. Днепропетровск, Украина, e-mail: alexlagosh@mail.ru
[2]ассистент кафедры автомобили и автомобильное хозяйство, Государственное высшее учебное заведение «Национальный горный университет», г. Днепропетровск, Украина, e-mail: lenala@ua.fm

Аннотация. В настоящее время роль компьютерных технологий в обучении возрастает. В первую очередь это связано с поиском новых, более продуктивных форм и методов обучения. Интерактивные компьютерные модели могут быть использованы в качестве средств обучения, причем для выполнения основных функций в учебном процессе.

Ключевые слова: информация, модель, информационная компьютерная модель, метод.

ON TEACHING METHODOLOGY DEVELOPMENT IN CONDITIONS TEACHING INFORMATIZATION

A. Lagoshny[1], O. Lagoshna[2]
[1]Senior Lecturer of Applied Mathematics Department, State Higher Educational Institution "Dnieper State Academy of Civil Engineering and Architecture", Dnepropetrovsk, Ukraine, e-mail: alexlagosh@mail.ru