tutorial

Windows App SDK

Slide Game

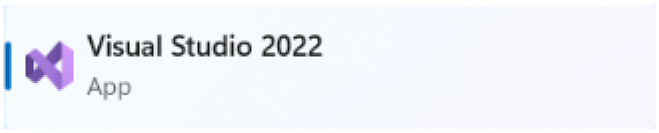
Slide Game

Slide Game shows how to create a game based on those sliding puzzle games where the objective is to place all the numbers in descending numerical order by **Clicking** on a **Square** to move it into the empty position using a toolkit from **NuGet** using the **Windows App SDK**.

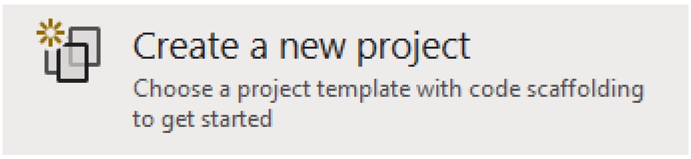
Step 1

Follow **Setup and Start** on how to get **Setup** and **Install** what you need for **Visual Studio 2022** and **Windows App SDK**.

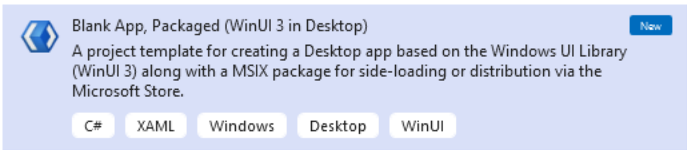
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



Once **Visual Studio 2022** has started select **Create a new project**.

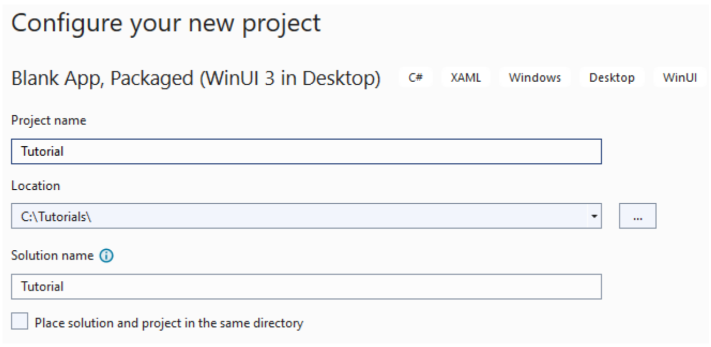

Create a new project
Choose a project template with code scaffolding to get started

Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**.


Blank App, Packaged (WinUI 3 in Desktop)

A project template for creating a Desktop app based on the Windows UI Library (WinUI 3) along with a MSIX package for side-loading or distribution via the Microsoft Store.

After that in **Configure your new project** type in the **Project name** as *SlideGame*, then select a Location and then select **Create** to start a new **Solution**.


Configure your new project

Blank App, Packaged (WinUI 3 in Desktop)

Project name: Tutorial

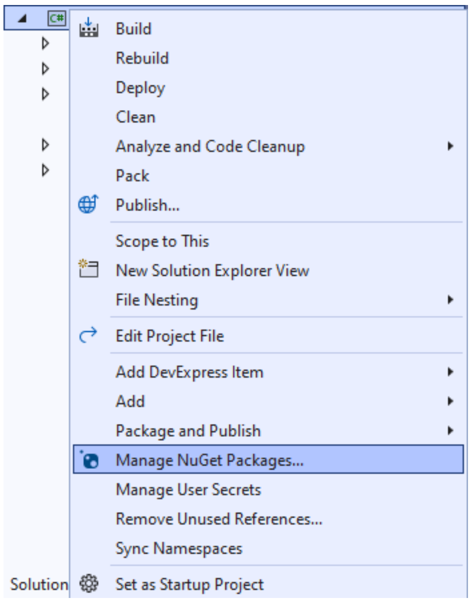
Location: C:\Tutorials\

Solution name: Tutorial

☐ Place solution and project in the same directory

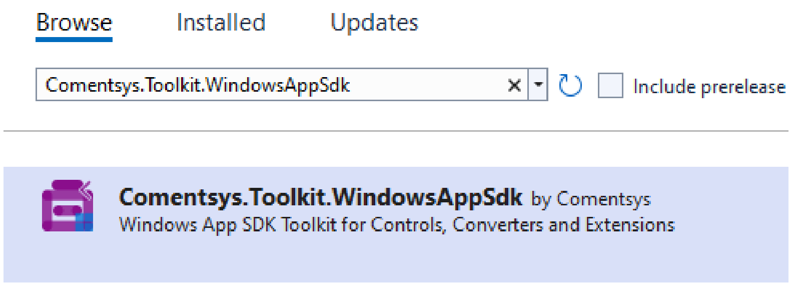
Step 2

Then in **Visual Studio** within **Solution Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Manage NuGet Packages...**



Step 3

Then in the **NuGet Package Manager** from the **Browse** tab search for **Comentsys.Toolkit.WindowsAppSdk** and then select **Comentsys.Toolkit.WindowsAppSdk by Comentsys** as indicated and select **Install**



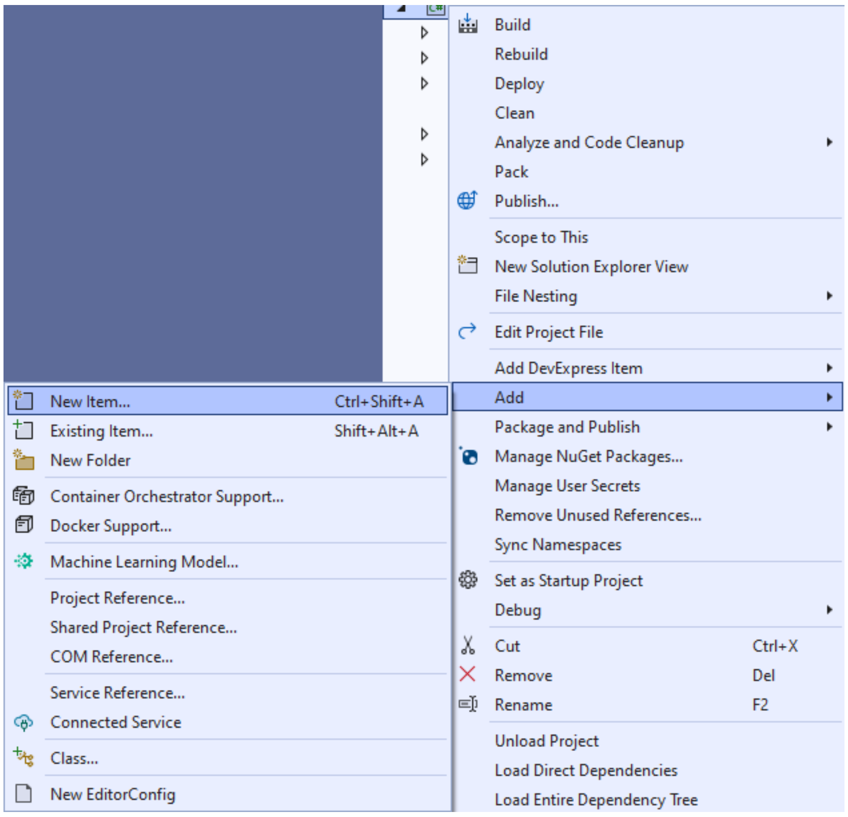
Comentsys.Toolkit.WindowsAppSdk

Comentsys.Toolkit.WindowsAppSdk by Comentsys
Windows App SDK Toolkit for Controls, Converters and Extensions

This will add the package for **Comentsys.Toolkit.WindowsAppSdk** to your **Project**. If you get the **Preview Changes** screen saying **Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.** You can read the message and then select **OK** to **Install** the package, then you can close the **tab** for **Nuget: SlideGame** by selecting the **x** next to it.

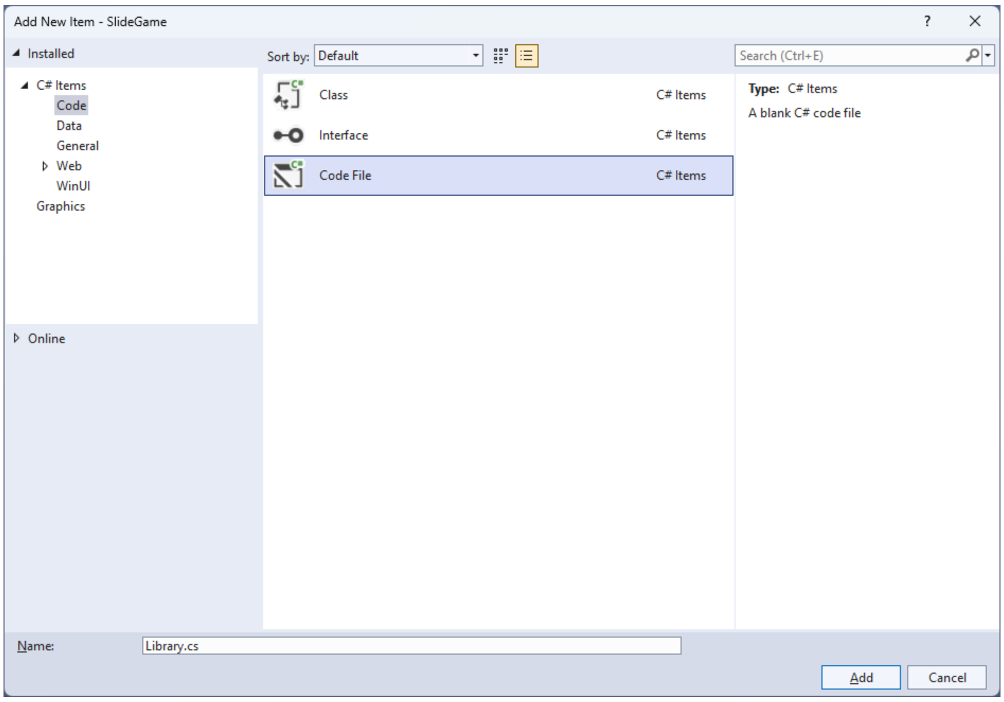
Step 4

Then in **Visual Studio** within **Solution Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item...**



Step 5

Then in **Add New Item** from the **C# Items** list, select **Code** and then select **Code File** from the list next to this, then type in the name of *Library.cs* and then **Click** on **Add**.



Type: C# Items
A blank C# code file

Name: Library.cs

Step 6

You will now be in the **View** for the **Code** of *Library.cs*, within this first type the following **Code**:

```
using Comentsys.Toolkit.WindowsAppSdk;
using Microsoft.UI;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Input;
using Microsoft.UI.Xaml.Media;
using System;
using System.Collections.Generic;
using System.Linq;

namespace SlideGame;

public class Item : Piece
{
    public Item(int row, int column, int index) =>
        (Row, Column, Value) = (row, column, $"{index}");

    public int Row { get; set; }
    public int Column { get; set; }
}

public class Library
{
    private const string title = "Slide Game";
    private const int canvas_size = 400;
    private const int size = 4;

    private readonly Random _random = new((int)DateTime.UtcNow.Ticks);
    private readonly int[,] _board = new int[size, size];

    private Dialog _dialog;
    private Canvas _canvas;

    private int _moves;
    private List<int> _values;

    // Choose, IsValid & IsComplete

    // Update & Move

    // Play & Setup

    // Layout & New

}
```

`Class` defined so far *Library.cs* has **`using`** for package of **`Comentsys.Toolkit.WindowsAppSdk`** and others. It also has a **`namespace`** as it will contain multiple **Classes** although usually those are defined in separate files. *Library.cs* contains a **`Class`** for **`Item`** and for **`Library`** which has **Constants** to represent things needed in the game and there are **Variables** to keep track of values used in the game.

Step 7

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Choose, IsValid & IsComplete`** type the following **Methods**:

```
private List<int> Choose(int minimum, int maximum, int total) =>
    Enumerable.Range(minimum, maximum)
        .OrderBy(r => _random.Next(minimum, maximum))
            .Take(total).ToList();

private bool IsValid(int row, int column) =>
    row >= 0 && column >= 0 && row <= 3 &&
    column <= 3 && _board[row, column] == 0;

private bool IsComplete()
{
    int previous = _board[0, 0];
    for (int row = 0; row < size; row++)
    {
        for (int column = 0; column < size; column++)
        {
            if (_board[row, column] < previous)
                return false;
            previous = _board[row, column];
        }
    }
    return true;
}
```

`Choose` is used to produce a list of unique randomised numbers, **`IsValid`** will check if a position is acceptable and **`IsComplete`** will be used to check if the game has finished correctly.

Step 8

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Update & Move`** type the following **Methods**:

```
private void Update()
{
    _canvas.Children.Clear();
    for (int row = 0; row < size; row++)
    {
        for (int column = 0; column < size; column++)
        {
            if (_board[row, column] > 0)
            {
                var index = _board[row, column];
                var piece = new Item(row, column, index)
                {
                    Foreground = new SolidColorBrush(Colors.White),
                    Fill = new SolidColorBrush(Colors.Black),
                    Height = _canvas.Height / size,
                    Width = _canvas.Width / size,
                    IsSquare = true
                };
                piece.PointerReleased += (object sender,
                    PointerRoutedEventArgs e) =>
                    Play(sender as Item);
                Canvas.SetTop(piece, row * (_canvas.Width / size));
                Canvas.SetLeft(piece, column * (_canvas.Width / size));
                _canvas.Children.Add(piece);
            }
        }
    }
}

private void Move(Item item, int row, int column)
{
    _moves++;
    _board[row, column] = _board[item.Row, item.Column];
    _board[item.Row, item.Column] = 0;
    item.Row = row;
    item.Column = column;
    Update();
    if (IsComplete())
        _dialog.Show($"Correct in {_moves} Moves");
}
```

`Update` is used to update the **`Canvas`** with the current position of the items and will use a **Method** of **`Play`** that will be defined in the next **Step** and **`Move`** is used to set the location of an **`Item`** and will use **`Update`** and will check if the game is finished using **`IsComplete`**.

Step 9

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Play & Setup`** type the following **Methods**:

```
private void Play(Item item)
{
    if (IsValid(item.Row - 1, item.Column))
        Move(item, item.Row - 1, item.Column);
    else if (IsValid(item.Row, item.Column + 1))
        Move(item, item.Row, item.Column + 1);
    else if (IsValid(item.Row + 1, item.Column))
        Move(item, item.Row + 1, item.Column);
    else if (IsValid(item.Row, item.Column - 1))
        Move(item, item.Row, item.Column - 1);
}


public void Setup()
{
    int index = 1;
    _values = Choose(1, _board.Length - 1, _board.Length - 1);
    _values.Insert(0, 0);
    for (int row = 0; row < size; row++)
    {
        for (int column = 0; column < size; column++)
        {
            _board[row, column] = _values[index++];
            if (index == size * size)
                index = 0;
        }
    }
}
```

`Play` is used to position an **`Item`** and will use **`IsValid`** to check if the position is valid then use **`Move`** to place the **`Item`** in the position and **`Setup`** is used to initialise the game using **`Choose`**.

Step 10

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Layout & New`** type the following **Methods**:

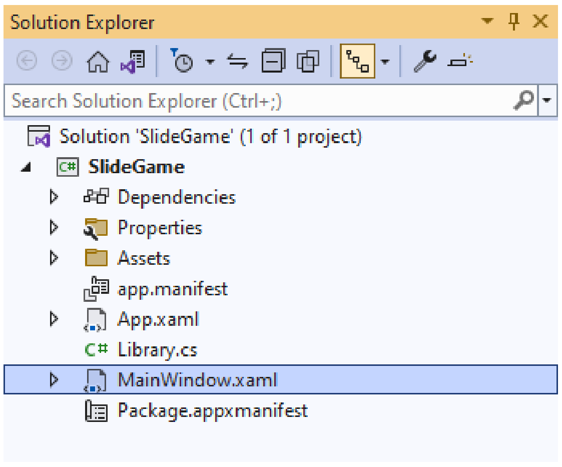
```
public void Layout(Grid grid)
{
    grid.Children.Clear();
    _canvas = new Canvas()
    {
        Height = canvas_size,
        Width = canvas_size
    };
    grid.Children.Add(_canvas);
}

public void New(Grid grid)
{
    _dialog = new Dialog(grid.XamlRoot, title);
    Layout(grid);
    Setup();
    Update();
}
```

`Layout` will create the layout for the game with a **`Canvas`** and **`New`** will setup and start a new game.

Step 11

Then from **Solution Explorer** for the **Solution** double-click on **MainWindow.xaml** to see the **XAML** for the **Main Window**.



Solution 'SlideGame' (1 of 1 project)
SlideGame
Dependencies
Properties
Assets
app.manifest
App.xaml
Library.cs
MainWindow.xaml
Package.appxmanifest

Step 12

In the **XAML** for **MainWindow.xaml** there be some **XAML** for a **`StackPanel`**, this should be **Removed** by removing the following:

```
<StackPanel Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Center">
    <Button x:Name="myButton" Click="myButton_Click">Click Me</Button>
</StackPanel>
```

Step 13

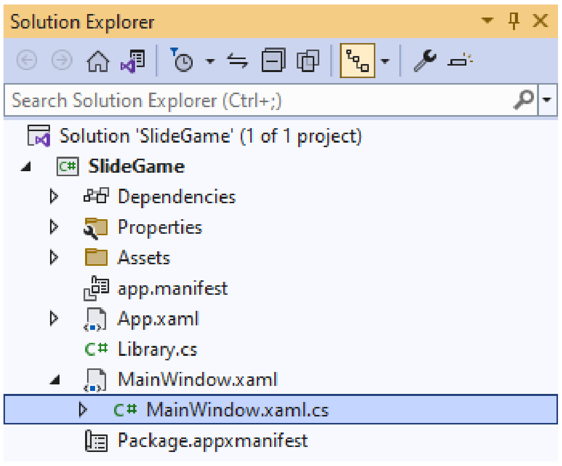
While still in the **XAML** for **MainWindow.xaml** above **`</Window>`**, type in the following **XAML**:

```
<Grid>
    <Viewbox>
        <Grid Margin="50" Name="Display"
        HorizontalAlignment="Center"
        VerticalAlignment="Center" Loaded="New"/>
    </Viewbox>
    <CommandBar VerticalAlignment="Bottom">
        <AppBarButton Icon="Page2" Label="New" Click="New"/>
    </CommandBar>
</Grid>
```

This **XAML** contains a **`Grid`** with a **`Viewbox`** which will scale a **`Grid`**. It has a **`Loaded`** event handler for **`New`** which is also shared by the **`AppBarButton`**.

Step 14

Then, within **Solution Explorer** for the **Solution** select the arrow next to **MainWindow.xaml** then double-click on **MainWindow.xaml.cs** to see the **Code** for the **Main Window**.



Step 15

In the **Code** for **MainWindow.xaml.cs** there be a **Method** of **`myButton_Click(...)`** this should be **Removed** by removing the following:

```
private void myButton_Click(object sender, RoutedEventArgs e)
{
    myButton.Content = "Clicked";
}
```

Step 16

Once **`myButton_Click(...)`** has been removed, type in the following **Code** below the end of the **Constructor** of **`public MainWindow() { ... }`**:

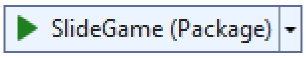
```
private readonly Library _library = new();

private void New(object sender, RoutedEventArgs e) =>
    _library.New(Display);
```

Here an **Instance** of the **Class** of **`Library`** is created then below this is the **Method** of **`New`** that will be used with **Event Handler** from the **XAML**, this **Method** uses Arrow Syntax with the **`=>`** for an Expression Body which is useful when a **Method** only has one line.

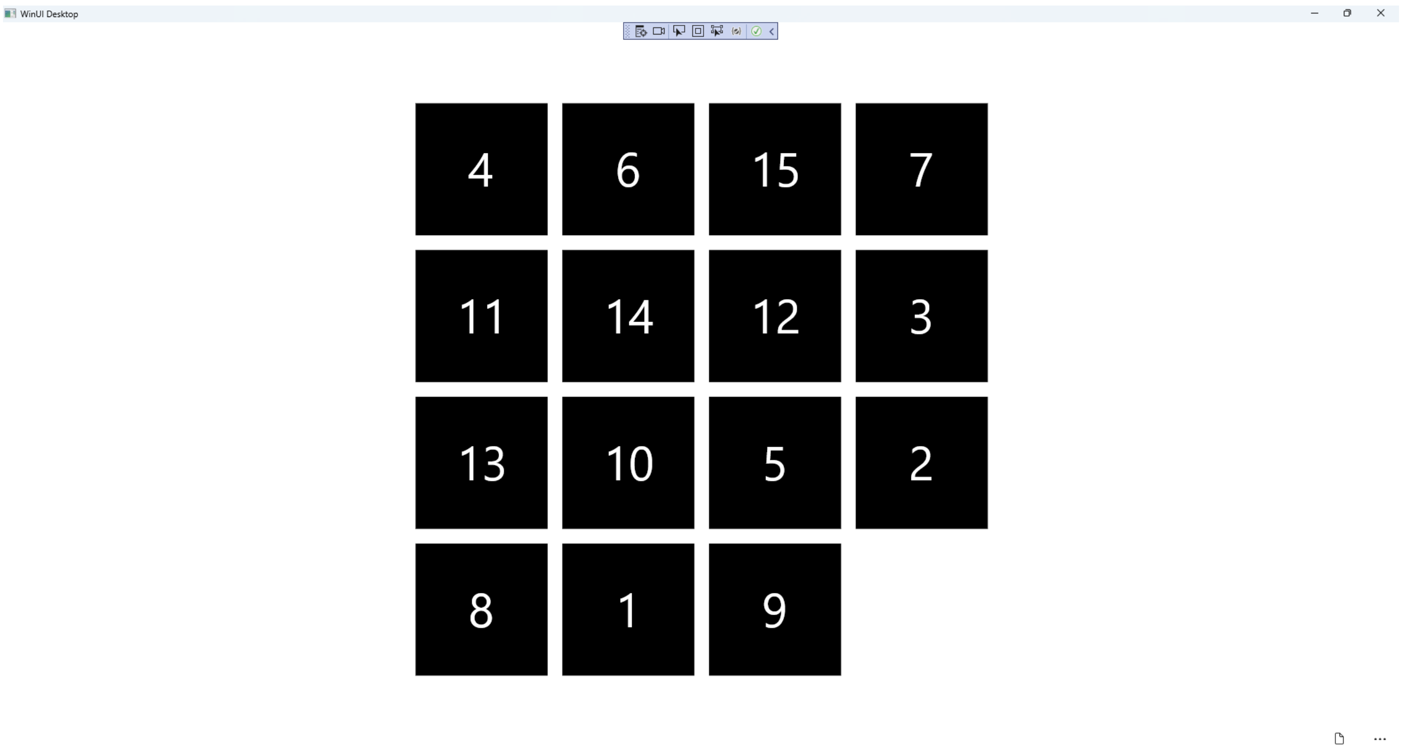
Step 17

That completes the **Windows App SDK** application. In **Visual Studio 2022** from the **Toolbar** select **SlideGame (Package)** to **Start** the application.



Step 18

Once running you win by putting all the numbers in **Order** from *1* to *15* from left to right by selecting the **Square** next to the the empty slot to move it into that slot and then continue until all the **Squares** are in the correct **Order** to complete the game, or you can select *New* to start a new game.



Step 19

To **Exit** the **Windows App SDK** application, select the **Close** button from the top right of the application as that concludes this **Tutorial** for **Windows App SDK** from tutorialr.com!