Вывод. Была разработана автоматизированная информационная система центра переводов для иностранных публикаций.

Были построены диаграммы IDEF0, IDEF3, DFD, разработаны математические методы решения задач, спроектировано информационное обеспечение. Также были решены все поставленные задачи, разработаны технологический процесс обработки данных и программное обеспечение.

ЛИТЕРАТУРА

1. Суздальцев В.А., Осипова А.Л., Зарайский С.А., Проектирование информационных систем. Учебное пособие. Казань: Изд-во Казан. гос. техн. ун-та, 2007. – 86 с.

УДК 004.415

ОСОБЛИВОСТІ РОЗРОБКИ ДОДАТКУ ЗА ДОПОМОГОЮ МОВИ XAML

Н.В. Карпенко[1], І.А. Гаращук[2], І.І. Зірніченко[3]

[1]кандидат фізико-математичних наук, доцент кафедри ЕОМ, Дніпровський національний університет імені Олеся Гончара, м. Дніпро, Україна, e-mail: karpenko_nadija@mail.ru

[2]студент 5 курсу, кафедра ЕОМ, Дніпровський національний університет імені Олеся Гончара, м. Дніпро, Україна, e-mail: igor_garashchuk@ukr.net

[3]студент 4 курсу, кафедра ЕОМ, Дніпровський національний університет імені Олеся Гончара, м. Дніпро, Україна, e-mail: vanyok789@gmail.com

Анотація. У статті розглянуто особливості розробки додатків за технологією Windows Presentation Foundation. Показано покрокове створення додатку, що містить анімацію, лише використовуючи мову XAML.

Ключові слова: WPF, XAML, анімація елементів інтерфейсу, створення додатку з анімацією.

THE FEATURES OF DEVELOPING APPLICATIONS BY MEANS OF XAML

Nadiia Karpenko[1], Igor Garashchuk[2], Ivan Zirnichenko[3]

[1] Ph.D. in Physics and Mathematical Sciences, Associate Professor, Department of Computer, Dnepr National University named by Oles Honchar, Dnepr, Ukraine, e-mail: karpenko_nadija@mail.ru

[2] 5th year student, Department of Computer, Dnepr National University named by Oles Honchar, Dnepr, Ukraine, e-mail: igor_garashchuk@ukr.net

[3] 4th year student, Department of Computer, Dnepr National University named by Oles Honchar, Dnepr, Ukraine, e-mail: vanyok789@gmail.com

Abstract. The peculiarities of application development technology Windows Presentation Foundation are considered. The incremental build of applications with animation by using only XAML language is shown.

Keywords: WPF, XAML, animated interface elements, creating applications with animation.

Введення: Найважливішим фактором розвитку суспільства в умовах глобальної інформатизації стає високий рівень ефективної системи передачі знань. При правильному використанні комп'ютерні технології надають позитивний ефект у розвитку та навчанні дітей і дорослих. Експериментально доведено, що при грамотному підборі програм та ігор у людини краще розвивається логічне мислення, поліпшується координація очей і рук, стимулюється предметна, образна та зорова пам'ять. Навчальний додаток розвиває спостережливість, вміння концентруватися та виділяти головне, вміння відстежувати декілька процесів одночасно [1, 2]. Тому сучасна освіта, з метою підвищення якості, відкритості та доступності начальних програм, інтенсивно використовує засоби інформаційних і телекомунікаційних технологій. Графічні можливості дисплеїв персональних комп'ютерів і гнучкі мови програмування дозволяють зробити комп'ютерне навчання дуже наочним.

В даний час найбільш перспективною технологією для створення Windows-додатків з насиченим інтерфейсом є технологія Windows Presentation Foundation (WPF) [3]. Завдяки застосуванню стилів і шаблонів у цій технології вирішена проблема налаштування елементів управління. Окрім того, WPF дозволяє підтримувати анімацію, звук, відео та має декларативний інтерфейс користувача, що є її значною перевагою перед іншими технологіями.

Мета роботи: визначити особливості розробки Windows-додатку за допомогою мови XAML.

Матеріали та результати дослідження: Середовищем для розробки Windows-додатків, яке підтримує технологію WPF, є Microsoft Visual Studio. Але можна також використовувати Microsoft Expression Blend, який має WYSIWYG-редактор для дизайну інтерфейсів, основаних на XAML. Це є дуже зручним, оскільки найпростіший додаток можна зробити використовуючи тільки мову XAML. Для цього слід мати чітке уявлення про те, як відбувається компонування інтерфейсу, яким чином встановлюється зв'язок між компонентами та особливості створення анімації.

Розглянемо побудову найпростішого додатку, який відтворює роботу секундоміра. По-перше, слід визначити поле, де буде відбуватись побудова. Для того, щоб створений додаток був адаптований до будь-якого розміру екрану, слід елементи, з яких складається картинка (інтерфейс) упакувати у

Viewbox. Панель Viewbox має лише один дочірній елемент, що може утримувати множину фігур, які при зміні розмірів екрану синхронно змінюють свої розміри.

Стандартними панелями компонування у технології WPF є Grid, DockPanel, WrapPanel, StackPanel та Canvas. Враховуючи особливості кожної панелі, для вирішення поставленої задачі найбільш придатною є Grid. Таким чином, контейнери компонування будуть вкладені один в один, як показано на рис. 1. Причому панель Grid може містити в собі інші панелі компонування. Тому досить зручно розмістити циферблат секундоміра та кнопки управління не тільки у різних комірках Grid, але й у різних панелях компонування, наприклад:

```
<Viewbox>
 <Grid>
 <Grid Grid.Row="0">
 …
 </Grid>
 <Grid Grid.Row="1" >
 ....
 <Grid.ColumnDefinitions>
 <ColumnDefinition></ColumnDefinition>
 <ColumnDefinition></ColumnDefinition>
 <ColumnDefinition></ColumnDefinition>
 </Grid.ColumnDefinitions>
 </Grid>
 <Grid.RowDefinitions>
 <RowDefinition></RowDefinition>
 <RowDefinition></RowDefinition>
 </Grid.RowDefinitions>
 </Grid>
</Viewbox>
```

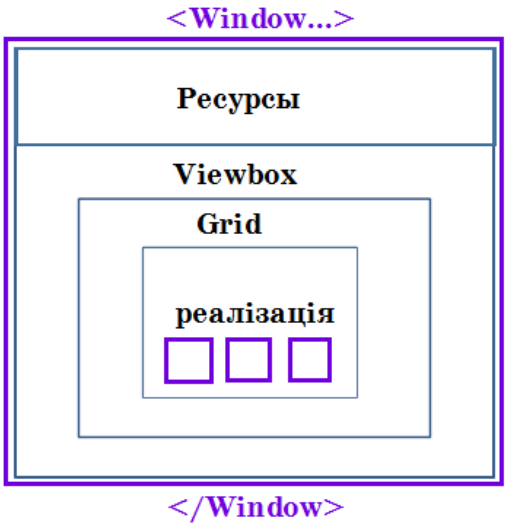

Рисунок 1 – Ілюстрація принципу компонування у технології WPF

Для оформлення циферблату можна самостійно зробити відрисовку стрілок та поділок , як це було зроблено в [4], але найпростішим рішенням є вставлення відповідної картинки, яка заздалегідь буде додана до ресурсів:

```
<Image Name="image1" Stretch="Fill" Source="image1.jpg" Width="400" />
```

Наступним кроком буде нанесення стрілок на циферблат. Нижченаведений код ілюструє відрисовку секундної стрілки та її трансформацію під час виконання анімації. Хвилинна та годинна стрілка мають аналогічний код.

```
<Line Name="Seconds" X1="200" Y1="200" X2="200" Y2="20" Stroke="Red"
                        StrokeThickness="2" RenderTransformOrigin="0.5,0.5">
<Line.RenderTransform>
<RotateTransform x:Name="AnimatedRotateTransform" Angle="0" />
 </Line.RenderTransform>
</Line>
```

Координати X1, Y1, X2, Y2 визначають початкову та кінцеву точки, між якими проведено лінію (стрілка годинника). Так як усі стрілки одним кінцем розташовані в центрі, то розміри поля Width="400" слід поділити навпіл: X1="200" Y1="200". Також стрілки відрізняються довжиною: Y = Y1 - Y2. Отже, для хвилинної та годинної стрілки значення координати Y2 повинно лежати у межах 20 < Y2 <200, наприклад, для хвилинної стрілки Y2="60", а для годинної - Y2="100".

Для того, щоб секундомір розпочав роботу, до кожної стрілки слід прив'язати подію, яка запускатиме виконання анімації. Нехай це буде натискання на кнопку *Start*. Щоб можна було вказати елемент управління, який запускатиме анімацію, йому слід надати ім'я, наприклад Name="Start" (див. вищенаведений код). Окрім того, потрібно вказати:

- ✓ ім'я елемента;
- ✓ властивість, до якої буде застосовано анімацію;
- ✓ кінцеве значення, що набуватиме властивість;
- ✓ тривалість однієї повної анімації;
- ✓ поведінку анімації після закінчення циклу анімації, а саме:

```
<EventTrigger SourceName="Start" RoutedEvent="Button.Click">
 <EventTrigger.Actions>
 <BeginStoryboard Name="Sec">
 <Storyboard>
 <DoubleAnimation Storyboard.TargetName="AnimatedRotateTransform"
 Storyboard.TargetProperty="Angle"
 To="360" Duration="0:1:0" RepeatBehavior="Forever" />
 </Storyboard>
 </BeginStoryboard>
 </EventTrigger.Actions>
 </EventTrigger>
```

Вищенаведений фрагмент коду ілюструє анімацію секундної стрілки. Властивість, до якої застосовано анімацію – кут. Для кожної стрілки він змінюється від 0° до 360°, а той час як тривалість анімації – різна. Для секундної

стрілки тривалість анімації – 1 хвилина, для хвилинної стрілки – 1 година, для годинної стрілки – 12 годин.

Особливістю даної побудови є те, що триггери слід винести окремо, наприклад, створити колекцію Triggers (рис. 2), так як вони не працюють з анімаціями, коли прикріплені до самого елементу [5].

Поставлена задача передбачає, що користувач може не тільки запустити секундомір, але й зупинити час у потрібний момент, продовжити відлік часу після зупинки секундоміру та обнулити секундомір. Для цього повинні бути передбачені відповідні елементи управління, наприклад, кнопки *Стоп* та *Сброс*:

```
<Grid Grid.Row="1" >
<Button Name="Start" Height="30" Width="Auto" Grid.Row="1" Grid.Column="0" >Старт</Button>
<Button Name="Stop" Height="30" Width="Auto" Grid.Row="1" Grid.Column="1" >Стоп</Button>
<Button Name="Reset" Height="30" Width="Auto" Grid.Row="1" Grid.Column="2">Сброс</Button>
<Grid.ColumnDefinitions>
<ColumnDefinition></ColumnDefinition>
<ColumnDefinition></ColumnDefinition>
<ColumnDefinition></ColumnDefinition>
</Grid.ColumnDefinitions>
</Grid>
```

```
<Window.Triggers>
    <EventTrigger SourceName="Start" RoutedEvent="Button.Click">
        ... // для секундної стрілки
    </EventTrigger>

... // опис подій початку анімації для хвилинної та годинної стрілки

    <EventTrigger SourceName="Pause" RoutedEvent="Button.Click">
        <PauseStoryboard BeginStoryboardName="Sec"/>
    </EventTrigger>

... // опис подій призупинення анімації для хвилинної та годинної стрілки

    <EventTrigger SourceName="Stop" RoutedEvent="Button.Click">
        <StopStoryboard BeginStoryboardName="Sec"/>
    </EventTrigger>

... // опис події закінчення анімації для хвилинної та годинної стрілки

</Window.Triggers>
```

Рисунок 2 – Колекція Triggers

Тоді подія зупинення відліку часу, яка повинна бути прив'язана до кнопки *Стоп*, матиме такий вигляд:

```
<EventTrigger SourceName="Stop" RoutedEvent="Button.Click">
<PauseStoryboard BeginStoryboardName="Sec"/>
</EventTrigger>
```

де RoutedEvent="Button.Click" – подія, яка зупинятиме анімацію (натискання кнопки); SourceName="Stop" – назва елементу управління, до якого прив'язана подія (кнопка *Стоп*). Зупинити можна лише ту анімацію, яка має ім'я, тому

запис PauseStoryboard BeginStoryboardName="Sec" – це команда призупинити анімацію, яка має ім'я *Sec*. Такий самий код слід записати для зупинення анімації для хвилинної та годинної стрілки (але вони будуть мати інше ім'я).

Якщо після того, як була зупинена анімація, знов натиснути кнопку *Старт*, то відлік часу продовжиться з того місця, де була зупинка.

Для обнулення часу на секундомірі слід колекцію Triggers поповнити ще трьома подіями, які матимуть наступний вигляд:

```
        <EventTrigger SourceName="Reset" RoutedEvent="Button.Click">
<StopStoryboard BeginStoryboardName="Sec"/>
</EventTrigger>
```

де StopStoryboard BeginStoryboardName="Sec" скидає значення відповідної стрілки.

Остаточно інтерфейс додатку-секундоміра матиме вигляд, наведений на рис. 3.

Рисунок 3 – Інтерфейс додатку-секундоміра

Для того, щоб додаток мав іконку, слід відкрити властивості проекту та додати картинку з розширенням .ico (рис. 4). Після цього слід зібрати рішення (рис. 5).

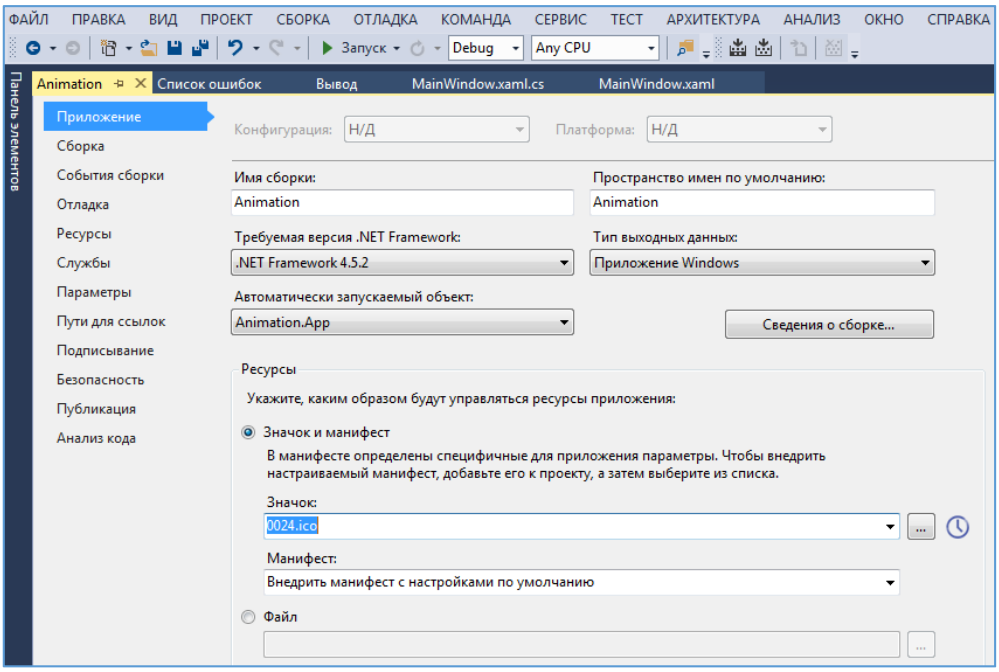

Рисунок 4 – Додавання іконки до готового проекту

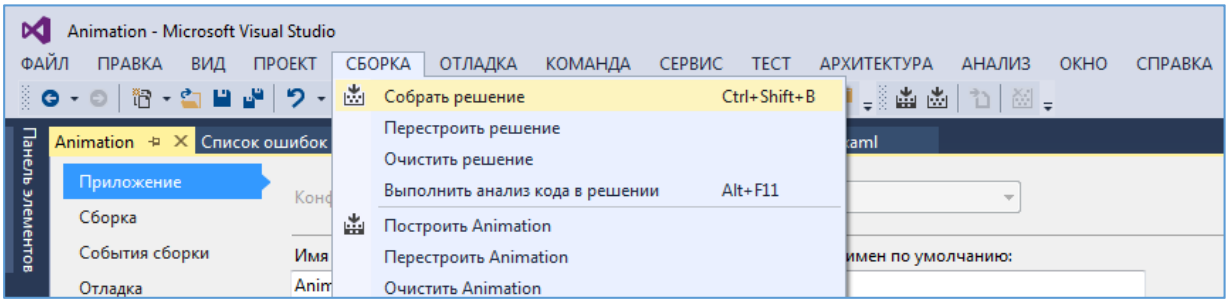

Рисунок 5 – Зборка проекту

У папці проекту в bin\Debug буде знаходитись готовий додаток, який можна використовувати самостійно, наприклад, викинути на робочий стіл (рис. 6) або поділитися з друзями.

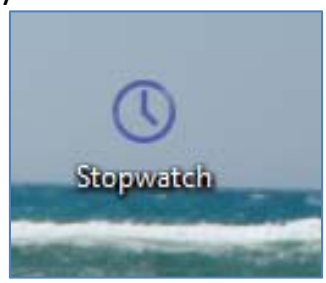

Рисунок 6 – Готовий додаток

Висновки: У статті визначено особливості розробки додатків за технологією Windows Presentation Foundation, а саме: компонування, робота з анімацією та подіями, які її викликають. На прикладі розробки секундоміра детально показано створення додатку лише мовою XAML.

ЛІТЕРАТУРА

1. Есипова Н.Д. Развитие логического мышления младших школьников на внеклассных занятиях по математике с использованием ЭВМ. Научная библиотека диссертаций и авторефератов disserCat http://www.dissercat.com/content/razvitie-logicheskogo-myshleniya-mladshikh-shkolnikov-na-vneklassnykh-zanyatiyakh-po-matemat#ixzz4bgCHqioz

2. Сенкевич Г. Компьютер для людей с ограниченными возможностями. - СПб.: БВХ-Петербург, 2014. – 320 с.: ил

3. Натан А. WPF 4. Подробное руководство. - Пер. с англ. - СПб.: Символ-Плюс, 2011. - 880 с.

4. Чарльз Петцольд. Программирование для Microsoft Windows 8 (6 издание). Разработка приложений для Windows Store на C# и XAML.– СПб.: Питер, 2014. – 1008 с.

5. Управление воспроизведением. [Электронный ресурс]. Режим доступа: https://professorweb.ru/my/WPF/graphics_and_animation/level15/15_9.php — Загл. с экрана.

УДК 004.91

ИССЛЕДОВАНИЕ ОШИБОК ВО ВХОДЯЩИХ ДАННЫХ ДЛЯ ИНФОРМАЦИОННЫХ СИСТЕМ В ОБРАЗОВАНИИ

Д.С. Ковальчук[1], В.В. Герасимов[2]

[1] студент 4 курса, кафедра электронных вычислительных машин, Днепровский национальный университет им. Олеся Гончара, г. Днепр, Украина, e-mail: dimon_koval2612@ukr.net

[2] ассистент кафедры электронных вычислительных машин, Днепровский национальный университет им. Олеся Гончара, г. Днепр, Украина, e-mail: gerasimov@dsu.dp.ua

Аннотация. В работе приведены анализ и решение особенностей и сложностей в организации процесса формирования унифицированной отчетной документации об успеваемости студентов посредством специализированной информационной системы.

Ключевые слова: программа, информационная система, ошибка, данные, отчетный документ, способ исправления ошибок.

RESEARCH OF INCOMING DATA MISTAKES OF INFORMATION SYSTEMS IN EDUCATION

D.S. Kovalchuk[1], V.V. Gerasimov[2]

[1] student, Department of Computer, Oles Honchar Dnepr National University, Dnepr, Ukraine, e-mail: dimon_koval2612@ukr.net

[2] assistant, Department of Computer, Oles Honchar Dnepr National University, Dnepr, Ukraine, e-mail: gerasimov@dsu.dp.ua