1 Windows 操作系统概述

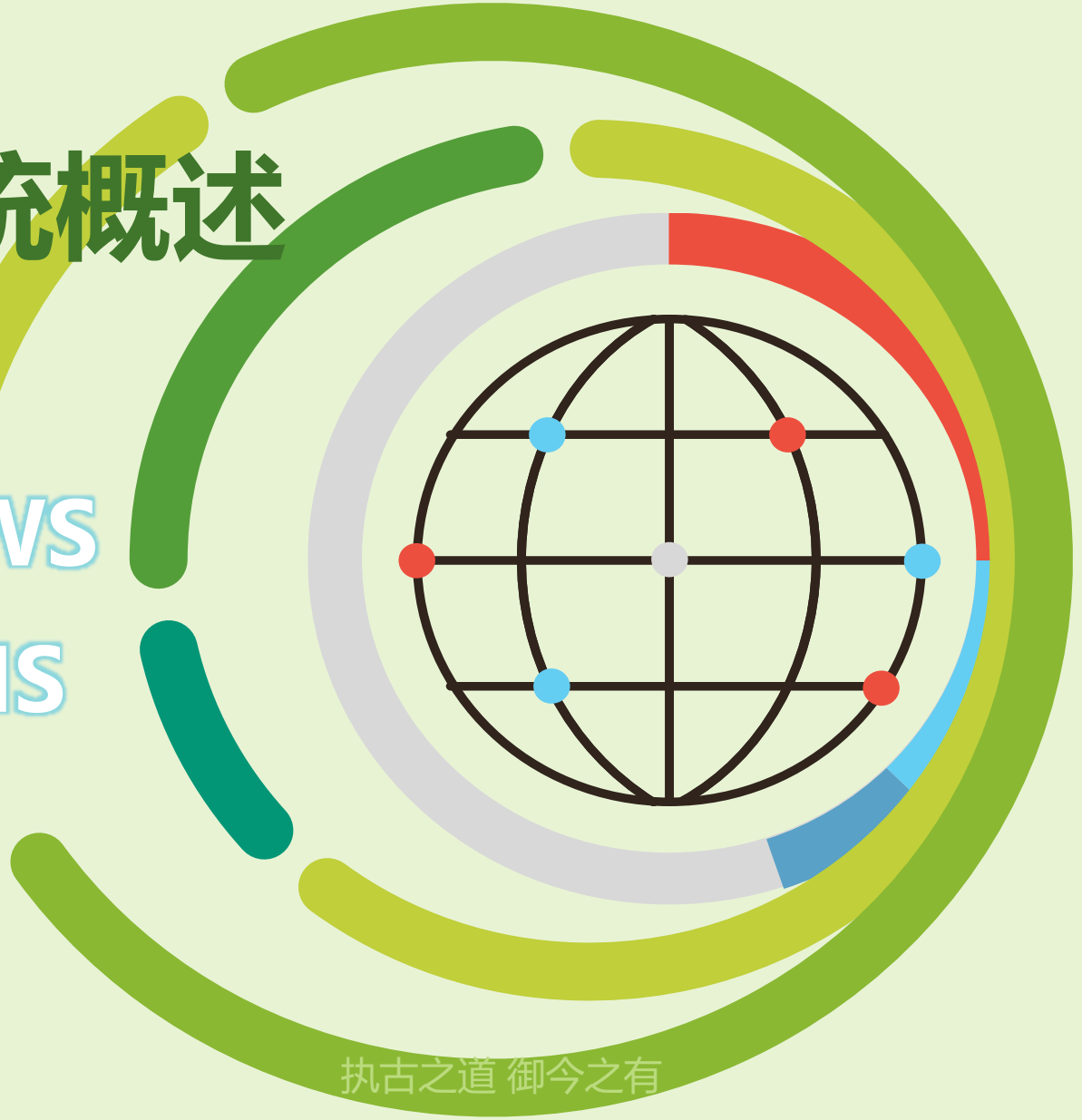
PRINCIPLE OF WINDOWS AND ITS APPLICATIONS

School of CS

Jicheng Hu

jicheng @ yahoo . com

https://gitee.com/wuhanuniversity/



outlines

- 1.1 Introduction
- 1.2 Windows Programming
- 1.3 Windows Form and WPF
- 1.4 UWP, XAML and FLUENT
- 1.5 winRT and WinUI, WebView2

Python ?

PTVS

本次课要求

- **理解** WINDOWS编程模型和框架
- **掌握** Visual Studio Community 2019
- **熟悉** winUI、XAML、C++/winRT
- **了解** WPF、C#/winRT、FLUENT、MFC……

1.1 Windows简介

- Windows 在PC上被广泛使用和普及
- 大多数桌面应用程序基于Windows
- 在智能制造的时代风口，Windows程序设计大有用武之地

Windows程序设计是编程技术人员应该掌握的一项基本技能

涵盖社区、云、IoT、AI、VR...

Windows发展趋势：提供易用的开发环境

- 融合LINUX，内建对LINUX的支持
- 拥抱开源，微软成为最大的开源社区贡献者，并收购了github
  - 开源 VisualStudio Code
  - 开源 WPF, Windows Forms, and WinUI

开发社区

R tools？ RTVS

https://blogs.windows.com/windowsdeveloper/2018/12/04/announcing-open-source-of-wpf-windows-forms-and-winui-at-microsoft-connect-2018/

- 通过Azure形成应用分发的云端战略
- 通过Microsoft 365、Edge卡点边缘计算

智能云和智能边缘

- 通过Visual Studio Code Tools for AI来促使开发者将训练任务提交到Azure Machine Learning，Azure Batch AI，Open Platform for AI或者Linux GPU工作站（例如Azure GPU虚拟机）上运行，开发者可以使用统一的图形用户界面管理云端训练任务和文件
- ONNX项目及ML.NET打造开源跨平台人工智能开发框架 PTVS?
- 开源深度学习框架CNTK(Computational Network Toolkit)
- 通过Fluent布局VR、AR、MR交互，融合到WINDOWS程序开发
- Project Rome —— consistent cross-device and cross-platform app experiences that seamlessly ...

中台?

AI前端
AI后端
AI应用

Windows程序设计编程技术如此众多！

上手快……精通难

好用 vs 领先？

该从哪里开始?

追本溯源……与时俱进

Windows 的发展及技术演进

- DOS => GUI => GDI+ => WPF -> UWP -> FLUENT
- 16位 => 32位 => 64位

https://developer.microsoft.com/en-us/windows/windows-10-for-developers

- 2018: ML, Fluent Design System, Mix Reality……
- 2019: WinUI, XAML, C++/winRT, sub-Linux, MSIX, Project Rome, webView2……
- 2020: WinUI 3.0 preview 2, Window 10X, CS/winRT, winrt-rs, docker in WSL, Windows AI, Project Reunion ……

AI 时代技术进化的速度越来越快

紧追时代/技术的步伐才能

不被时代淘汰……不被AI淘汰……

Windows 编程技术发展趋势展望

前端跨平台融合

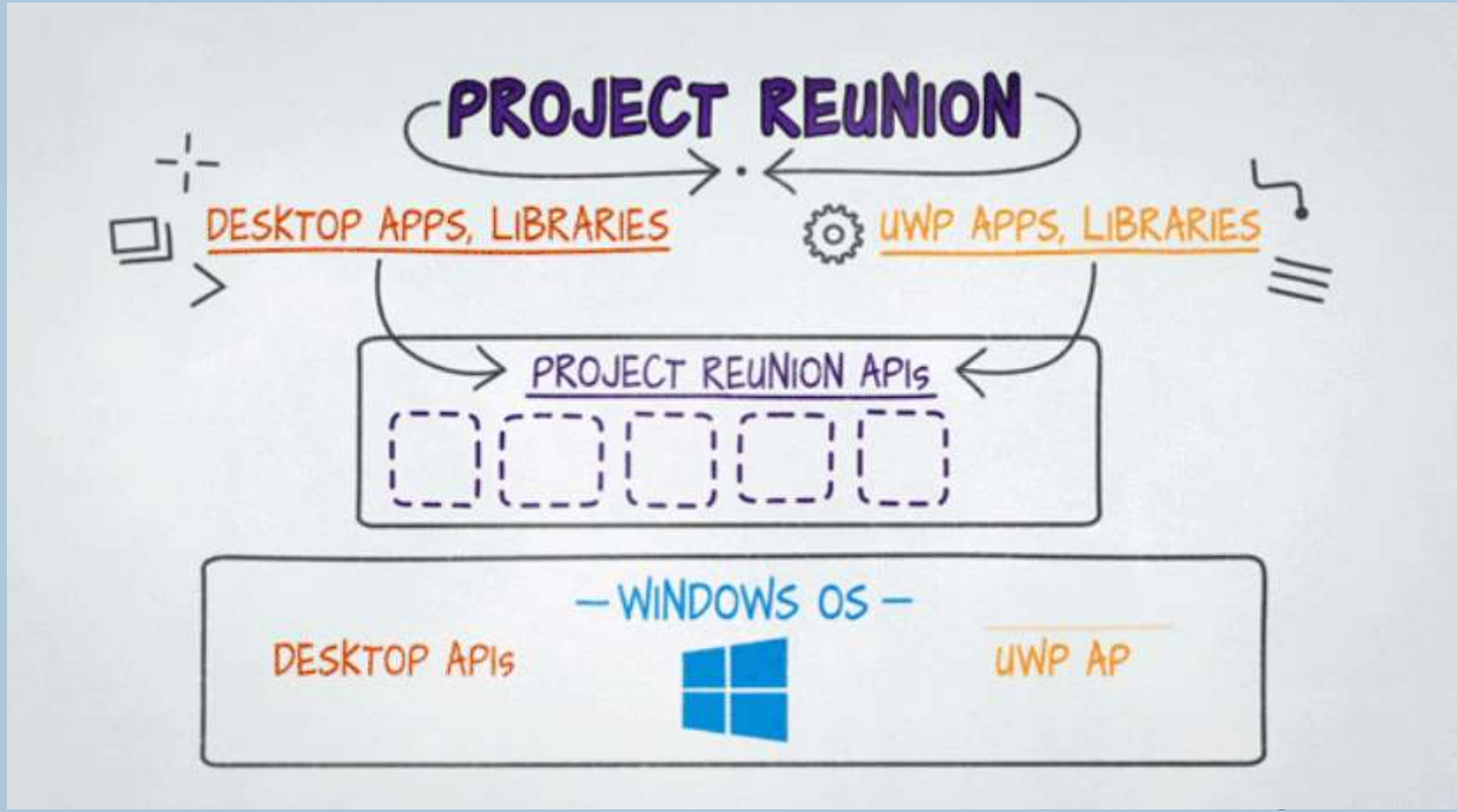
编码与设计分离

API COM 封装

AI 加持

实时协作

云端原生



....

MIUI的例子

eUnion

CD

C++, RUST

Scripts, C#, JAVA, ……

Windows 编程技术发展趋势展望

前端跨平台融合

编码与设计分离

API COM 封装

AI 加持

实时协作

云端原生

- **fusionware across platforms (GUI)**
  - **wine, docker, webView2, electron，Angular, vue，React Native, Qt……**
- **UI / UX design separating, VR/AR supported UX**
- **winRT projecting to different languages, project ReUnion**

MIUI的例子

UWP 改头换面 project ReUnion

- AppContainer 沙箱隔离环境，严控权限
- WinRT 新的Windows API，COM的进化版
- WinRT XAML 基于WinRT API框架的一套新的XAML UI，现在终于有了正式的名字，WinUI

Windows的主要特点

➢ 面向对象

窗口、菜单、事件皆是对象

对话框与各种控件是一些特殊的窗口

对界面元素的操作和消息/事件的处理都按照对象进行。对这些对象的属性和操作，由相关数据结构和API调用函数（或由其封装的MFC和.NET框架中的类）提供。

Windows的主要特点

- 面向对象

数据结构

```
typedef struct tagCREATESTRUCTA {
    LPVOID      lpCreateParams;
    HINSTANCE   hInstance;
    HMENU       hMenu;
    HWND        hwndParent;
    int         cy;
    int         cx;
    int         y;
    int         x;
    LONG        style;
    LPCSTR      lpszName;
    LPCSTR      lpszClass;
    DWORD       dwExStyle;
} CREATESTRUCTA, *LPCREATESTRUCTA;
typedef struct tagCREATESTRUCTW {
    LPVOID      lpCreateParams;
    HINSTANCE   hInstance;
    HMENU       hMenu;
    HWND        hwndParent;
    int         cy;
    int         cx;
    int         y;
    int         x;
    LONG        style;
    LPCWSTR     lpszName;
    LPCWSTR     lpszClass;
    DWORD       dwExStyle;
} CREATESTRUCTW, *LPCREATESTRUCTW;
#ifdef UNICODE
typedef CREATESTRUCTW CREATESTRUCT;
typedef LPCREATESTRUCTW LPCREATESTRUCT;
#else
typedef CREATESTRUCTA CREATESTRUCT;
typedef LPCREATESTRUCTA LPCREATESTRUCT;
#endif // UNICODE
```

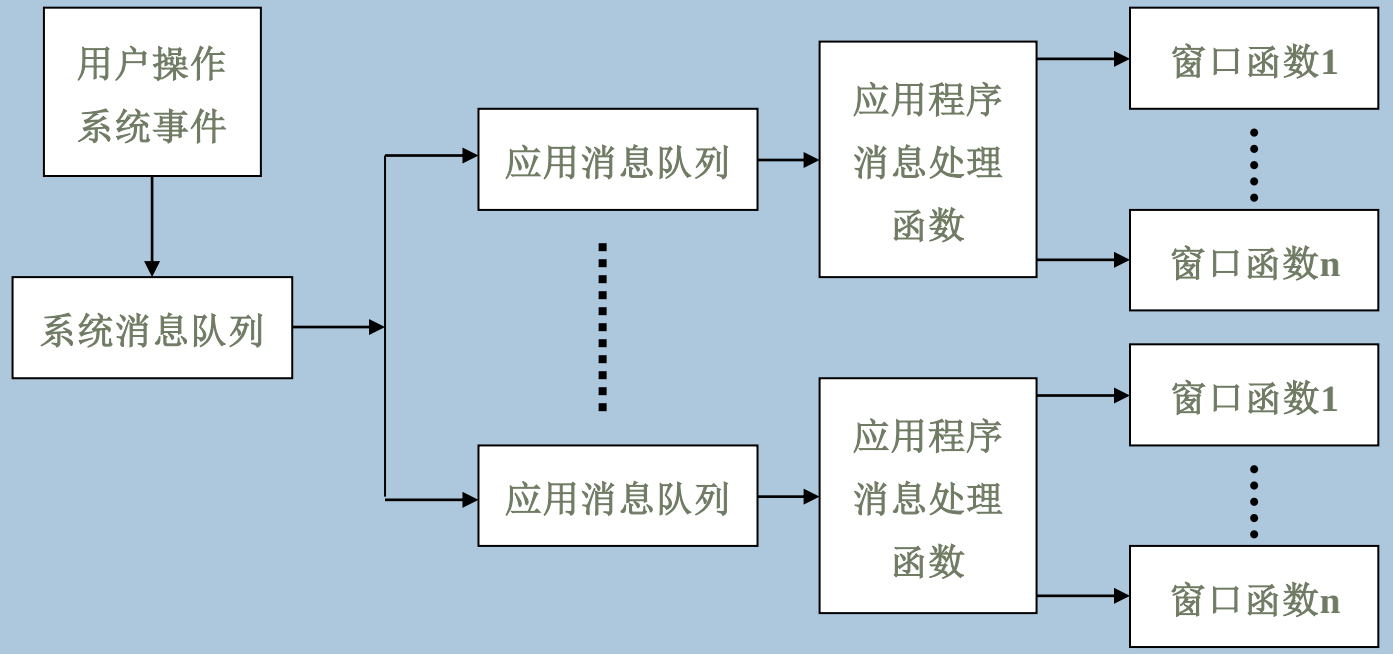
- 窗口、菜单、事件皆是对象

API 函数

```
WINUSERAPI
HWND
WINAPI
CreateWindowExA(
    _In_ DWORD dwExStyle,
    _In_opt_ LPCSTR lpClassName,
    _In_opt_ LPCSTR lpWindowName,
    _In_ DWORD dwStyle,
    _In_ int X,
    _In_ int Y,
    _In_ int nWidth,
    _In_ int nHeight,
    _In_opt_ HWND hWndParent,
    _In_opt_ HMENU hMenu,
    _In_opt_ HINSTANCE hInstance,
    _In_opt_ LPVOID lpParam);
```

Windows的主要特点

- 面向对象
- 消息/事件驱动



Windows的主要特点

- 面向对象
- 消息/事件驱动
- 资源共享与数据交换

Windows的主要特点

- 面向对象
- 消息/事件驱动
- 资源共享与数据交换

抢先式多任务操作系统

应用程序之间共享系统资源

Windows 编程时，必须时刻记住尽早释放不再使用的系统资源

避免系统资源耗尽而造成效率急剧降低

Windows的主要特点

- 面向对象
- 消息/事件驱动
- 资源共享与数据交换
- 设备无关的GDI

Windows提供了与设备无关的GDI。

应用程序可以通过调用GDI函数，在不同显卡、打印机和显示器上输出图形或文本

1.2.1 Windows程序开发流程

- 使用微软 Project 进行项目管理（甘
- 选取源码管理工具 (GitHub/Azure,
- 使用 IDE (Integrated Developm
  环境) 编写源码，通常采用 VS Code
  通常采用Keil µVision
- 在 IDE 中进行 Debug （F5, F9, F1
- 编写单元测试代码进行测试，测试完
- 编写集成测试
- 稳定版本发布，更新master分支

用)
发
开发

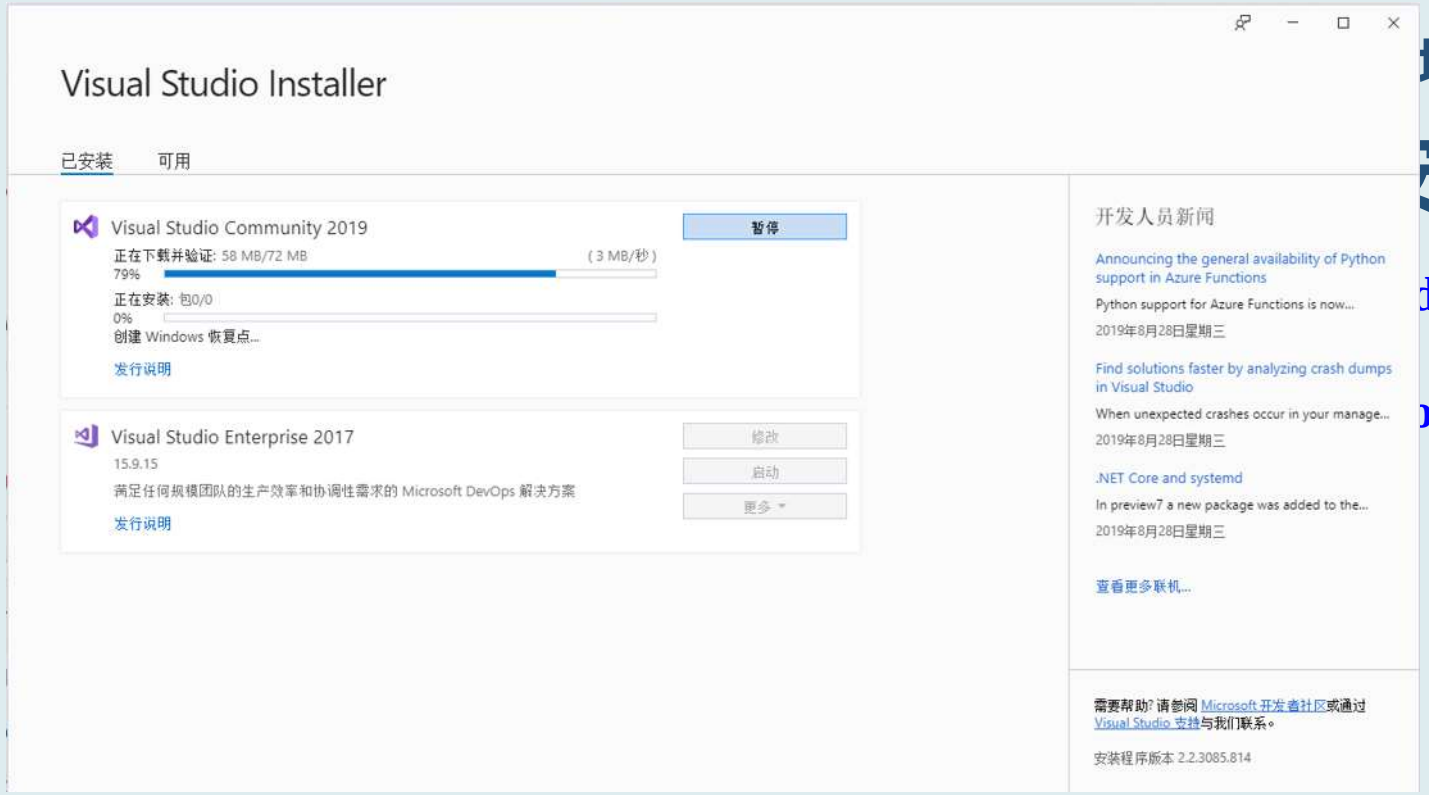
GitHub Action CI/CD

The architecture of a dev team, in sil
Code review
Meeting minutes

Visual S


正在修改 — Visual Studio Community 2019 — 16.2.3
工作负载
单个组件
语言包
安装位置
使用 C++ 的移动开发
使用 C++ 对 iOS、Android 或 Windows 生成跨平台应用程序。
使用 C++ 的游戏开发
充分使用 C++ 生成由 DirectX、Unreal 或 Cocos2d 提供技术支持的专业游戏。
其他工具集 (6)
数据存储和处理
使用 SQL Server、Azure Data Lake 或 Hadoop 连接、开发和测试数据解决方案。
数据科学和分析应用程序
用于创建数据科学应用程序的语言和工具，包括 Python 和 F#。
Visual Studio 扩展开发
为 Visual Studio 创建加载项和扩展，包括新命令、代码分析器和工具窗口。
Office/SharePoint 开发
使用 C#、VB 和 JavaScript 创建 Office 和 SharePoint 外接程序、SharePoint 解决方案和 VSTO 外接程序。
使用 C++ 的 Linux 开发
创建和调试在 Linux 环境中运行的应用程序。
.NET Core 跨平台开发
使用 .NET Core、ASP.NET Core、HTML/JavaScript 和包括 Docker 支持的容器生成跨平台应用程序。
安装详细信息
Visual Studio 核心编辑器
Python 开发 *
Node.js 开发
.NET 桌面开发 *
使用 C++ 的桌面开发
通用 Windows 平台开发
数据科学和分析应用程序 *
可选
F# 桌面语言支持
Python 语言支持
Python miniconda
Python Web 支持
Python 本机开发工具
单个组件 *

1.2.2 Windows程序开发IDE



境）

装

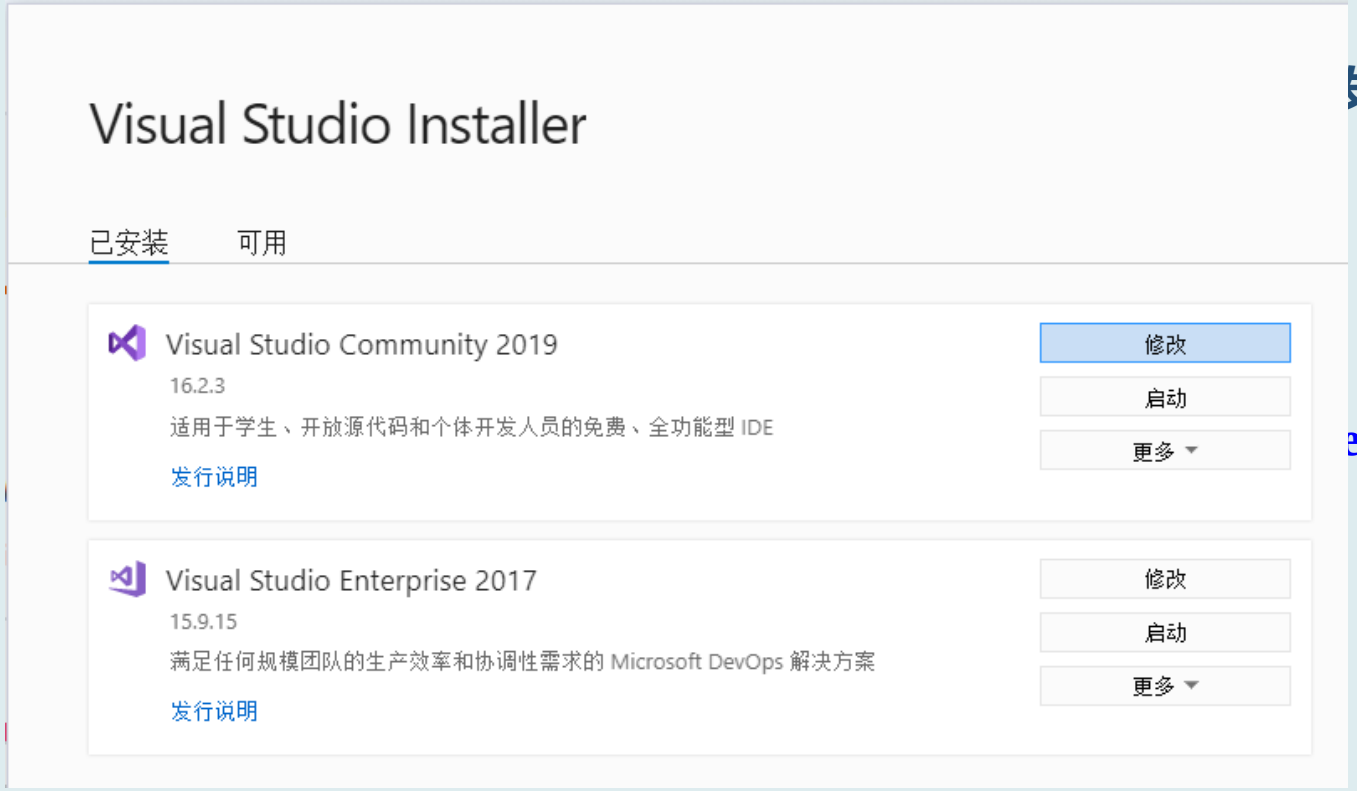
date

ownload

VisualStudio 是 Windows 程序员

应该必须掌握的一款优秀的 IDE

1.2.2 Windows程序开发IDE



发环境）

安装

Update

et/download

VisualStudio 是 Windows 程序员

应该必须掌握的一款优秀的 IDE

Visual Studio Community 2019 extensions

- Qt Visual Studio Tools
- C++/WinRT
- Windows Template Studio
- Visual Studio IntelliCode
- GitHub extension for Visual Studio
- Gitee extension for Visual Studio
- Microsoft Visual Studio Installer Projects
- R tools ? … python
- Markdown Editor

1.2.3 Windows编程语言的选择

- 在Visual Studio提供的各种语言工具中，只有用Visual C++才能编写传统的Windows应用程序。VC也是VS中唯一的一种可以同时[混合]编写非托管（API与MFC/ATL）程序和托管（.NET）程序的工具，
- VS中的其他语言工具（如C#、VB和F# 等）则只能编写.NET环境下的托管程序
- 本课程同时使用 C++ 与 C# 来进行教学, python, node.js
- 参考阅读材料 https://docs.microsoft.com/en-us/windows/apps/desktop/choose-your-platform

开发效率与运行效率常常是一对矛盾

多多动手练习是学习本课程的

唯一诀窍

Windows编程语言

C++ 越来越精英化 远离生产专注研发

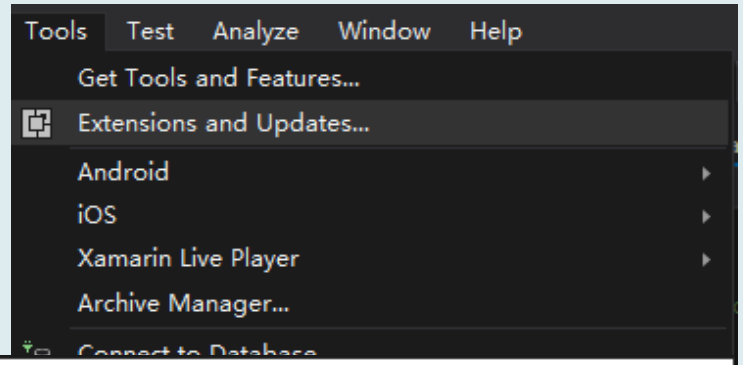
- 建议选修 C++ 课程，随着计算智能的进步C++大有用武之地
- C#是本课程的先修课程，建议选修或自学
- 逐步熟练掌握XAML

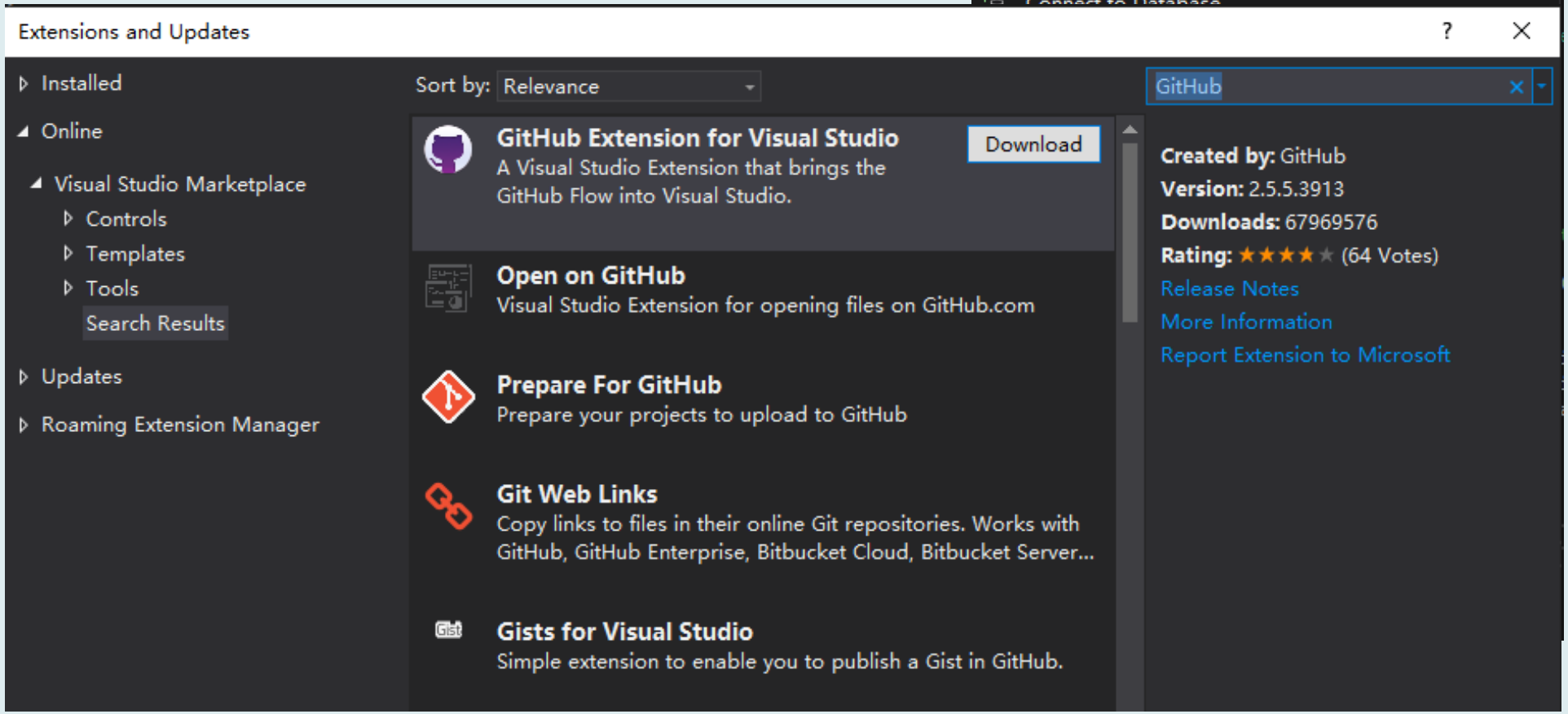
"C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off". Yes, I said something like that (in 1986 or so). What people tend to miss, is that what I said there about C++ is to a varying extent true for all powerful languages. As you protect people from simple dangers, they get themselves into new and less obvious problems. Someone who avoids the simple problems may simply be heading for a not-so-simple one. One problem with very supporting and protective environments is that the hard problems may be discovered too late or be too hard to remedy once discovered. Also, a rare problem is harder to find than a frequent one because you don't suspect it.

Bjarne Stroustrup http://www.stroustrup.com/bs_faq.html

1.2.4 用gitHub做代码管理

- Tools => Extensions and Updates
- 在Online中搜索GitHub
- 点击下载GitHub Extension for VS





1.3 WINDOWS Form与WPF应用程序

- homework: surf the following web pages

https://docs.microsoft.com/en-us/windows/desktop/rpc/the-programming-model
http://programmingexamples.wikidot.com/windows-programming-model

- Windows编程模型有较大的改变，云计算快速普及的时代MS现在主推Azure
  - https://azure.microsoft.com/zh-cn/overview/what-is-azure/
  - 传统的桌面开发模式依然有市场，但在快速向云端迁移
  - 云计算、移动计算、边缘计算、桌面计算、普适计算将群雄逐鹿
  - Win10 在不断发展，新的Windows编程模型依然在逐渐形成过程中

VS中Windows 应用程序类型

应用程序类型与开发语言有一定的关系

- VC++
  - 基于控制台的应用程序
  - 基于对话框的应用程序
  - 单文档应用程序
  - 多文档应用程序
  - 基于html的应用程序

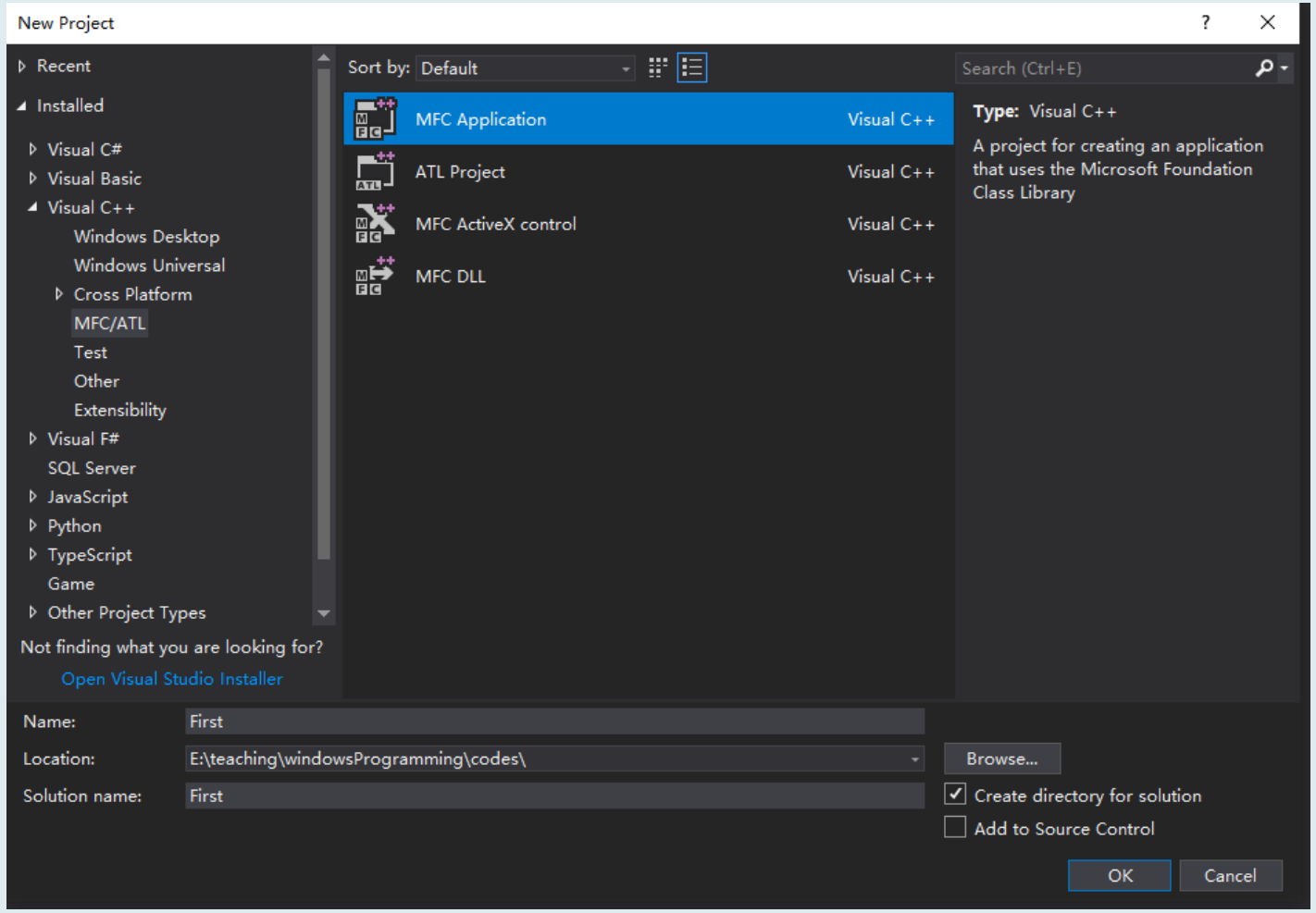
- C#
  - 控制台应用程序
  - Windows窗体应用程序
  - WPF应用程序
  - ASP.NET Web应用程序
  - WCF服务应用程序
  - ……

基于对话框的应用程序

□ 安装MFC

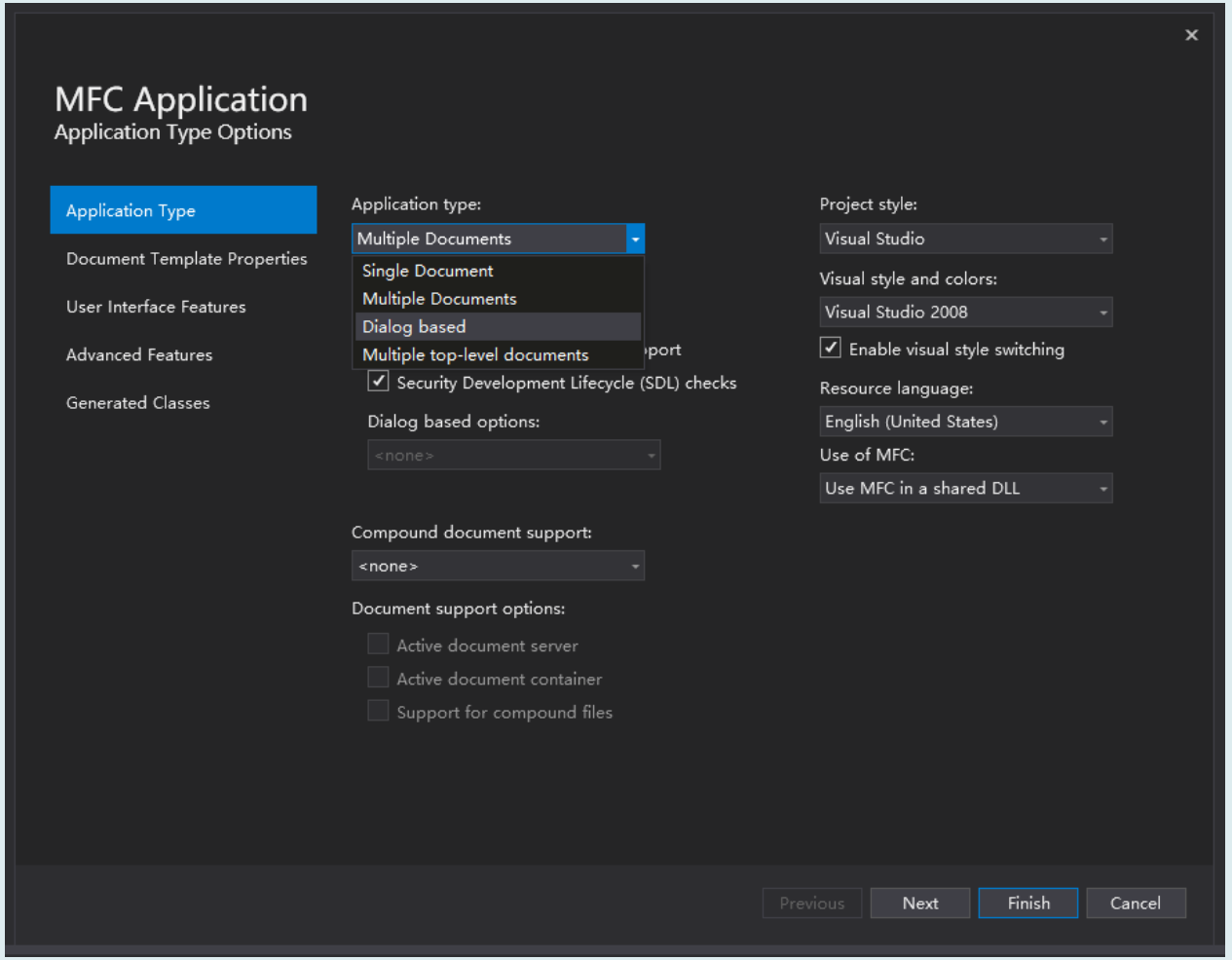
基于对话框的应用程序

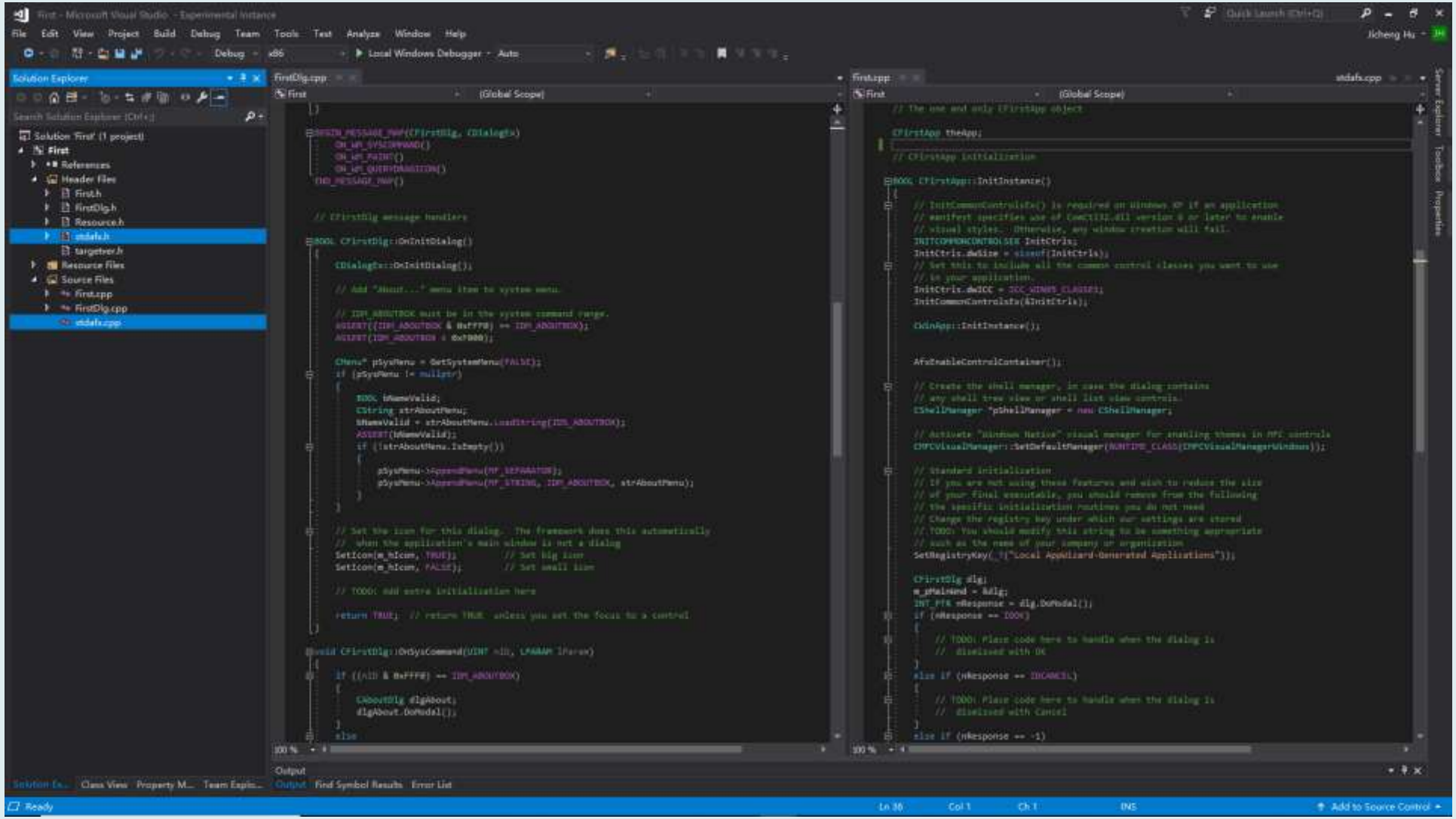
- File => new => Project => Visual C++ => MFC/ATL => MFC Application



基于对话框的应用程序

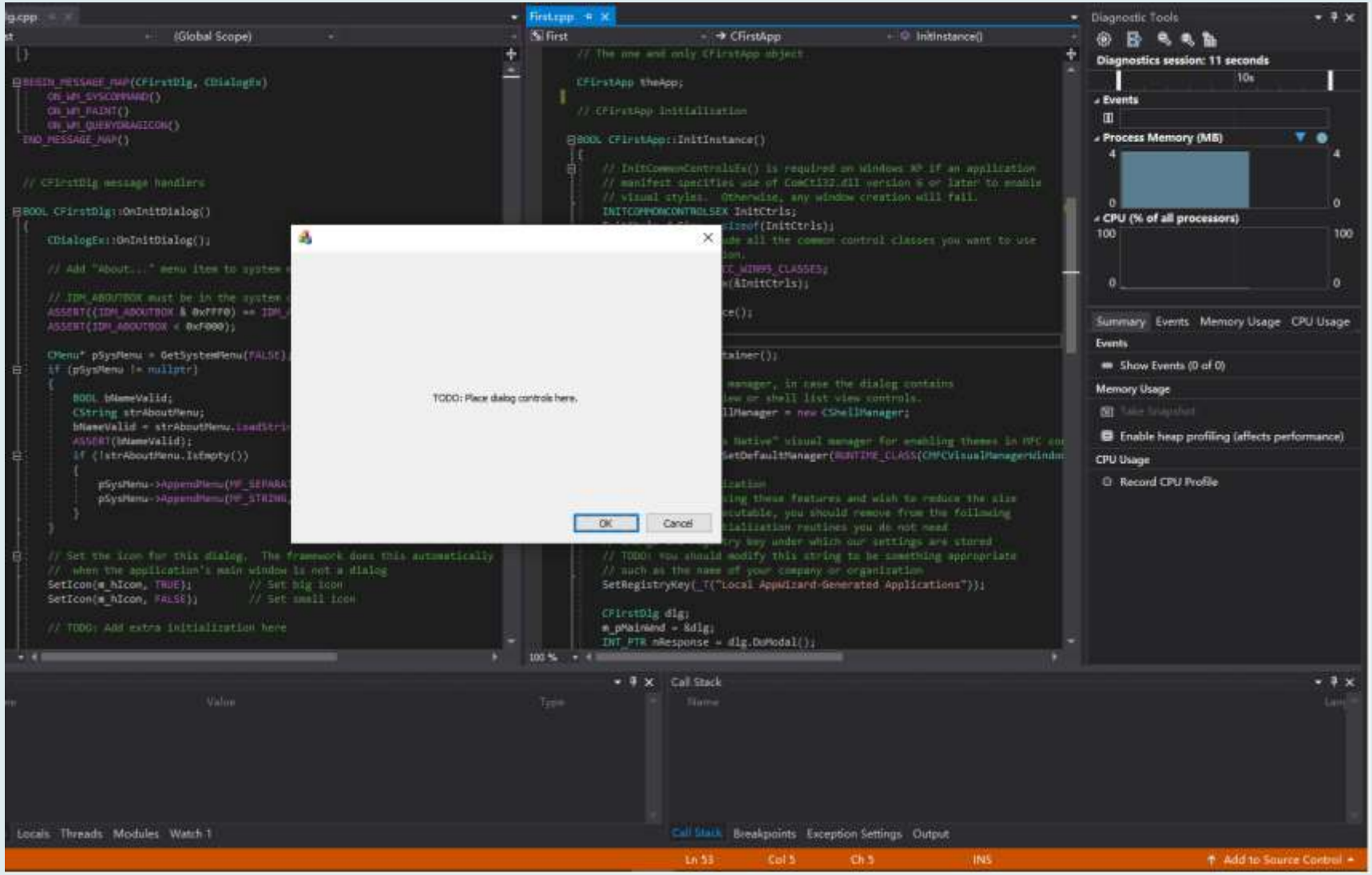
- File => new => Project => Visual C++ => MFC/ATL => MFC Application => Dialog based



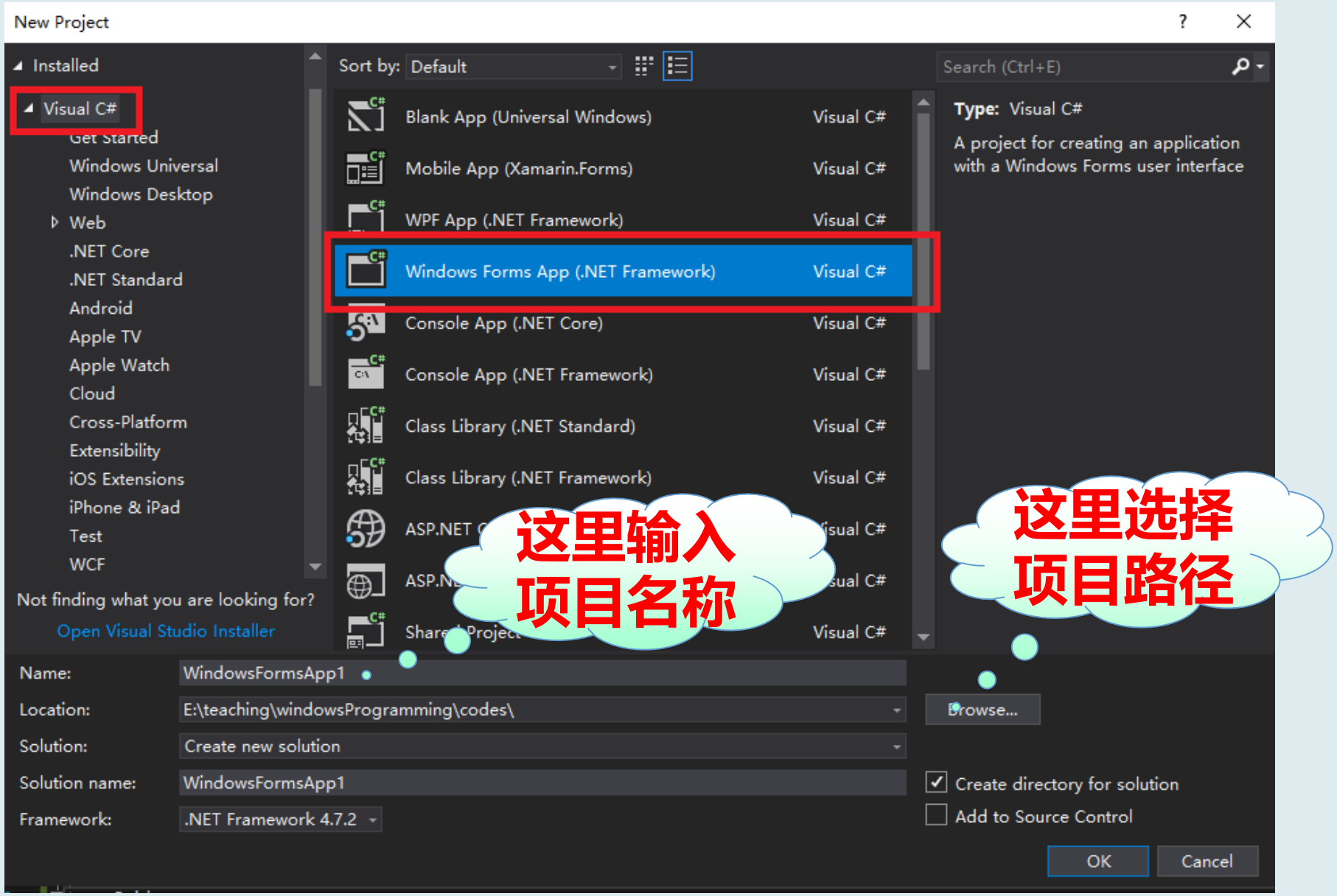


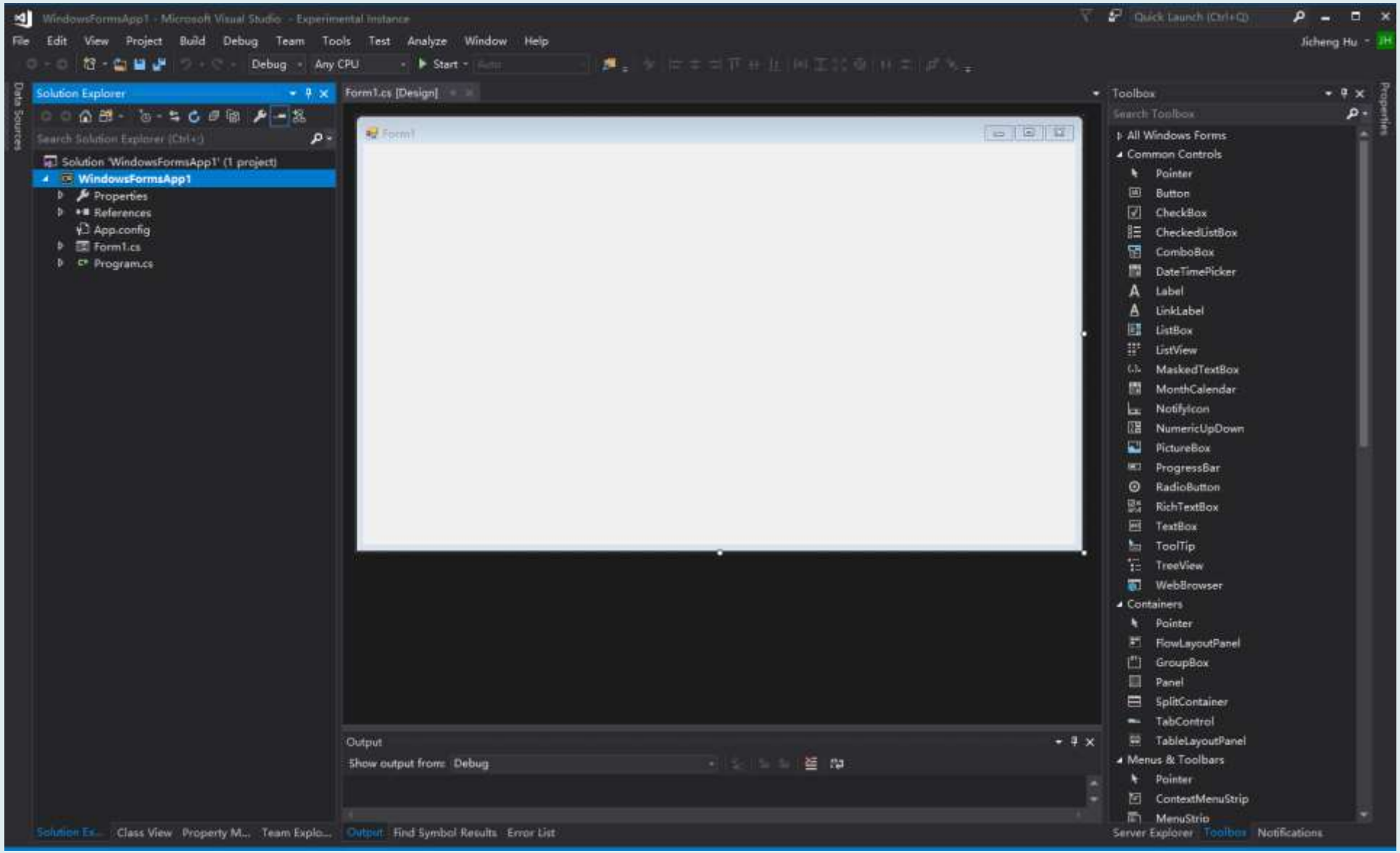
基于对话框的应用程序

- F7编译 => F5 start debugging



Windows窗体应用程序



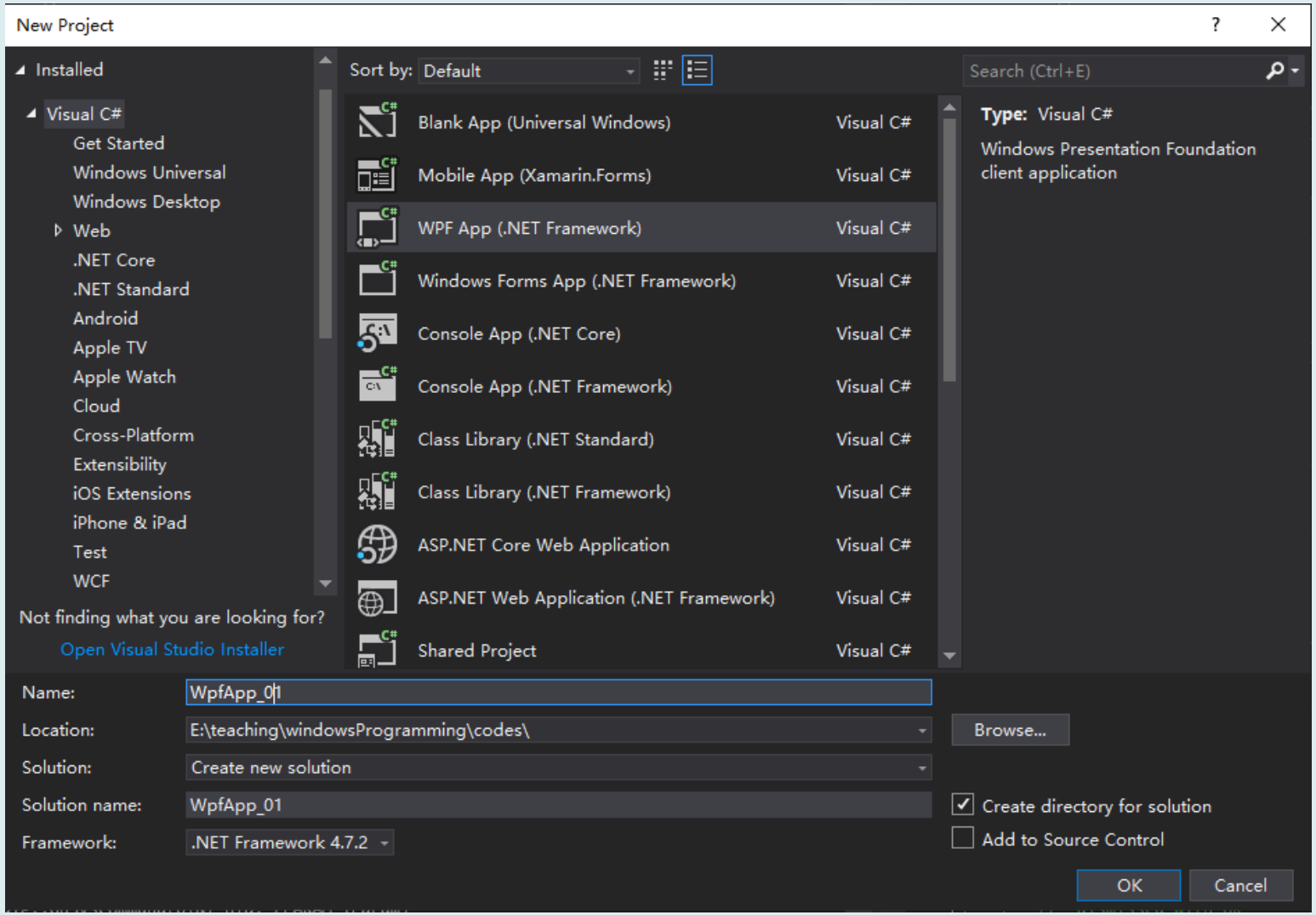

WindowsFormsApp1 - Microsoft Visual Studio - Experimental Instance
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU Start
Solution Explorer
Solution 'WindowsFormsApp1' (1 project)
WindowsFormsApp1
Properties
References
App.config
Form1.cs
Program.cs
Form1.cs [Design]
Form1
Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Containers
Pointer
FlowLayoutPanel
GroupBox
Panel
SplitContainer
TabControl
TableLayoutPanel
Menus & Toolbars
Pointer
ContextMenuStrip
MenuStrip
Output
Show output from: Debug
Class View
Property M...
Team Explo...
Find Symbol Results
Error List
Server Explorer
Toolbox
Notifications

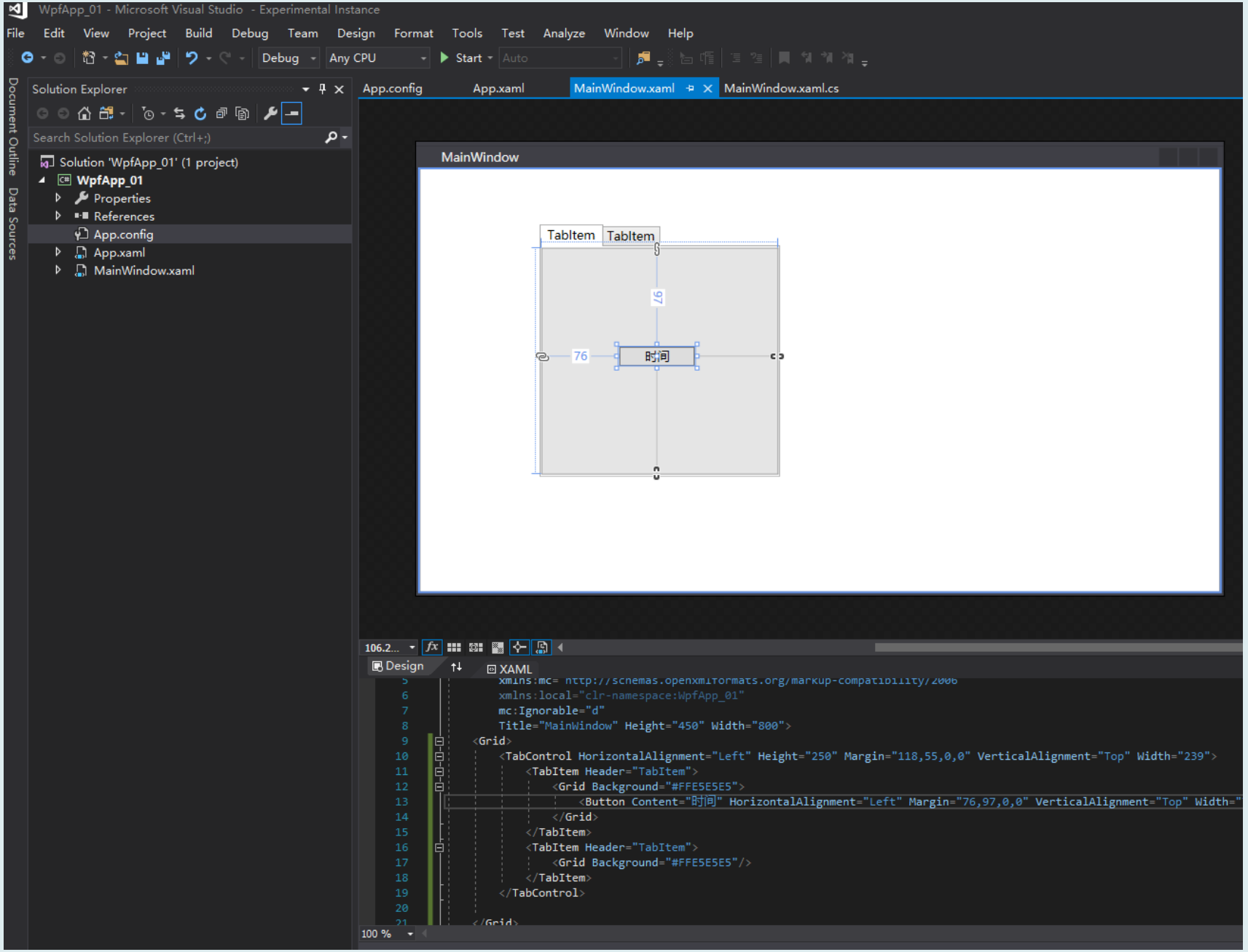
WPF应用程序

- Windows Presentation Foundation，用于生成较好视觉体验的 Windows 应用程序
- 既可创建独立桌面应用程序，也可创建浏览器承载的应用程序
- WPF 的核心是一个与分辨率无关并且基于向量的呈现引擎
- WPF 包含在 .NET Framework 中，作为 .NET Framework 的一个子集存在，其类型大多位于 System.Windows 命名空间
- 界面设计使用可扩展应用程序标记语言 (XAML)
- 使用C# 或 VB实例化类、设置属性、调用方法以及处理事件

WPF应用程序

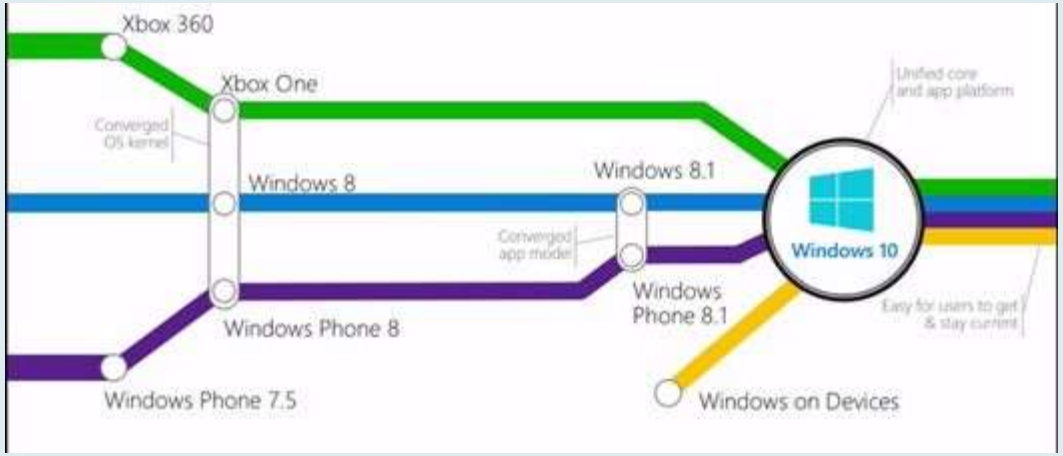
程序界面：基于XML的XAML语言定制；　　程序逻辑：C#语言实现




WpfApp_01 - Microsoft Visual Studio - Experimental Instance
File Edit View Project Build Debug Team Design Format Tools Test Analyze Window Help
Debug Any CPU Start Auto
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WpfApp_01' (1 project)
WpfApp_01
Properties
References
App.config
App.xaml
MainWindow.xaml
App.config App.xaml MainWindow.xaml MainWindow.xaml.cs
MainWindow
TabItem TabItem
97
76
时间
Design XAML
xmlns:mc= http://schemas.openxmlformats.org/markup-compatibility/2006
xmlns:local="clr-namespace:WpfApp_01"
mc:Ignorable="d"
Title="MainWindow" Height="450" Width="800">
<Grid>
<TabControl HorizontalAlignment="Left" Height="250" Margin="118,55,0,0" VerticalAlignment="Top" Width="239">
<TabItem Header="TabItem">
<Grid Background="#FFE5E5E5">
<Button Content="时间" HorizontalAlignment="Left" Margin="76,97,0,0" VerticalAlignment="Top" Width="
</Grid>
</TabItem>
<TabItem Header="TabItem">
<Grid Background="#FFE5E5E5"/>
</TabItem>
</TabControl>
</Grid>
100 %

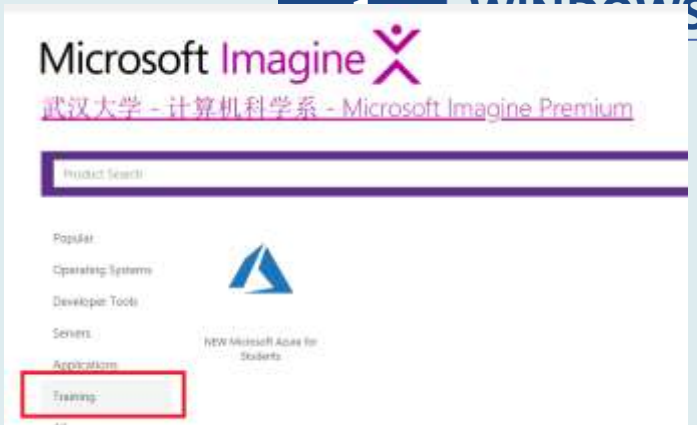
1.4 UWP与Fluent Design

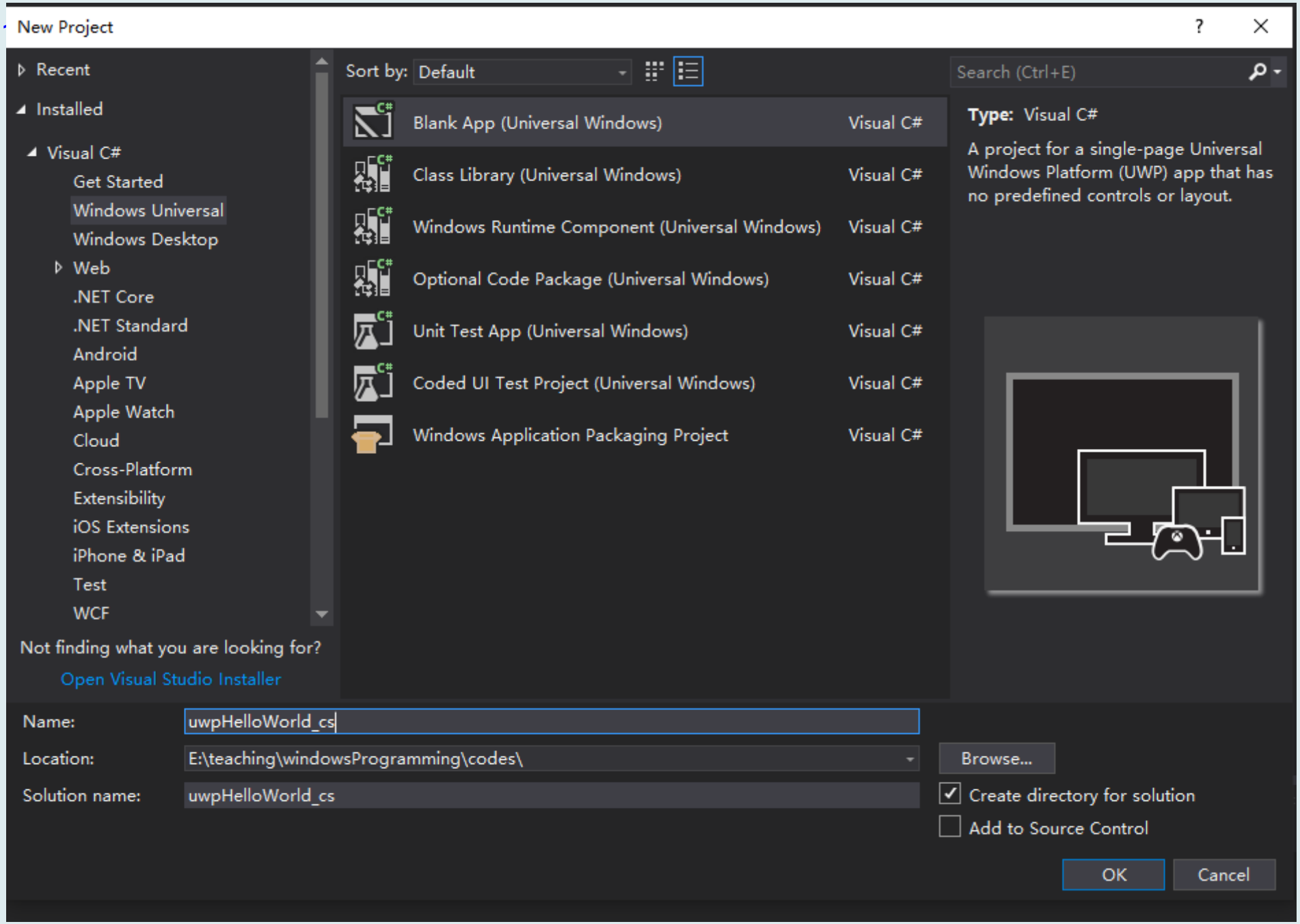
- 近年来WINDOWS编程技术发展迅速
- Universal Windows Platform（通用Windows平台）
  - 微软新提出的一种应用种类：通过统一的开发平台，使开发者针对其开发的代码在多种不同的设备上实现共享，并为用户提供统一的使用体验
  - Windows 10 应用商店里所有的程序都是UWP应用
  - UWP基于.NET Framework，也可用VC++开发
  - 也可采用基于Xamarin的.NET框架，完成对安卓、iOS的跨平台支持
  - 桌面应用程序转换器(Desktop Application Converter)，可以把现有的桌面应用程序（.NET 4.6.1 或 Win32）转换成 UWP程序



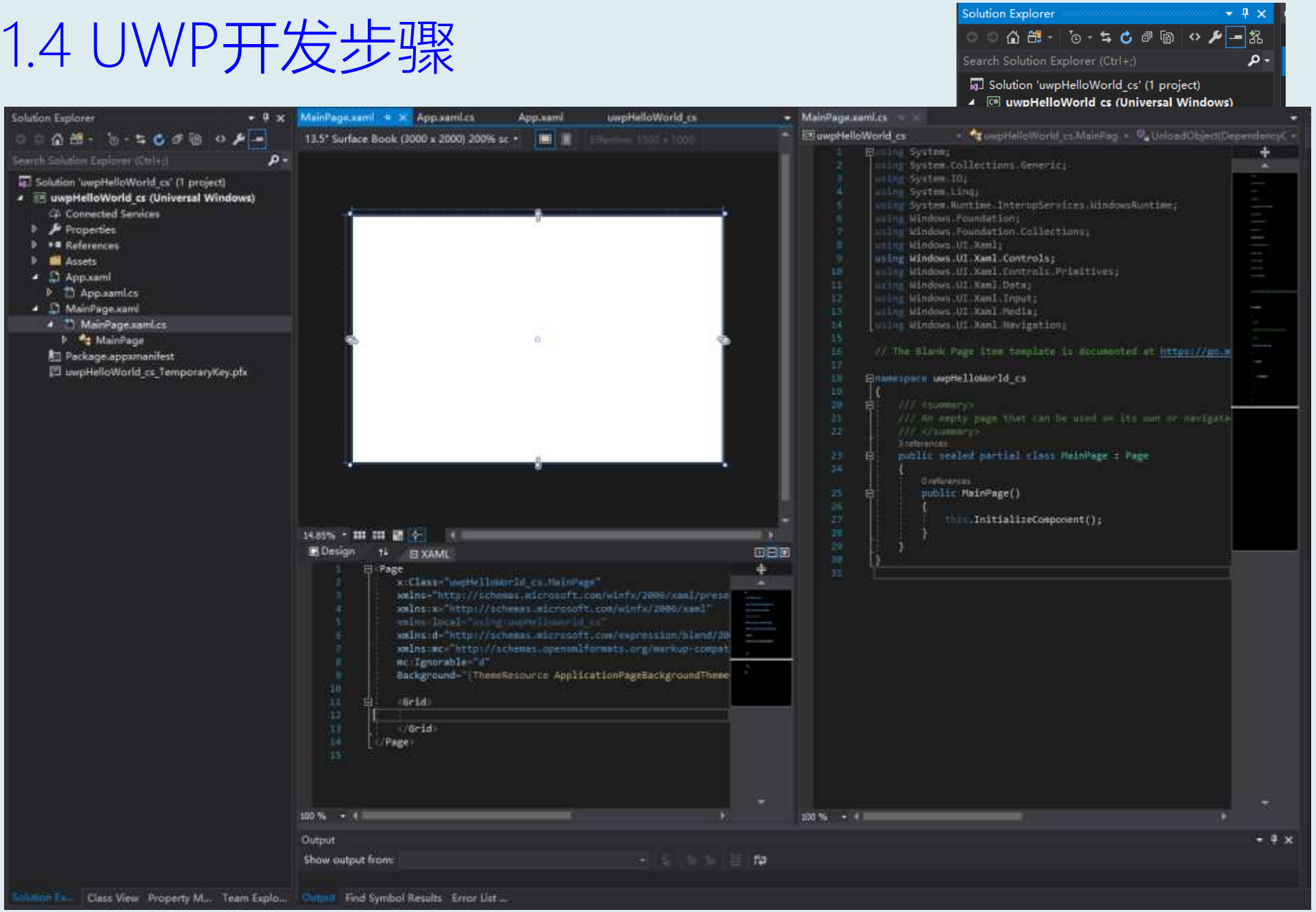
1.4.1 UWP开发步骤

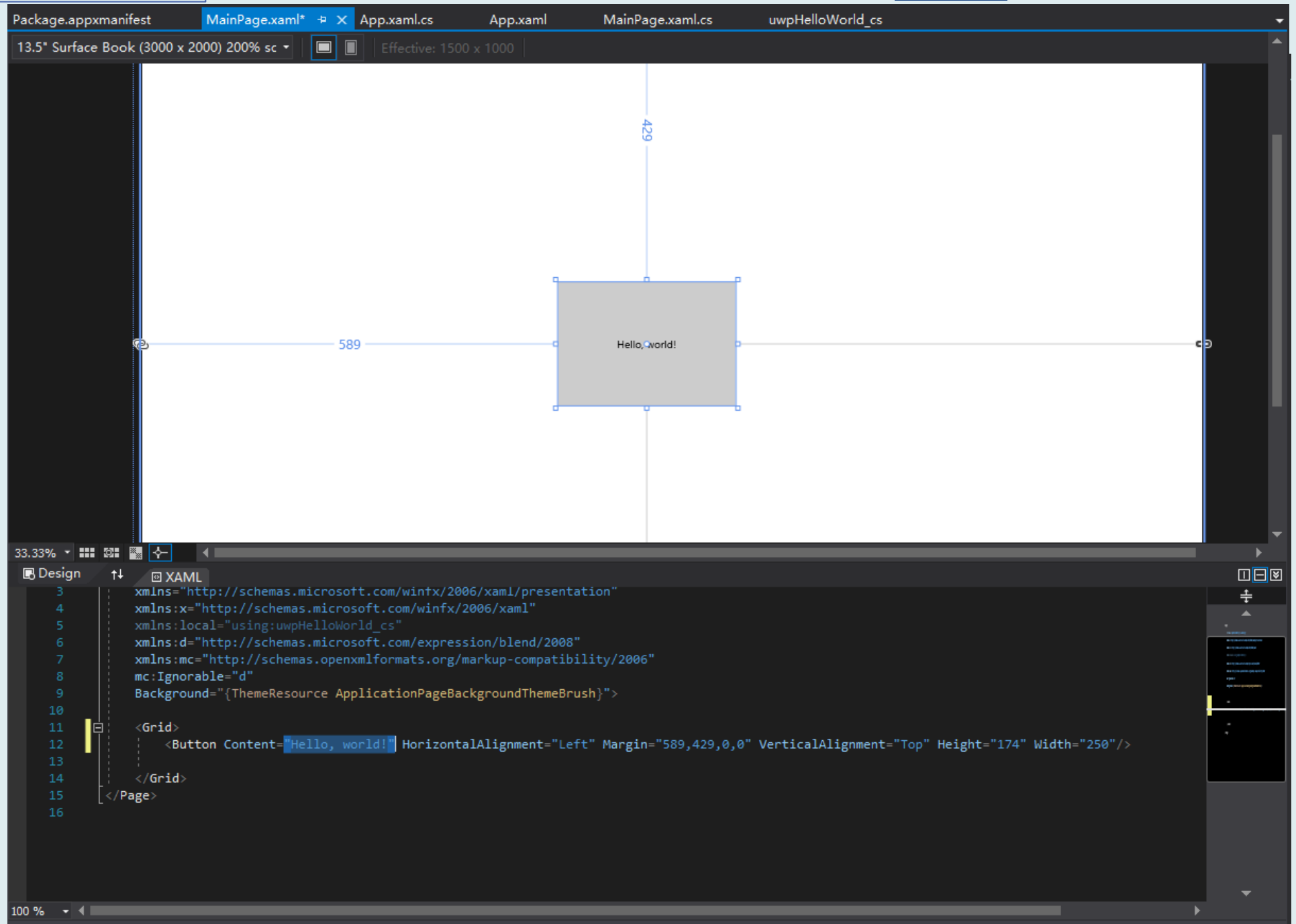
- 注册微软开发者账户
  - 计算机学院dreamSpark点击training




New Project
Recent
Installed
Visual C#
Get Started
Windows Universal
Windows Desktop
Web
.NET Core
.NET Standard
Android
Apple TV
Apple Watch
Cloud
Cross-Platform
Extensibility
iOS Extensions
iPhone & iPad
Test
WCF
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
Blank App (Universal Windows) Visual C#
Class Library (Universal Windows) Visual C#
Windows Runtime Component (Universal Windows) Visual C#
Optional Code Package (Universal Windows) Visual C#
Unit Test App (Universal Windows) Visual C#
Coded UI Test Project (Universal Windows) Visual C#
Windows Application Packaging Project Visual C#
Type: Visual C#
A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.
Name: uwpHelloWorld_cs
Location: E:\teaching\windowsProgramming\codes\
Solution name: uwpHelloWorld_cs
Browse...
Create directory for solution
Add to Source Control
OK
Cancel

1.4 UWP开发步骤


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'uwpHelloWorld_cs' (1 project)
uwpHelloWorld_cs (Universal Windows)
Connected Services
Properties
References
Assets
App.xaml
App.xaml.cs
MainPage.xaml
MainPage.xaml.cs
MainPage
Package.appxmanifest
uwpHelloWorld_cs_TemporaryKey.pfx
MainPage.xaml
App.xaml.cs
App.xaml
uwpHelloWorld_cs
13.5" Surface Book (3000 x 2000) 200% sc
Design
XAML
MainPage.xaml.cs
Output
Show output from:
Class View
Property M...
Team Explo...
Find Symbol Results
Error List ...

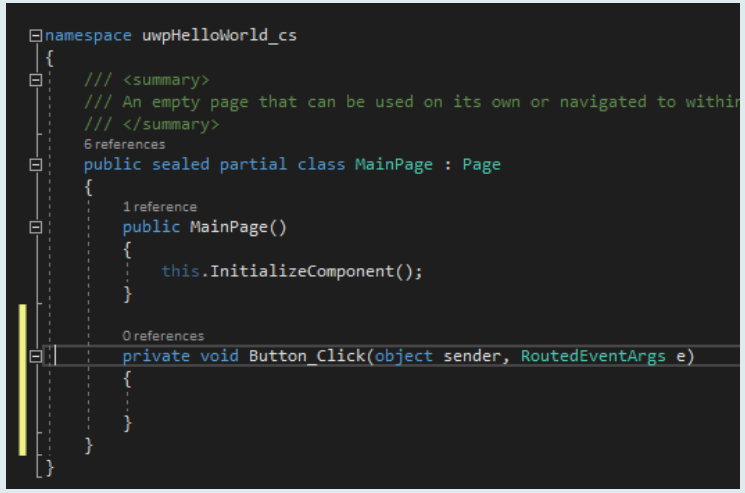

Package.appxmanifest
MainPage.xaml*
App.xaml.cs
App.xaml
MainPage.xaml.cs
uwpHelloWorld_cs
13.5" Surface Book (3000 x 2000) 200% sc
Effective: 1500 x 1000
429
589
Hello, world!
33.33%
Design
XAML
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:local="using:uwpHelloWorld_cs"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d"
Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
<Grid>
<Button Content="Hello, world!" HorizontalAlignment="Left" Margin="589,429,0,0" VerticalAlignment="Top" Height="174" Width="250"/>
</Grid>
</Page>
100 %

1.4 UWP开发步骤 – 事件处理

- 双击设计画布中的按钮控件， Visual Studio 会自动为该按钮创建事件处理方法
  - private void Button_Click (object sender, RoutedEventArgs e)
  - 更改该方法：

```
private async void Button_Click(object sender, RoutedEventArgs e)
{
        MediaElement mediaElement = new MediaElement();
        var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();

        Windows.Media.SpeechSynthesis.SpeechSynthesisStream stream = await
                synth.SynthesizeTextToStreamAsync("Hello, World!");
        mediaElement.SetSource(stream, stream.ContentType);
        mediaElement.Play();
}
```



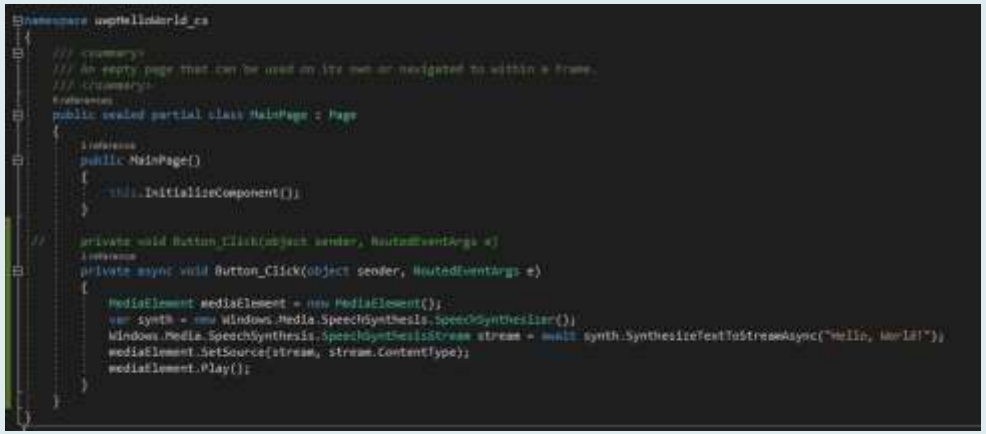
使用 Windows API 创建一个语音合成对象

提供给该对象一些要说的文本

有关使用 SpeechSynthesis 的详细信息

参阅 SpeechSynthesis 命名空间文档

https://docs.microsoft.com/en-us/uwp/api/Windows.Media.SpeechSynthesis



- F5、F7
  - 点击Hello, world按钮, 出现Text To Speech效果

Voice synthesis

Texture synthesis

近2年热点之一

1.4.2 Fluent Design System

□ 参考阅读网页

- FLUENT官网 https://www.microsoft.com/design/fluent/
- https://docs.microsoft.com/en-us/windows/uwp/design/fluent-design-system/index

□ 五大核心元素：

- Light（光感）
- Depth（深度）
- Motion（动画）
- Material（材质）
- Scale（缩放）

硬件成本的快速下降将极大推动技术的进步与普及

nVidia 的实时光线追踪技术与

机器学习使Fluent的前景充满遐想

Design toolkits for Fluent Design

- 参考阅读网页
  - https://docs.microsoft.com/en-us/windows/uwp/design/downloads/index
- Figma toolkit
- Sketch toolkit

个人观点：

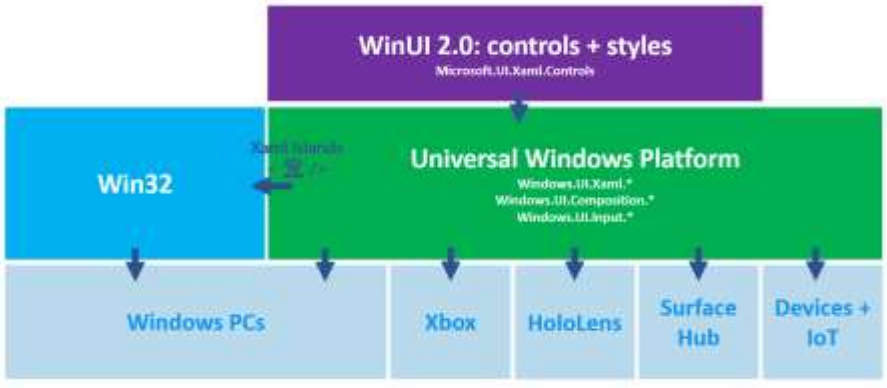
Fluent 的出现意味着coding与designing的分离

并将逐步发展到 UI 与 UX 的分离

未来windows软件的生产将是： 编码+设计

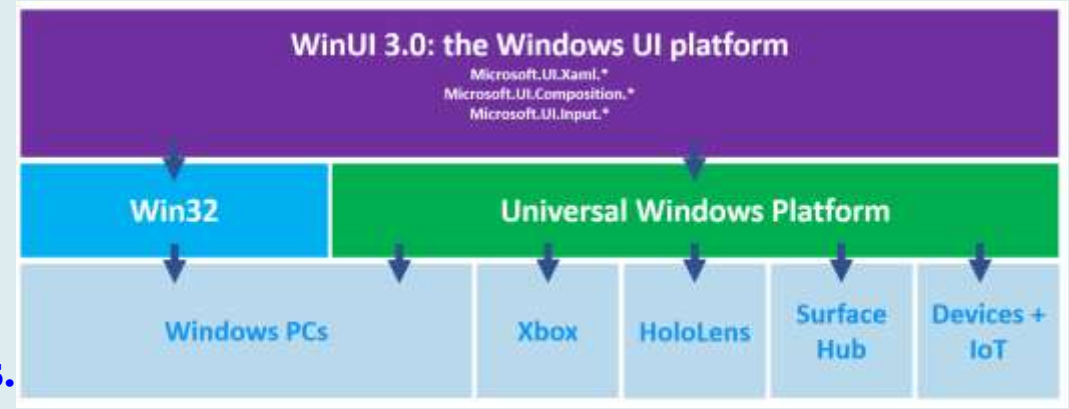
技术+艺术

Evolution of WinUI



By completely decoupling XAML, composition, and input APIs from the Windows 10 SDK, the scope of WinUI 3 includes the full Windows 10 native UI platform.

All new XAML features will eventually ship as part of WinUI. The existing UWP XAML APIs that ship as part of the OS will no longer receive new feature updates.



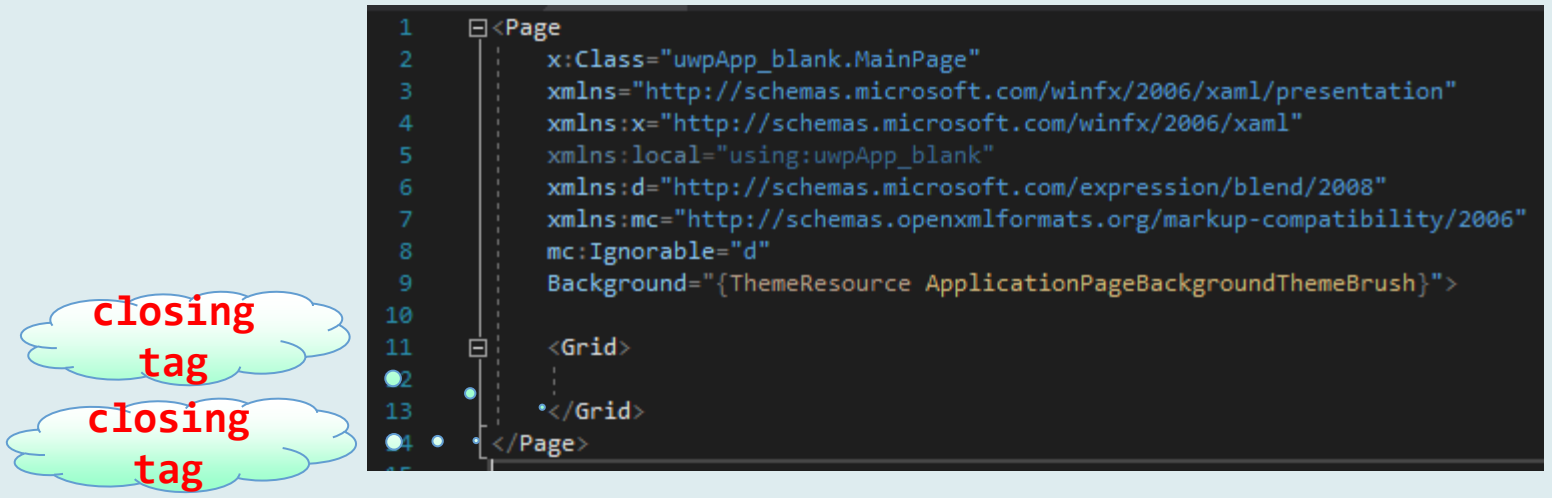
https://docs.microsoft.com/en-us/windows/apps/winui/

从捆绑到釜底抽薪!

1.5.1 XAML

⇐ WHY?

- stands for eXtensible Application Markup Language
- is a type of XML



- nodes (also known as tags, or elements)
  - Page - has numerous attributes which help to further describe the element
  - Grid
- Nested Elements - The <Page></Page> contain the <Grid></Grid> element

1.5.2 WinUI

MIDL 3.0 is a particularly convenient way to define C++/WinRT runtime classes.

https://docs.microsoft.com/en-us/uwp/midl-3/troubleshooting

- Windows UI 库是使用 C++/WinRT 编写的
- 下面详述如何向 C++/WinRT 项目添加对 Windows UI (WinUI) 库的支持

https://docs.microsoft.com/zh-cn/windows/uwp/cpp-and-winrt-apis/simple-winui-example?cid=kerryherger

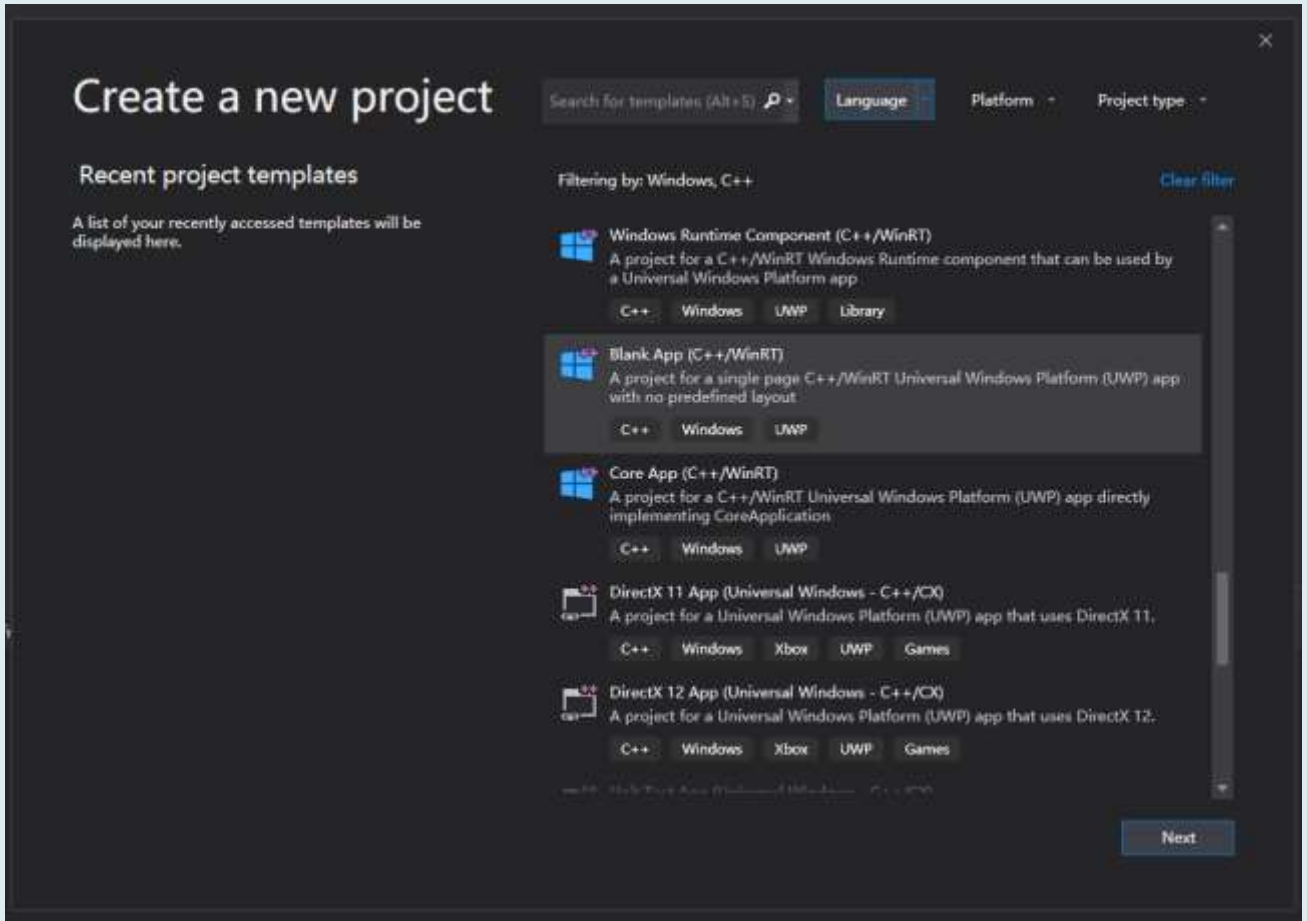
Windows Runtime components with C++/WinRT

https://docs.microsoft.com/en-us/windows/uwp/winrt-components/create-a-windows-runtime-component-in-cppwinrt

1.5.2 WinUI

C++/WinRT WinUI Example

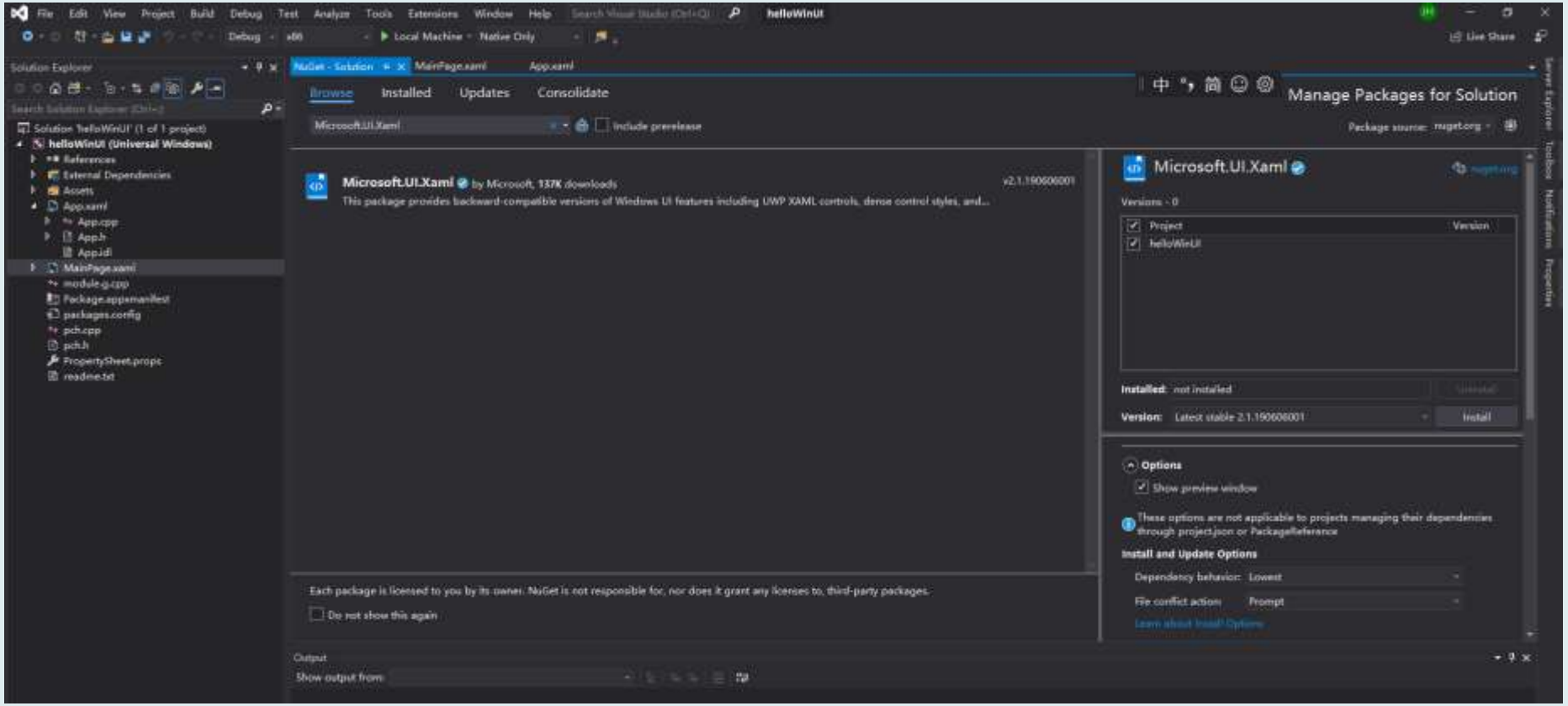
- Create a Blank App (helloWinUI)
  - In Visual Studio, create a new project using the Blank App (C++/WinRT) project template, and name it HelloWinUI



1.5.2 WinUI

◻ Install the Microsoft.UI.Xaml NuGet package

Click Project > Manage NuGet Packages... > Browse, search Microsoft.UI.Xaml in the search box, and click to install the package into your project



1.5.2 WinUI

- Declare WinUI application resources

Open App.xaml and paste the following markup between the existing opening and closing Application tags.

```
<Application.Resources>
    <XamlControlsResources xmlns= "using:Microsoft.UI.Xaml.Controls"/>
</Application.Resources>
```

1.5.2 WinUI

- Add a WinUI control to MainPage

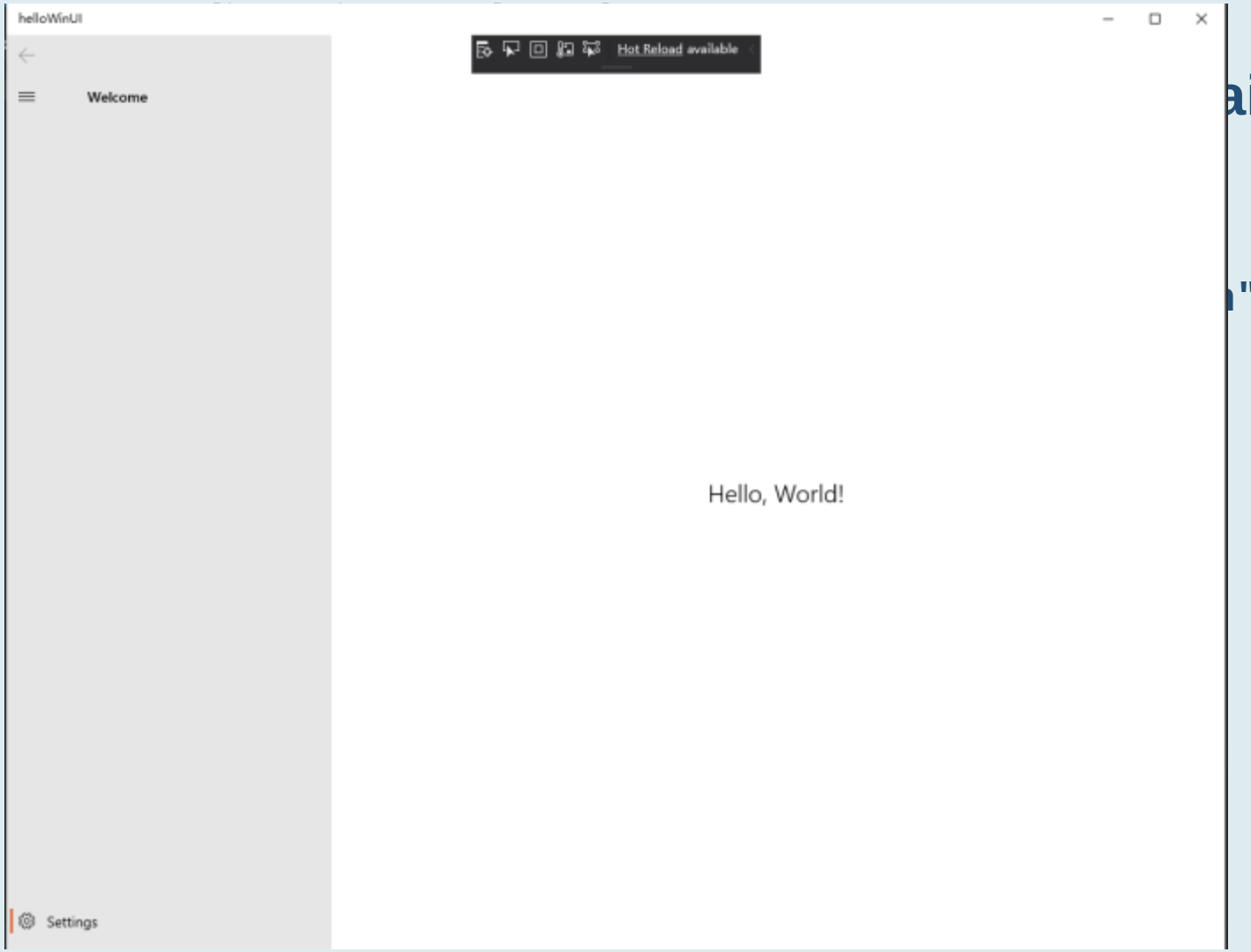
Open MainPage.xaml. In the existing opening Application tag there are some xml namespace declarations. Add the xml namespace declaration

xmlns:muxc="using:Microsoft.UI.Xaml.Controls".

Then, paste the following markup between the existing opening and closing Page tags, overwriting the existing StackPanel element.

```
<muxc:NavigationView PaneTitle="Welcome">
    <TextBlock Text="Hello, WinUI!" VerticalAlignment="Center"
HorizontalAlignment="Center" Style="{StaticResource TitleTextBlockStyle}"/>
</muxc:NavigationView>
```

1.5.2 WinUI



ainPage.h

"

Next version of 1.6: VS installer, Node.js or PTVS

VS installer

https://blog.csdn.net/qq_35970739/article/details/80690037

Tutorial: Create a Node.js and React app in Visual Studio

https://docs.microsoft.com/en-us/visualstudio/javascript/tutorial-nodejs-with-react-and-jsx?view=vs-2020

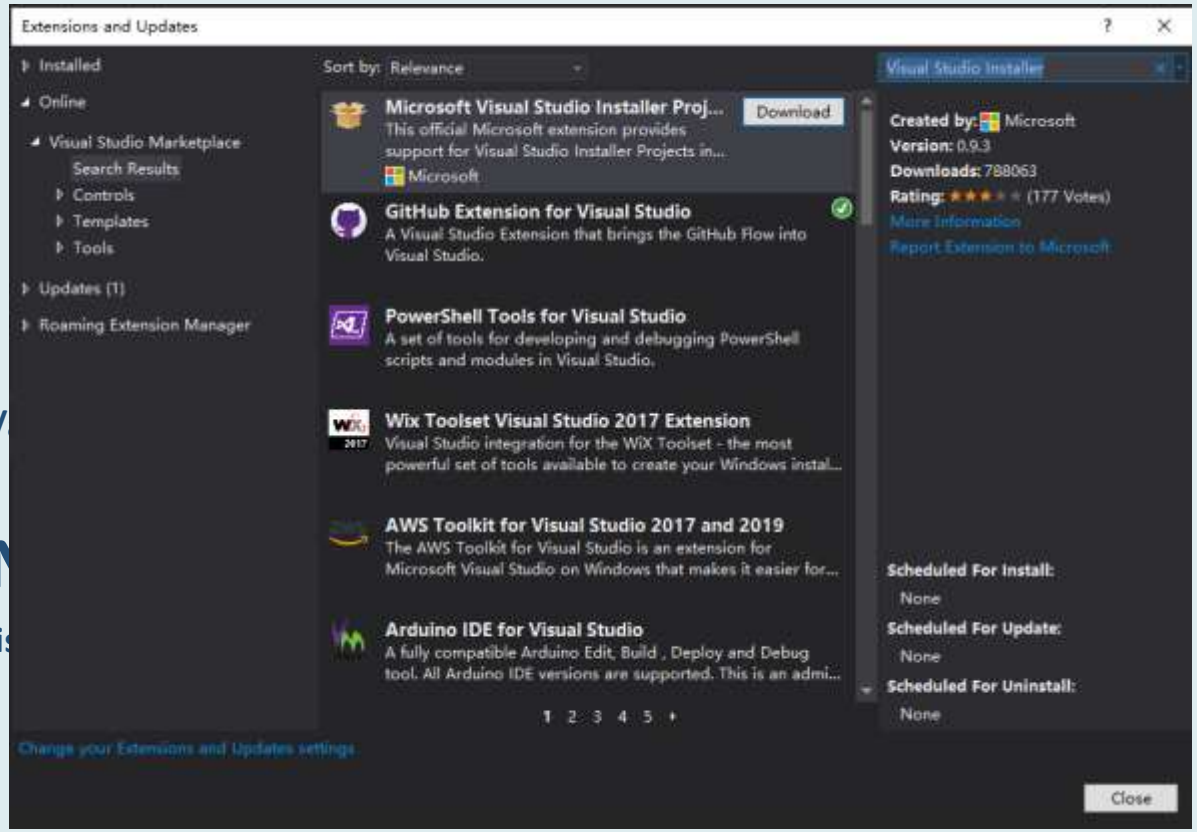
Next version of 1.6: VS installer, Node.js or PTVS

VS installer

https://blog.csdn.net/qq_35970739/

Tutorial: Create a N

https://docs.microsoft.com/en-us/vi



本次课总结

- 熟悉Visual Studio开发环境
  - 简单的MFC程序
  - WPF程序设计
  - UWP程序设计
  - WinUI 与 C++/WinRT
- 教学资料及示例
  - https://gitee.com/wuhanuniversity
- 一些背景知识及技术发展趋势