WinUI 3를 사용한 녹음기 제작

우제현*, 이수용*, 심재창*
*안동대학교 컴퓨터공학과
e-mail : wjh2335@naver.com

Creating a Recorder Using WinUI 3

Jehyun Woo*, Suyong Lee*, Jaechang Shim*
*Dept. of Computer Engineering, Andong National University

요 약

마이크로소프트사는 자신들의 생태계 강화를 위해 많은 시도를 하였고 실제로 데스크탑 운영체제 부분에서 윈도우를 통해 많은 사용자를 모을 수 있었지만 개발 환경 측면에서는 많은 사용자를 모으지는 못하였다. 본 논문에서는 이전 마이크로소프트사가 왜 개발 환경 통합에 실패했는지, 윈도우의 개발 환경이 어떻게 변해 왔는지 알아볼 것이며, 마이크로소프트사가 개발 환경에 변화를 주기 위해 새로 출시한 WinUI 3가 개발자들에게 편의성을 제공해 줄 수 있을지 직접 녹음기를 구현해 보며 연구해 보고자 한다.

1. 서론

2020 년 한국의 인터넷 이용률은 96.5% 이다.[1] 그중 윈도우(Windows)를 사용하고 있는 인원들만 90%를 넘어가고 있다. 통계를 통해 알 수 있듯이 현재 대부분의 개발자들이 소프트웨어를 개발함에 있어 윈도우를 무시할 수 없다. 윈도우 운영 체제를 개발한 마이크로소프트사 또한 자신들의 압도적인 점유율을 알기 때문에 윈도우에 최적화된 프레임워크를 만들고, API 지원 등 개발자들이 더 편하게 작업할 수 있는 개발 환경을 제공하며 개발자들을 자신의 플랫폼에 종속시키려 하고 있다. 마이크로소프트사는 Windows Desktop(win32) Apps MFC(Microsoft Foundation Class), WinForms, WPF(Windows Presentation Foundation)를 만들었고, Windows UWP(Universal Windows Platform) Apps UWP XAML, WinUI 2 를 만들었다.[2] 하지만 App 종류에 따른 API 변화, 사용 방법이 다른 라이브러리 등 사용의 편의성 저하로 개발자들을 모으는데 실패했다. 이후 마이크로소프트사에서 이 문제를 해결하기 위해 차세대 WinUI 프레임워크 WInUI 3 를 개발하여 다시 개발자들을 모으고 있다. 본 논문에서는 실제로 WinUI 3 를 사용하여 녹음기를 제작하며 WinUI 3 를 체험해 보고자 한다.

2. 관련연구

WinUI 3 는 Windows 앱 SDK 에 포함된 UI 플랫폼 구성 요소로 Windows10(1809+), Windows11 의 Windows 앱 SDK 에서 지원하는 앱과 호환되는 WinUI 프레임워크이다.[3] WinUI 3 가 개발되기까지 마이크로소프트사는 여러 라이브러리 프레임워크, GUI(Graphical User Interfac)를 통해 개발 환경을 변화시켰다.

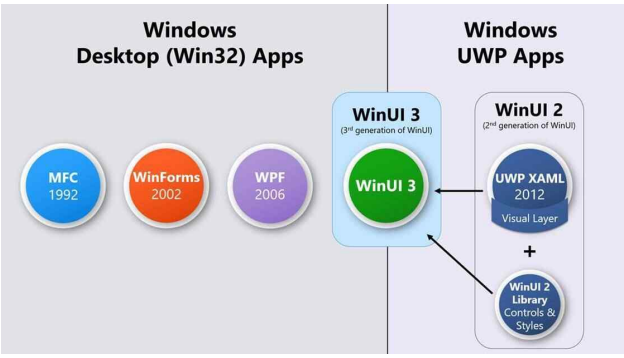

그림 1. 개발 환경 변화

WinUI 3 를 알기 위해서는 이전의 프레임워크들을 알아야 한다. WinUI 3 는 모든 윈도우 개발자에게 UX 프레임워크를 제공하기 위한 목적으로 MFC, WinForms, WPF 와 UWP 앱을 지원하기 위해 첨단 UI 프레임워크로 제작했다. 따라서 WinUI 3 를 알기 위해서는 이전의 개발 환경을 알아야 한다. MFC 는 1992 년 발표되었으며 이 시기부터 소프트웨어 개발에 C 를 대체하여 C++이 도입되기 시작했다. MFC 또한 앱을 만들기 위한 C++ 및 객체 지향 UI 프레임워크로 구성되었다. 그 당시 기준으로 개발자가 손쉬운 GUI 기반의 제작 환경을 제공받을 수 있다는 장점으로 사용자들을 모을 수 있었다. 이후 2002 년 마이크로소프트사의 .NET 프레임워크와 WinForms 이 발표되었다. WinForms 는 .NET 프레임워크를 기반으로 하여 향상된 프로그래밍 환경을 제공하여 개발자의 편의와 생산성을 높일 수 있었지만 .NET 응용 프로그램에서만 사용할 수 있게 되어 기존 MFC 사용자들을 흡수할 수 없었다. 2006 년 발표된 WPF 또한 XAML 을 사용하여 GUI 기반이 더 좋아졌지만 .NET 기반으로 MFC 사용자를 끌어오지 못했고, XAML 의 추가로 기존 WinForms 사용자 또한

끌어들이지 못했다. 이후 2012 년 다시 프레임워크를 통합하기 위해 마이크로소프트사는 GUI 프레임워크 UWP XAML 을 발표했다. 기존 데스크탑 개발을 넘어 Xbox, HoloLens 등 플랫폼에서 자유로운 앱 개발을 통해 프레임워크를 통일하려 했다. .NET 과 C++를 기반으로 그래픽 엔진, 내장 Fluent Design Controls & Styles, 고성능 데이터 바인딩을 제공해 향상된 성능을 가졌지만, UWP 로 빌드 된 앱은 UWP 앱으로 구분되어 기존 MFC, WinForms, WPF 로 빌드 된 데스크탑(Win32)앱과 호환되지 않는다. 기존 앱과는 호환되지 않고 UWP XAML 자체가 기존 데스크탑앱과 API 부터 라이브러리 등 생태계가 달라 데스크탑앱에 익숙해져 있던 일부 개발자들이 UWP XAML 을 기피하게 되며 마이크로소프트사의 개발 환경 통합이 실패하게 된다. 이후 2018 년 WinUI 2 가 출시됐지만 WinUI 2 또한 UWP 앱과 같은 이유로 호평을 받지 못했다. 그러고 최근 다시 개발 환경의 변화를 위해 WinUI 3 가 출시되었다. 코드 베이스는 UWP XAML 이며 WinUI 2 의 controls & styles 을 계승하여 UWP 앱을 빌드함으로 이전 UWP XAML 보다 향상된 성능을 가졌고 UWP XAML 이 윈도우와 함께 배포되어 일부 윈도우 업데이트가 불가능한 사용자는 UWP XAML 의 모든 기능을 사용할 수 없었는데, WinUI 3 는 윈도우와 분리되어 항상 윈도우를 최신 버전으로 업데이트하지 않아도 앱을 최신 상태로 유지할 수 있고, 병목 상태가 발생하지 않는 장점으로 UWP XAML 의 사용자를 끌어들였다. 추가로 UWP 앱이 기본이지만 데스크탑앱의 UI layer 를 직접 사용할 수 있어 기존의 UI 프레임워크도 대체할 수 있어 기존 MFC, WinForms, WPF 사용자들 또한 끌어들일 수 있다. WinUI 3 는 이전까지 없던 모든 플랫폼을 통합하는 진보된 프레임워크로 다양한 API, 호환성, 플랫폼에 종속되지 않는 장점과 이전에 없던 기능들을 제공하는 최신 프레임워크로 기존 UWP 앱 사용자와 데스크탑앱 사용자들을 끌어들이고 하나로 통합시킬 수 있을 것으로 보인다.[4][5]

3. WinUI 3 를 사용한 녹음기 구현

본 논문의 중심 내용은 XAML 과 C++을 사용한 UI 구현의 초점을 두고 설명할 것이다. 가장 먼저 그림 2 와 같이 Visual Studio 2022 를 통해 WinUI 3 프로젝트를 시작해 준다.

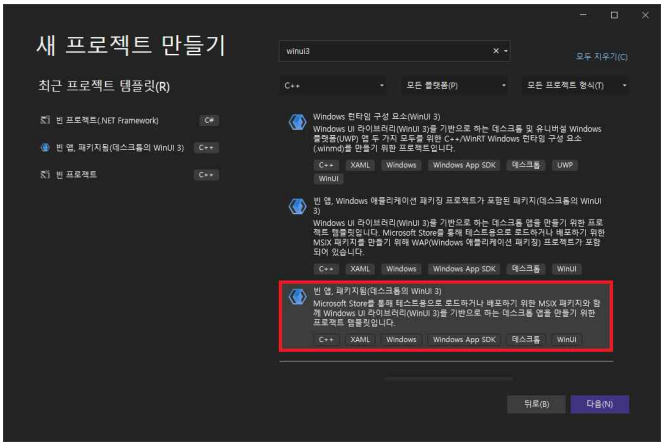

그림 2. WinUI 3 프로젝트 시작

처음 실행하면 빈 창이 뜨게 되는데 cpp 를 통해 창의 크기를 조정해 준다. 그림 3 과 같이 windowSize 의 Height 와 Width 를 230, 270 으로 설정하고 windowPoint 를 통해 화면 중앙에 창이 시작될 수 있도록 X, Y 값을 정해주고 m_mainAppWindow 클래스의 Resize(), Move()를 사용해 창이 나타날 위치와 창의 너비와 높이를 정해준다. 이후 Title 을 통해 제목을 정해준다.

```
MainWindow::MainWindow()
{
    InitializeComponent();

    m_mainAppWindow = GetAppWindowForCurrentWindow();

    winrt::Windows::Graphics::SizeInt32 windowSize;
    windowSize.Height = 230;
    windowSize.Width = 270;
    m_mainAppWindow.Resize(windowSize);

    winrt::Windows::Graphics::PointInt32 windowPoint;
    windowPoint.X = (1920 - windowSize.Width) / 2;
    windowPoint.Y = (1080 - windowSize.Height) / 2;
    m_mainAppWindow.Move(windowPoint);

    this->Title(L"WooSu 녹음기");
}
```

그림 3. 녹음기 창 크기 및 위치

그림 4 와 같이 MainWindow.Xaml 에서 Grid 와 RowDefinition 을 통해 창의 영역과 행을 구분한 뒤 각 영역에 Slider 를 통해 재생바, TextBlock 을 통해 재생시간, StackPanel 을 통해 재생, 정지, 녹음, 일시정지 버튼, StackPanel 을 통해 설정, 폴더 열기, 삭제 버튼을 구현해 준다.

```
<Grid>
    <Grid x:Name="recordPanel" Padding="10">
        <Grid.RowDefinitions>
            <RowDefinition Height="30"/>
            <RowDefinition Height="30"/>
            <RowDefinition Height="50"/>
            <RowDefinition Height="50"/>
        </Grid.RowDefinitions>

        <Slider Grid.Row="0"
                x:Name="_timeSlider" PointerCaptureLost="slider_Click" PointerMoved="slider_Click"
                Orientation="Horizontal" HorizontalAlignment="Center"
                Width="150" Value="0" Minimum="0" Maximum="100"/>

        <TextBlock Grid.Row="1"
                x:Name="_time" HorizontalAlignment="Right">0</TextBlock>

        <StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center">
            <Button Click="playButton_Click">▶</Button>
            <Button Click="stopButton_Click">■</Button>
            <Button x:Name="StartBtn" Width="75" Click="startButton_Click">●</Button>
            <Button x:Name="PauseBtn" Click="pauseButton_Click">⏸</Button>
        </StackPanel>

        <StackPanel Grid.Row="3" Orientation="Horizontal" HorizontalAlignment="Right" VerticalAlignment="Bottom">
            <Button Click="settingButton_Click">설정</Button>
            <Button Click="openfolderButton_Click">폴더열기</Button>
            <Button>삭제</Button>
        </StackPanel>
    </Grid>
```

그림 4. 창의 구분과 기능

그림 5 의 코드와 같이 설정창 구현을 해준다. 이전과 같이 MainWindow.Xaml 에서 Grid 와 RowDefinition 을 통해 창의 영역과 행을 구분한 뒤 Button 을 통해 X 종료 버튼, StackPanel 에 TextBlock 와 Slider 를 통해 볼륨을 조절해 줄 슬라이더 바와 범위 텍스트를 구현해 준다. 이때 Orientation 을 Vertical 로 하여 이전과 다르게 슬라이더 바가 수직으로 생성되게 한다. 이후 StackPanel 에 Button 과 TextBlock 를 통해 GitHub 연결 버튼과 제작자 이름 텍스트를 구현해 준다.

```
41          <Border Margin="4" Padding="4">
42              <Grid RowSpacing="3">
43                  <Grid.RowDefinitions>
44                      <RowDefinition Height="20"/>
45                      <RowDefinition Height="110"/>
46                      <RowDefinition Height="35"/>
47                  </Grid.RowDefinitions>
48                  <Button Grid.Row="0"
49                          Width="16" Padding="0" HorizontalAlignment="Right" VerticalAlignment="Top">X</Button>
50
51                  <StackPanel Grid.Row="1"
52                              Orientation="Horizontal" HorizontalAlignment="Center">
53                      <Grid RowSpacing="3">
54                          <Grid.RowDefinitions>
55                              <RowDefinition Height="20"/>
56                              <RowDefinition Height="60"/>
57                              <RowDefinition Height="20"/>
58                          </Grid.RowDefinitions>
59                          <TextBlock Grid.Row="0" VerticalAlignment="Top" FontSize="10">10db</TextBlock>
60                          <TextBlock Grid.Row="1" VerticalAlignment="Center" FontSize="10">0db</TextBlock>
61                          <TextBlock Grid.Row="2" VerticalAlignment="Top" FontSize="10">-10db</TextBlock>
62                      </Grid>
63                      <Slider Orientation="Vertical" VerticalAlignment="Top" Height="100" Value="0" Minimum="-10" Maximum="10"/>
64                      <Slider Orientation="Vertical" VerticalAlignment="Top" Height="100" Value="0" Minimum="-10" Maximum="10"/>
65                      <Slider Orientation="Vertical" VerticalAlignment="Top" Height="100" Value="0" Minimum="-10" Maximum="10"/>
66                      <Slider Orientation="Vertical" VerticalAlignment="Top" Height="100" Value="0" Minimum="-10" Maximum="10"/>
67                      <Slider Orientation="Vertical" VerticalAlignment="Top" Height="100" Value="0" Minimum="-10" Maximum="10"/>
68                      <Slider Orientation="Vertical" VerticalAlignment="Top" Height="100" Value="0" Minimum="-10" Maximum="10"/>
69                  </StackPanel>
70                  <StackPanel Grid.Row="2" Orientation="Horizontal">
71                      <Button HorizontalAlignment="Center" VerticalAlignment="Center">GitHub</Button>
72                      <TextBlock Foreground="White">ㅇㅇㅇ</TextBlock>
73                      <TextBlock HorizontalAlignment="Right" VerticalAlignment="Center">우재현, 이수홍 제작</TextBlock>
74                  </StackPanel>
75              </Grid>
76          </Border>
```

그림 5. 녹음기 설정창

각 버튼의 구현 방법은 다음과 같다. 설정창 버튼과 X 버튼의 경우 그림 6 과 같이 Visibility 를 사용하여 이전 화면을 접고 새 화면을 보이게 함으로 설정창이 열리고 닫히는 효과를 구현하였다.

```
void MainWindow::settingButton_Click(winrt::Windows::Foundat
{
    settingPanel().Visibility(Visibility::Visible);
    recordPanel().Visibility(Visibility::Collapsed);
}
void MainWindow::settingCloseButton_Click(winrt::Windows::Fo
{
    settingPanel().Visibility(Visibility::Collapsed);
    recordPanel().Visibility(Visibility::Visible);
}
```

그림 6. 설정창 X 종료 버튼

GitHub 버튼과 폴더 열기 버튼은 그림 7 과 같이 ShellExecute() 함수를 사용하여 인자로 원하는 경로를 주어 버튼을 누르면 GitHub URL 이 연결되고, 컴퓨터의 C 드라이브가 열리게 구현하였다.

```
void MainWindow::openfolderButton_Click(winrt::Windows::Foundati
{
    CString STR = _T("C:\");
    ShellExecute(NULL, L"open", STR, NULL, NULL, SW_SHOW);
}
void MainWindow::githubButton_Click(winrt::Windows::Foundation::
{
    CString STR = _T("https://github.com/wjh2335/WooSuRecord");
    ShellExecute(NULL, L"open", STR, NULL, NULL, SW_SHOW);
}
```

그림 7. 폴더열기 버튼과 GitHub 버튼

재생, 정지, 일시정지 버튼은 그림 8 과 같이 구현하였다. 재생 버튼을 누르면 cnt 값이 1 증가하고 값이 슬라이더와, 재생 시간에 반영된다. 정지 버튼을 누르면 cnt 값이 0 으로 초기화되고 값이 슬라이더와, 재생 시간에 반영된다. 녹음 기능을 구현 못 했기에 일시정지 버튼을 누르면 GetLocalTime 를 통해 현재 시간 값을 불러오도록 구현하였다.

```
void MainWindow::playButton_Click(winrt::Windows::Foundation::IInspectable const& sender, winrt::Micr
{
    std::string text_str;
    hstring text_hstr = _time().Text();
    text_str.assign(_First: text_hstr.begin(), _Last: text_hstr.end());
    int cnt = atoi(_String: text_str.c_str());
    ++cnt;
    if (cnt > _timeSlider().Maximum()) cnt = 0;
    _time().Text(value: to_hstring(value: cnt));
    _timeSlider().Value(value: cnt);
}
void MainWindow::stopButton_Click(winrt::Windows::Foundation::IInspectable const& sender, winrt::Micr
{
    std::string text_str;
    hstring text_hstr = _time().Text();
    text_str.assign(_First: text_hstr.begin(), _Last: text_hstr.end());
    int cnt = atoi(_String: text_str.c_str());
    cnt = 0;
    _time().Text(value: to_hstring(value: cnt));
    _timeSlider().Value(value: cnt);
}
void MainWindow::pauseButton_Click(winrt::Windows::Foundation::IInspectable const& sender, winrt::Mic
{
    SYSTEMTIME cur_time;
    GetLocalTime(lpSystemTime: &cur_time);
    wsprintf(g_time_str, L"%02d시%02d분%02d초", cur_time.wHour, cur_time.wMinute, cur_time.wSecond);
    _time().Text(value: g_time_str);
}
```

그림 8. 재생, 정지, 일시정지 버튼

녹음 버튼은 그림 9 와 같다. 처음 stat 값이 0 으로 버튼이 눌릴 때마다 state 값이 1,0 로 바꾸며 녹음 중을 1, 녹음이 아닌 경우를 0 으로 구분한다. 녹음을 시작하면 시작 시간 time(NULL)을 start 에 넣어준다. 그 후 버튼을 다시 눌러 녹음이 종료되면 종료 시간 time(NUll)을 end 에 넣어준다. 그 후 end-star 을 해줘 녹음된 시간을 구한 뒤 슬라이더와 재생 시간에 반영한다.

```
void MainWindow::startButton_Click(winrt::Windows::Foundation::IInspe
{
    if (state == 0)
    {
        state = 1;
        StartBtn().Content(box_value(value:L"녹음중..."));
        start = time(_Time:NULL);
        _time().Text(value:to_hstring(value:time(_Time:NULL) - start));
    }
    else
    {
        state = 0;
        StartBtn().Content(box_value(value:L"●"));
        end = time(_Time:NULL);
        _time().Text(value:to_hstring(value:end - start));
        _timeSlider().Maximum(value:end - start);
        _timeSlider().Value(value:end - start);
    }
}
```

그림 9. 녹음 버튼

마지막 삭제 버튼은 그림 10 과 같이 구현했다. cnt 에 0 을 넣어 재생 시간 텍스트와 슬라이더에 반영하고 길이가 변한 슬라이더 값을 100 으로 되돌려준다.

```
void MainWindow::deleteButton_Click(winrt::Windows::Foundation::IIns
{
    std::string text_str;
    hstring text_hstr = _time().Text();
    text_str.assign(_First:text_hstr.begin(), _Last:text_hstr.end());
    int cnt = atoi(_String:text_str.c_str());
    cnt = 0;
    _time().Text(value:to_hstring(value:cnt));
    _timeSlider().Value(value:cnt);
    _timeSlider().Maximum(value:100);
}
```

그림 10. 삭제 버튼

이 외에도 그림 11 과 같이 슬라이더가 이동값을 재생 시간으로 변환해주는 등 많은 코드가 사용되었다.

```
void MainWindow::slider_Click(winrt::Windows::Foundat
{
    _time().Text(to_hstring(_timeSlider().Value()));
}
```

그림 11. 슬라이더 재생 시간 연동

구현물로 그림 12 와 같은 소프트웨어를 완성하였다.[6][7]

그림 12. WinUI 3 로 구현한 녹음기

4. 결론

본 논문에서는 WinUI 의 차세대 프레임워크인 WinUI 3 를 직접 체험해 보며 개발 환경을 통합시킬 수 있는지 확인하기 위해 녹음기를 제작하였다. 본 논문에서는 실제로 녹음 기능의 구현은 실패했지만 UI 를 구현하며 차세대 개발 환경의 가능성이 충분하다고 기대할 수 있었다. XAML 과 C++를 사용하여 UI 를 쉽게 구현할 수 있었으며 오류 없이 작동하는 걸 확인할 수 있었다. 코드 베이스가 UWP XAML 이므로 기존 UWP XAML 사용자들은 이전 방식과 같은 형식으로 WinUI 3 를 사용할 수 있으며, WinUI 3 가 데스크탑앱과 호환되므로 MFC, WinForms, WPF 사용자들에게도 좋은 호응을 얻을 것을 예상할 수 있다. 기능적 측면 이외도 WinUI 3 는 현재 마이크로소프트사가 관리하는 프레임워크인 만큼 윈도우를 사용하고, 윈도우를 대상으로 하는 모든 사용자들에게 편의를 제공해 줄 수 있는 도구가 될 것이다. WinUI 3 이후 개발 환경의 변화가 오기 전까지는 마이크로소프트사의 최첨단 프레임워크로 지속적인 지원과 호환성을 바탕으로 강력한 생태계를 구성해 나갈 것이다. 남들보다 앞서 WinUI 3 를 체험해보고 연구할 가치가 있다고 기대한다.

참고문헌

[1]Internet usage rate(2020). https://kosis.kr/statHtml/statHtml.do?orgId=101&tblId=DT_2KAAA13_OECD(accessed June 7, 2022)

[2]windows programming, MFC, WINUI(2022). https://cafe.naver.com/simplecpp(accessed June 7, 2022)

[3]About WinUI. https://microsoft.github.io/microsoft-ui-xaml/about.html(accessed June 7, 2022)

[4].NET Conf Mini 21.08 Introducing WIN UI 3(2021). https://www.youtube.com/watch?v=GgOWfSonXgs(accessed June 5, 2022)

[5]Everything you need to know about WinUI(2020). https://www.youtube.com/watch?v=1ZBMHFoMVAs(accessed June 7, 2022)

[6]DMRecorder(2022). https://github.com/dimohy/DMRecorder(accessed May 23, 2022)

[7]Microsoft WindowsAppSDK Samples(2022). https://github.com/microsoft/WindowsAppSDK-Samples/tree/main/Samples/Windowing/cpp-winui(accessed May 30, 2022)