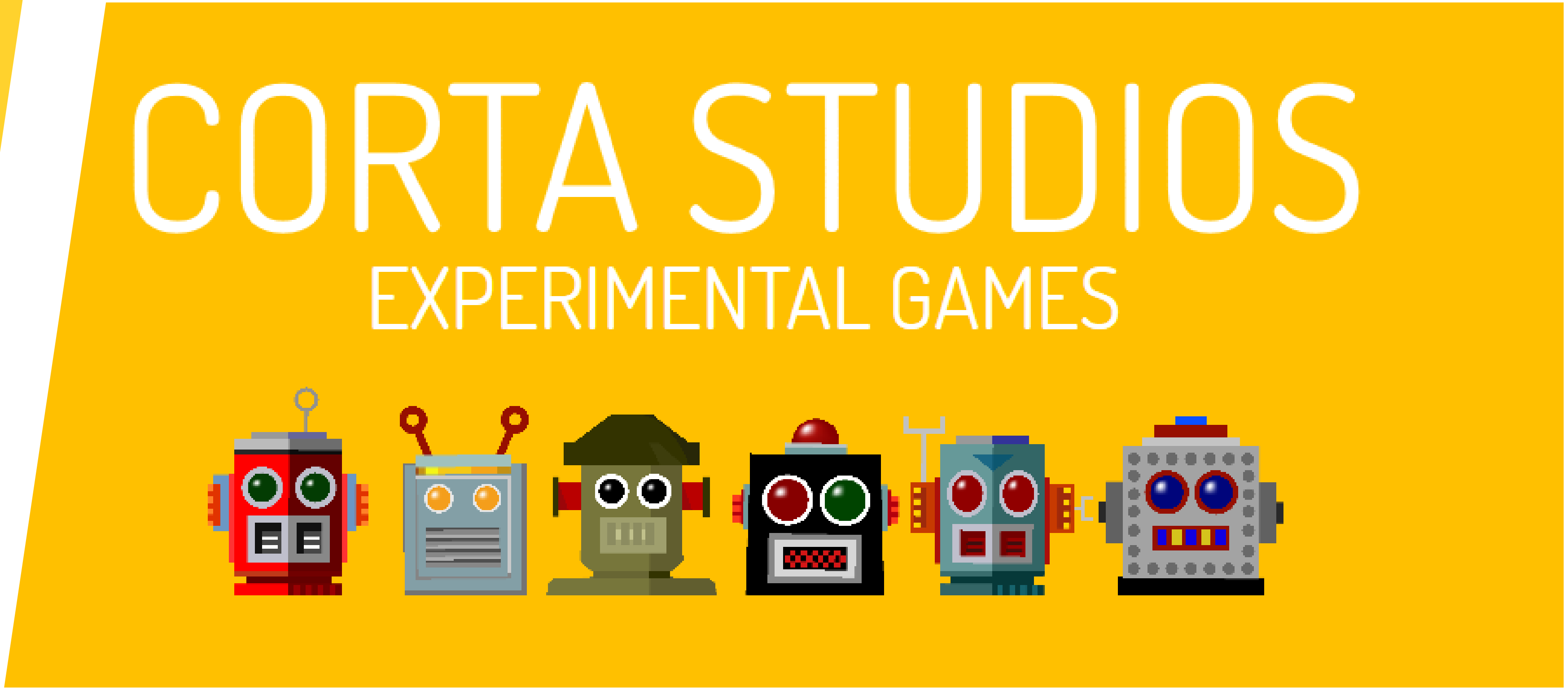
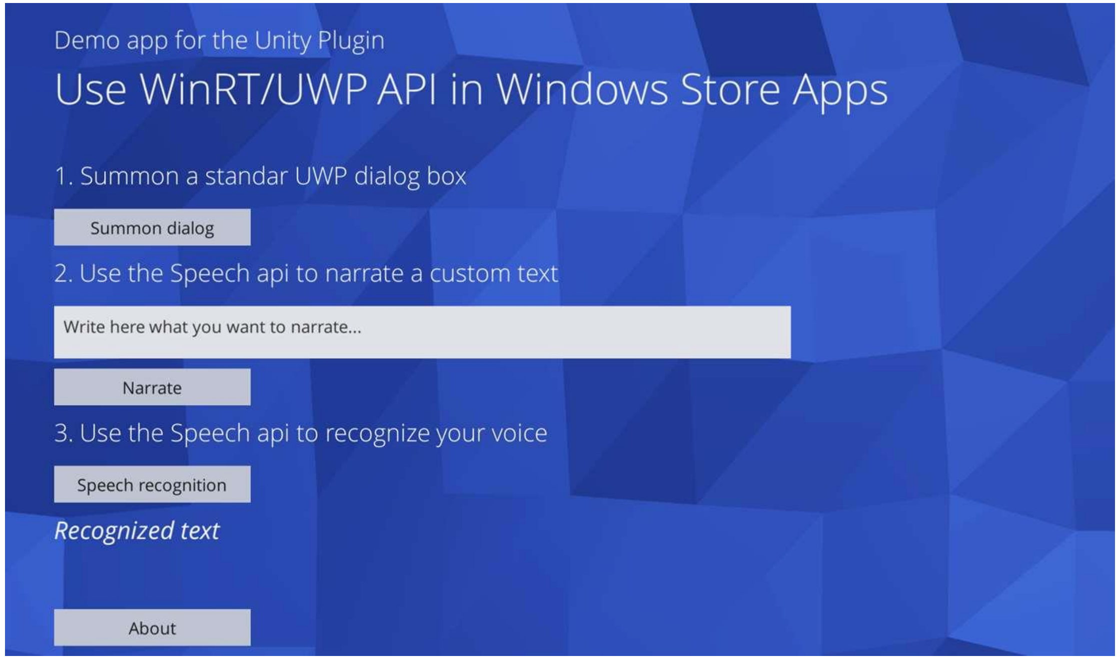


Use WinRT in Unity

Building the demo app

The Demo App



Demo app for the Unity Plugin

Use WinRT/UWP API in Windows Store Apps

1. Summon a standar UWP dialog box

2. Use the Speech api to narrate a custom text

3. Use the Speech api to recognize your voice

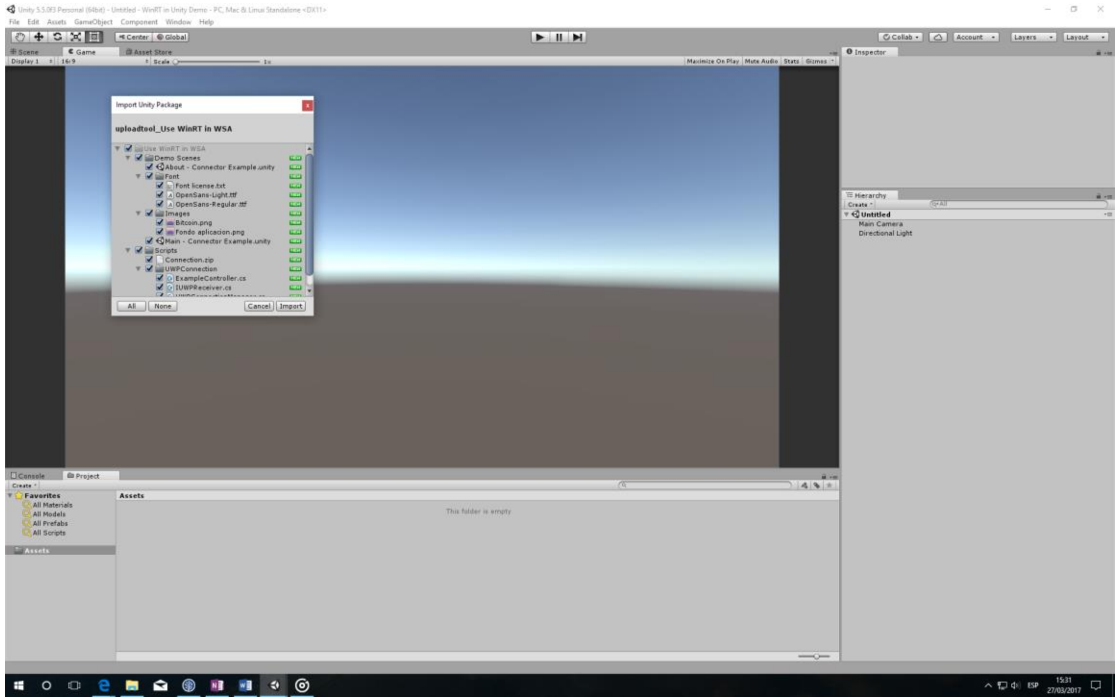
Recognized text

- This is a complementary documentation for the Unity Asset *Use WinRT APIs in Unity*, which you can find in the Unity Store.
- To demonstrate this asset, I have put together a demo app that you can download from the Microsoft Store here: https://www.microsoft.com/store/apps/9mxzlr596z4p
- The scenes to build this app are included in the asset, so you can learn to set up the asset while building the app yourself.

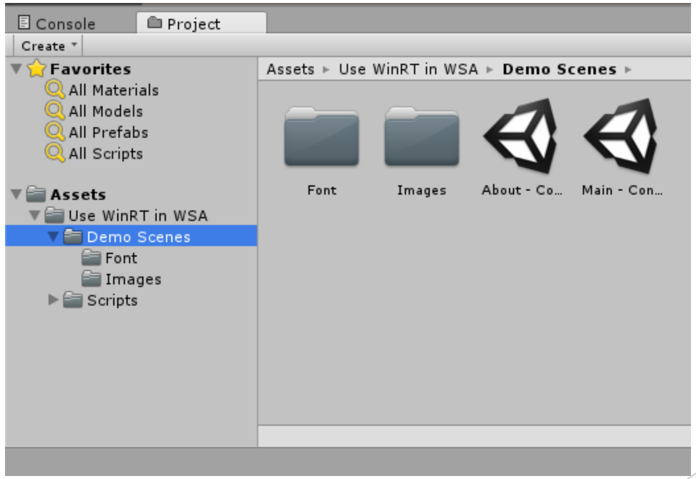
1. Build the WSA app

We are going to import the asset to Unity and use the demo scenes to build a Universal Windows Platform (UWP) project

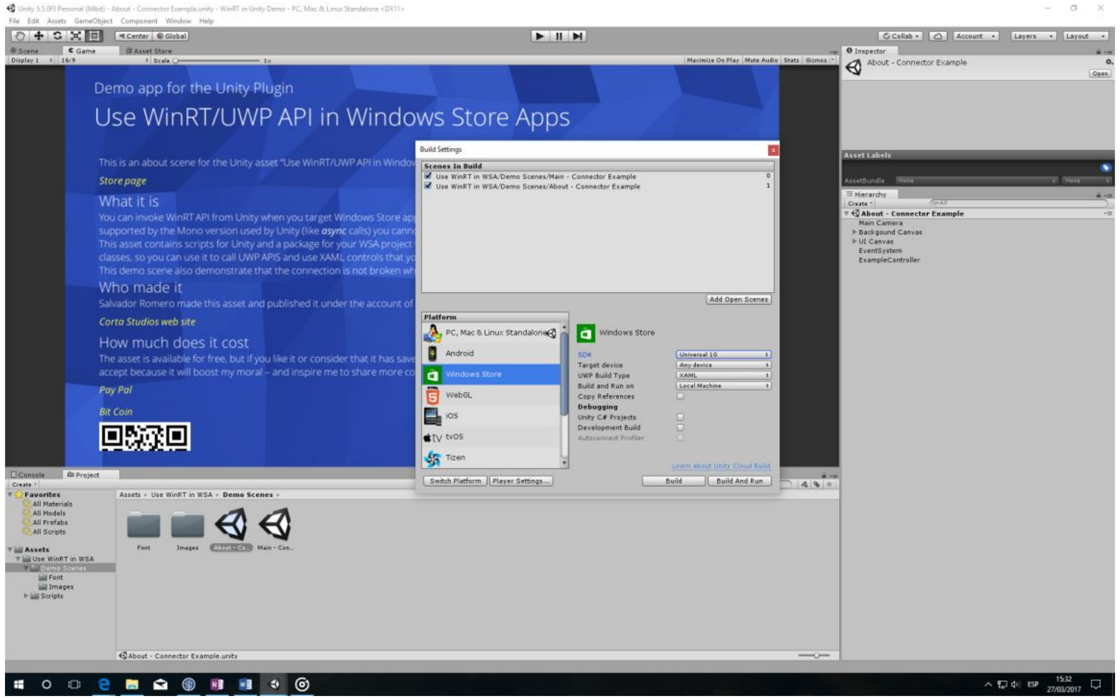
Import the package from the Asset Store



- You will find a *Main Scene* and an *About Scene* in the folder *Demo Scenes*.



Buil the WSA project



Build Settings

Scenes In Build

- ☑ Use WinRT in WSA/Demo Scenes/Main - Connector Example 0
- ☑ Use WinRT in WSA/Demo Scenes/About - Connector Example 1

Platform: Windows Store

SDK	Universal 10
Target device	Any device
UWP Build Type	XAML
Build and Run on	Local Machine
Copy References	□
Debugging	
Unity C# Projects	□
Development Build	□
Autoconnect Profiler	□

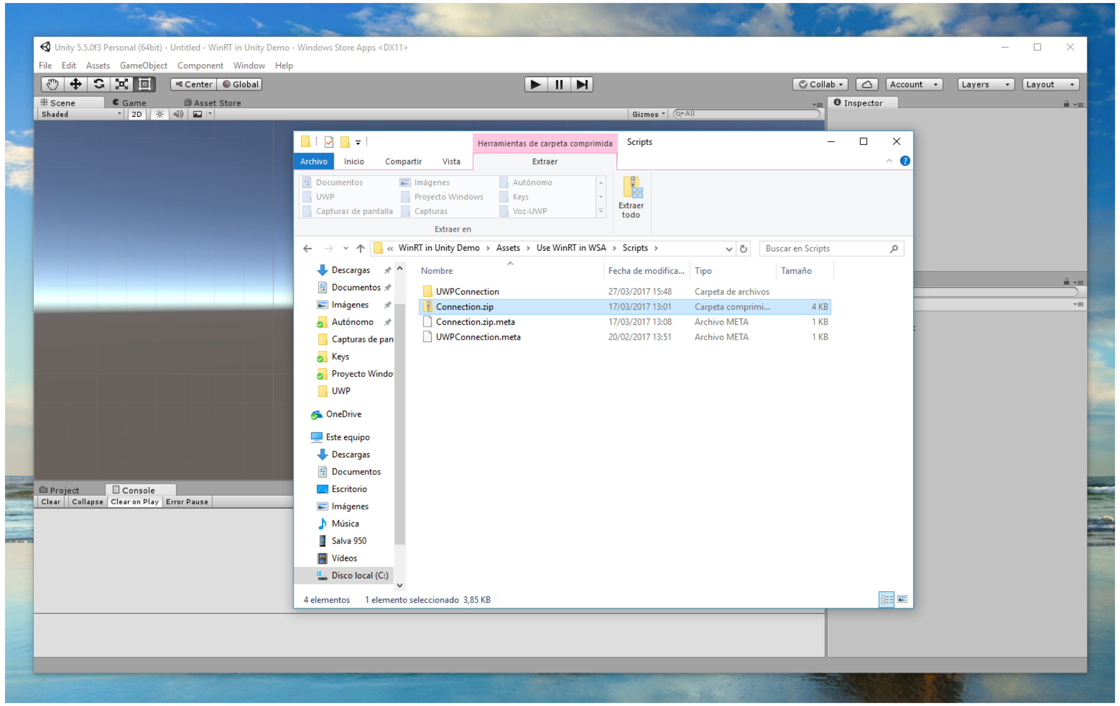
Unity 5.5.0f3 Personal (64bit) - About - Connector Example.unity - WinRT in Unity Demo - PC, Mac & Linux Standalone <DX11>
Demo app for the Unity Plugin
Use WinRT/UWP API in Windows Store Apps
Store page
What it is
Who made it
Corta Studios web site
How much does it cost
Pay Pal
Bit Coin
Add Open Scenes
PC, Mac & Linux Standalone
Android
Windows Store
WebGL
iOS
tvOS
Tizen
Learn about Unity Cloud Build
Switch Platform
Player Settings...
Build
Build And Run
Inspector
About - Connector Example
Hierarchy
Main Camera
Background Canvas
UI Canvas
EventSystem
ExampleController
Project
Assets › Use WinRT in WSA › Demo Scenes
Font
Images
About - Connector Example.unity

- Add both scenes to "Scenes in Build" in the build settings dialog.
- Set Windows Store as the target platform.
- Because this app uses some Windows 10 features (the speech API), select Universal 10 in the sdk.
- Click on build and select a folder for the UWP Project.

2. Configure the UWP project

You need to add some extra .cs files to the UWP, as well as modify the first page file

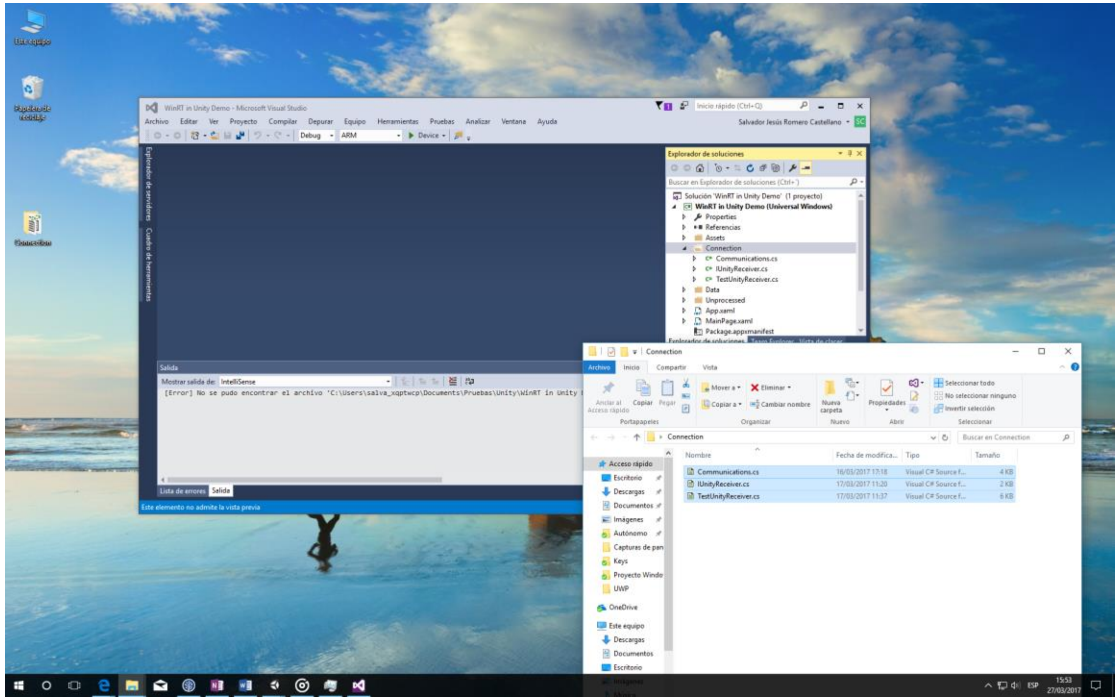
Unzip the .cs files for the UWP


Unity 5.5.0f3 Personal (64bit) - Untitled - WinRT in Unity Demo - Windows Store Apps <DX11>
File Edit Assets GameObject Component Window Help
Herramientas de carpeta comprimida
Scripts
Archivo Inicio Compartir Vista
Extraer
Documentos
UWP
Capturas de pantalla
Imágenes
Proyecto Windows
Capturas
Autónomo
Keys
Voz-UWP
Extraer en
Extraer todo
WinRT in Unity Demo › Assets › Use WinRT in WSA › Scripts ›
Buscar en Scripts
4 elementos 1 elemento seleccionado 3,85 KB

Nombre	Fecha de modifica...	Tipo	Tamaño
UWPConnection	27/03/2017 15:48	Carpeta de archivos	
Connection.zip	17/03/2017 13:01	Carpeta comprimi...	4 KB
Connection.zip.meta	17/03/2017 13:08	Archivo META	1 KB
UWPConnection.meta	20/02/2017 13:51	Archivo META	1 KB

- In Unity, under the *Scripts* folder in the asset, you will find a zip file with some extra classes for the UWP project.
- Unzip those files **outside** the Unity project (we don't want Unity to compile those files)

Import those files with Visual Studio



```
Solución 'WinRT in Unity Demo' (1 proyecto)
  WinRT in Unity Demo (Universal Windows)
    Properties
    Referencias
    Assets
    Connection
      Communications.cs
      IUnityReceiver.cs
      TestUnityReceiver.cs
    Data
    Unprocessed
    App.xaml
    MainPage.xaml
    Package.appxmanifest
```

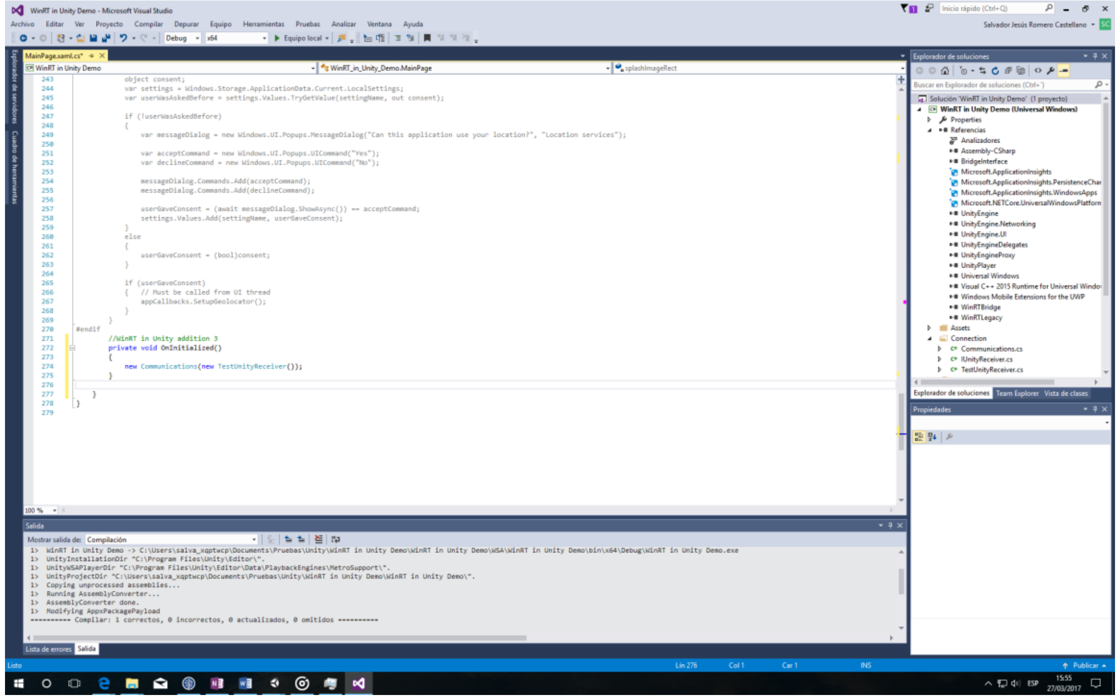
Mostrar salida de: IntelliSense

```
[Error] No se pudo encontrar el archivo 'C:\Users\salva_xqptwcp\Documents\Pruebas\Unity\WinRT in Unity
```

Nombre	Fecha de modifica...	Tipo	Tamaño
Communications.cs	16/03/2017 17:18	Visual C# Source f...	4 KB
IUnityReceiver.cs	17/03/2017 11:20	Visual C# Source f...	2 KB
TestUnityReceiver.cs	17/03/2017 11:37	Visual C# Source f...	6 KB

- Use Visual Studio to open the WSA solution that we have created in Unity.
- Create a folder called Connection in the project (not necessary, but aesthetically convenient).
- Import the unzipped *.cs* files in this folder. You can drag and drop or use the context menu.

Modify the file *MainPage.xaml.cs*

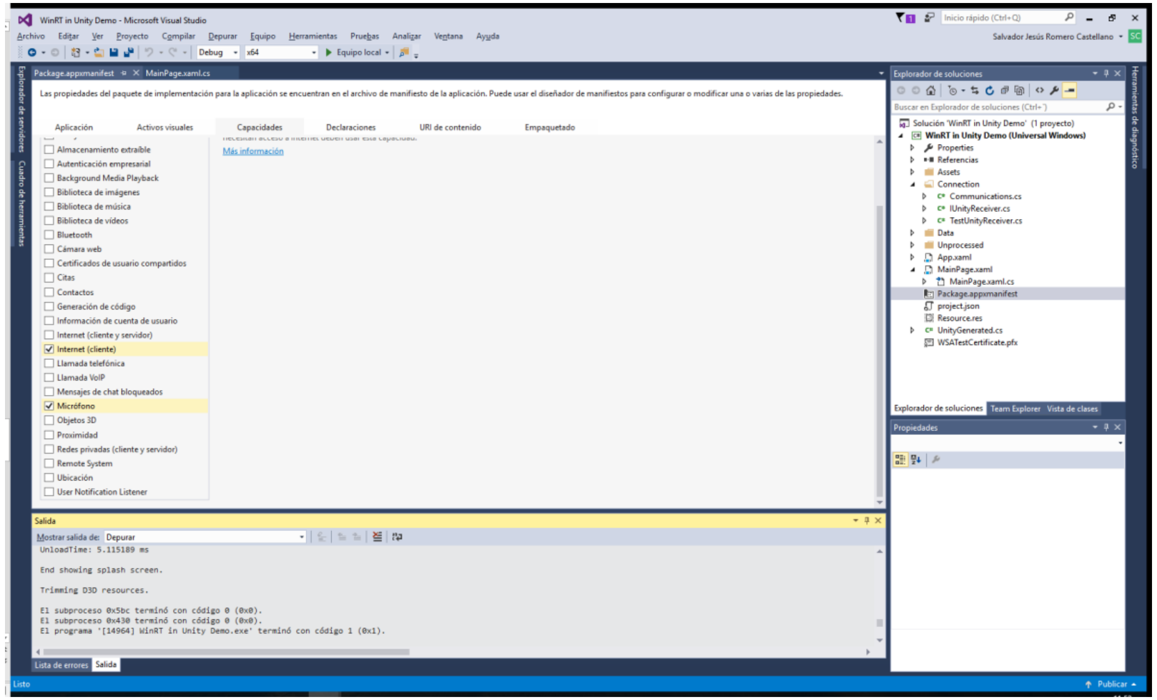

WinRT in Unity Demo - Microsoft Visual Studio
Archivo Editar Ver Proyecto Compilar Depurar Equipo Herramientas Pruebas Analizar Ventana Ayuda
Debug
x64
Equipo local
Salvador Jesús Romero Castellano
MainPage.xaml.cs*
WinRT in Unity Demo
WinRT_in_Unity_Demo.MainPage
splashImageRect
Explorador de soluciones
Solución 'WinRT in Unity Demo' (1 proyecto)
WinRT in Unity Demo (Universal Windows)
Properties
Referencias
Assets
Connection
Communications.cs
IUnityReceiver.cs
TestUnityReceiver.cs
Propiedades
Salida
Mostrar salida de: Compilación
Lista de errores
Salida
Listo
Lín 276
Col 1
Car 1
INS
Publicar

```
            object consent;
            var settings = Windows.Storage.ApplicationData.Current.LocalSettings;
            var userWasAskedBefore = settings.Values.TryGetValue(settingName, out consent);

            if (!userWasAskedBefore)
            {
                var messageDialog = new Windows.UI.Popups.MessageDialog("Can this application use your location?", "Location services");

                var acceptCommand = new Windows.UI.Popups.UICommand("Yes");
                var declineCommand = new Windows.UI.Popups.UICommand("No");

                messageDialog.Commands.Add(acceptCommand);
                messageDialog.Commands.Add(declineCommand);

                userGaveConsent = (await messageDialog.ShowAsync()) == acceptCommand;
                settings.Values.Add(settingName, userGaveConsent);
            }
            else
            {
                userGaveConsent = (bool)consent;
            }

            if (userGaveConsent)
            {   // Must be called from UI thread
                appCallbacks.SetupGeolocator();
            }
        }
#endif
        //WinRT in Unity addition 3
        private void OnInitialized()
        {
            new Communications(new TestUnityReceiver());
        }

    }
}
```

```
1>  WinRT in Unity Demo -> C:\Users\salva_xqptwcp\Documents\Pruebas\Unity\WinRT in Unity Demo\WinRT in Unity Demo\WSA\WinRT in Unity Demo\bin\x64\Debug\WinRT in Unity Demo.exe
1>  UnityInstallationDir "C:\Program Files\Unity\Editor\".
1>  UnityWSAPlayerDir "C:\Program Files\Unity\Editor\Data\PlaybackEngines\MetroSupport\".
1>  UnityProjectDir "C:\Users\salva_xqptwcp\Documents\Pruebas\Unity\WinRT in Unity Demo\WinRT in Unity Demo\".
1>  Copying unprocessed assemblies...
1>  Running AssemblyConverter...
1>  AssemblyConverter done.
1>  Modifying AppxPackagePayload
========== Compilar: 1 correctos, 0 incorrectos, 0 actualizados, 0 omitidos ==========
```

- You need to add some lines to the file *MainPage.xaml.cs*
- It’s very easy. Four lines of code in three different places. Please read the documentation to find out the exact lines and places.
- ***Tip:*** Don’t worry if the file is crowded with errors. They will disappear after you compile the project.

Modify app capabilities in Package.appxmanifest



Las propiedades del paquete de implementación para la aplicación se encuentran en el archivo de manifiesto de la aplicación. Puede usar el diseñador de manifiestos para configurar o modificar una o varias de las propiedades.

- □ Almacenamiento extraíble
- □ Autenticación empresarial
- □ Background Media Playback
- □ Biblioteca de imágenes
- □ Biblioteca de música
- □ Biblioteca de vídeos
- □ Bluetooth
- □ Cámara web
- □ Certificados de usuario compartidos
- □ Citas
- □ Contactos
- □ Generación de código
- □ Información de cuenta de usuario
- □ Internet (cliente y servidor)
- ☑ Internet (cliente)
- □ Llamada telefónica
- □ Llamada VoIP
- □ Mensajes de chat bloqueados
- ☑ Micrófono
- □ Objetos 3D
- □ Proximidad
- □ Redes privadas (cliente y servidor)
- □ Remote System
- □ Ubicación
- □ User Notification Listener

necesitan acceso a internet deben usar esta capacidad.

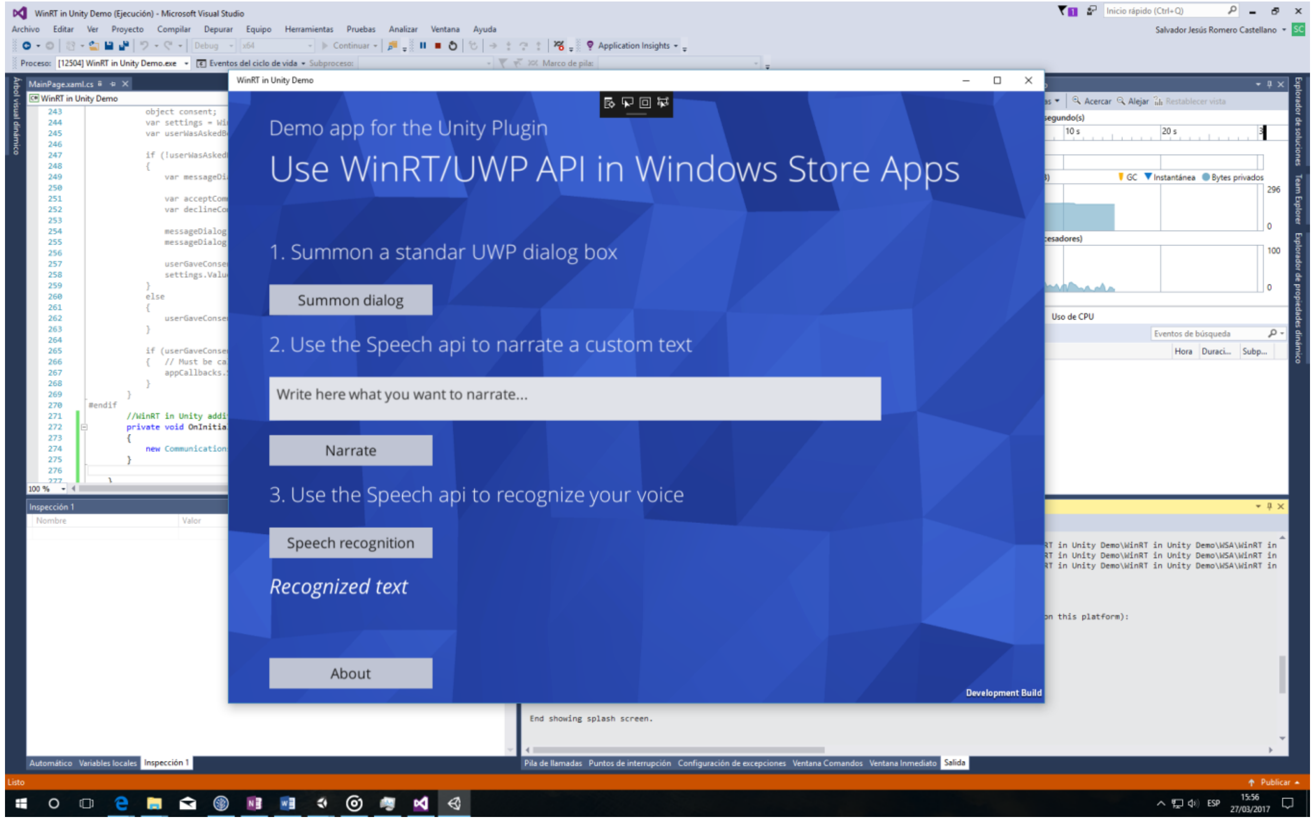
Más información

Mostrar salida de: Depurar

```
UnloadTime: 5.115189 ms

End showing splash screen.

Trimming D3D resources.

El subproceso 0x5bc terminó con código 0 (0x0).
El subproceso 0x430 terminó con código 0 (0x0).
El programa '[14964] WinRT in Unity Demo.exe' terminó con código 1 (0x1).
```

- This step is optional
- Open Package.appxmanifest and go to the capabilities tab.
- Make sure that internet (client) and Microphone are checked.
- Both capabilities are needed for the speech API.

Compile and run



Proceso: [12504] WinRT in Unity Demo.exe · Eventos del ciclo de vida · Subproceso: · Marco de pila:

```
            object consent;
            var settings = Wi
            var userWasAskedB

            if (!userWasAsked
            {
                var messageDi

                var acceptCom
                var declineCo

                messageDialog
                messageDialog

                userGaveConse
                settings.Valu
            }
            else
            {
                userGaveConse
            }

            if (userGaveConse
            {   // Must be ca
                appCallbacks.
            }
        }
#endif
        //WinRT in Unity addi
        private void OnInitia
        {
            new Communication
        }
```

WinRT in Unity Demo

Demo app for the Unity Plugin

Use WinRT/UWP API in Windows Store Apps

1. Summon a standar UWP dialog box

Summon dialog

2. Use the Speech api to narrate a custom text

Write here what you want to narrate...

Narrate

3. Use the Speech api to recognize your voice

Speech recognition

Recognized text

About

Development Build

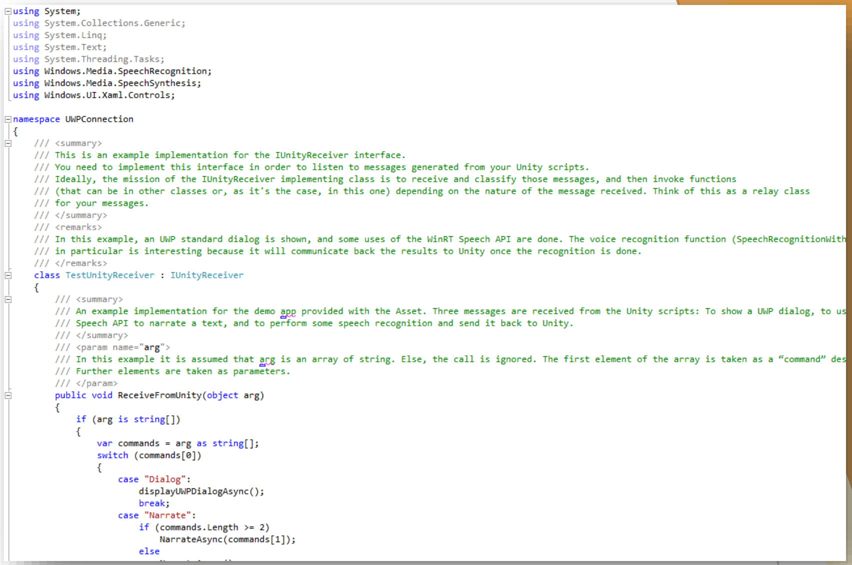
```
RT in Unity Demo\WinRT in Unity Demo\WSA\WinRT in
RT in Unity Demo\WinRT in Unity Demo\WSA\WinRT in
RT in Unity Demo\WinRT in Unity Demo\WSA\WinRT in

on this platform):

End showing splash screen.
```

About this asset

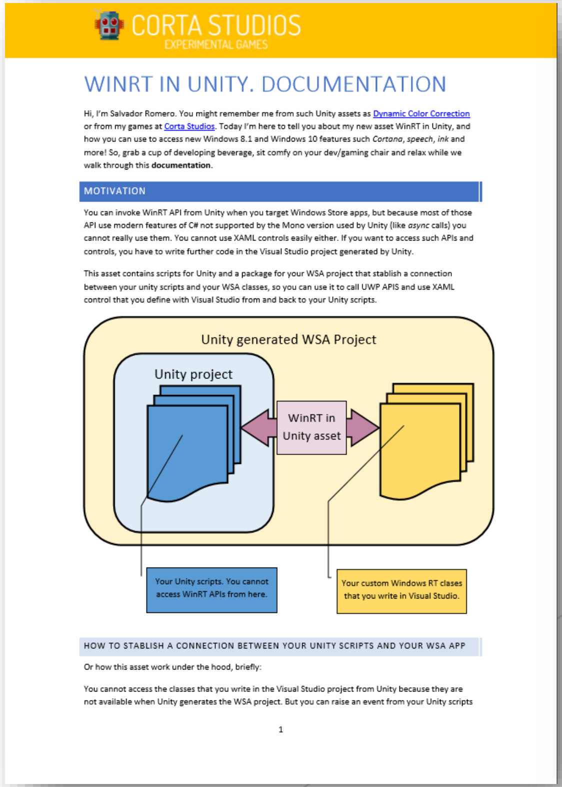
More misc. info here!

Donload links and documentation

- Download the demo app from the Microsoft Store: https://www.microsoft.com/store/apps/9mxzlr596z4p
- Download the asset from the Unity Asset Store: http://u3d.as/LnD
- Documentation: This Asset documentation comprises:
  - This presentation
  - A pdf document with detailed instruction on how to configure and use the asset.
  - All the scripts and .cs files have inline documentation



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.Media.SpeechRecognition;
using Windows.Media.SpeechSynthesis;
using Windows.UI.Xaml.Controls;

namespace UWPConnection
{
    /// <summary>
    /// This is an example implementation for the IUnityReceiver interface.
    /// You need to implement this interface in order to listen to messages generated from your Unity scripts.
    /// Ideally, the mission of the IUnityReceiver implementing class is to receive and classify those messages, and then invoke functions
    /// (that can be in other classes or, as it's the case, in this one) depending on the nature of the message received. Think of this as a relay class
    /// for your messages.
    /// </summary>
    /// <remarks>
    /// In this example, an UWP standard dialog is shown, and some uses of the WinRT Speech API are done. The voice recognition function (SpeechRecognitionWith
    /// in particular is interesting because it will communicate back the results to Unity once the recognition is done.
    /// </remarks>
    class TestUnityReceiver : IUnityReceiver
    {
        /// <summary>
        /// An example implementation for the demo app provided with the Asset. Three messages are received from the Unity scripts: To show a UWP dialog, to us
        /// Speech API to narrate a text, and to perform some speech recognition and send it back to Unity.
        /// </summary>
        /// <param name="arg">
        /// In this example it is assumed that arg is an array of string. Else, the call is ignored. The first element of the array is taken as a “command” des
        /// Further elements are taken as parameters.
        /// </param>
        public void ReceiveFromUnity(object arg)
        {
            if (arg is string[])
            {
                var commands = arg as string[];
                switch (commands[0])
                {
                    case "Dialog":
                        displayUWPDialogAsync();
                        break;
                    case "Narrate":
                        if (commands.Length >= 2)
                            NarrateAsync(commands[1]);
                        else
```



CORTA STUDIOS
EXPERIMENTAL GAMES

WINRT IN UNITY. DOCUMENTATION

Hi, I'm Salvador Romero. You might remember me from such Unity assets as Dynamic Color Correction or from my games at Corta Studios. Today I'm here to tell you about my new asset WinRT in Unity, and how you can use to access new Windows 8.1 and Windows 10 features such *Cortana*, *speech*, *ink* and more! So, grab a cup of developing beverage, sit comfy on your dev/gaming chair and relax while we walk through this **documentation**.

MOTIVATION

You can invoke WinRT API from Unity when you target Windows Store apps, but because most of those API use modern features of C# not supported by the Mono version used by Unity (like *async* calls) you cannot really use them. You cannot use XAML controls easily either. If you want to access such APIs and controls, you have to write further code in the Visual Studio project generated by Unity.

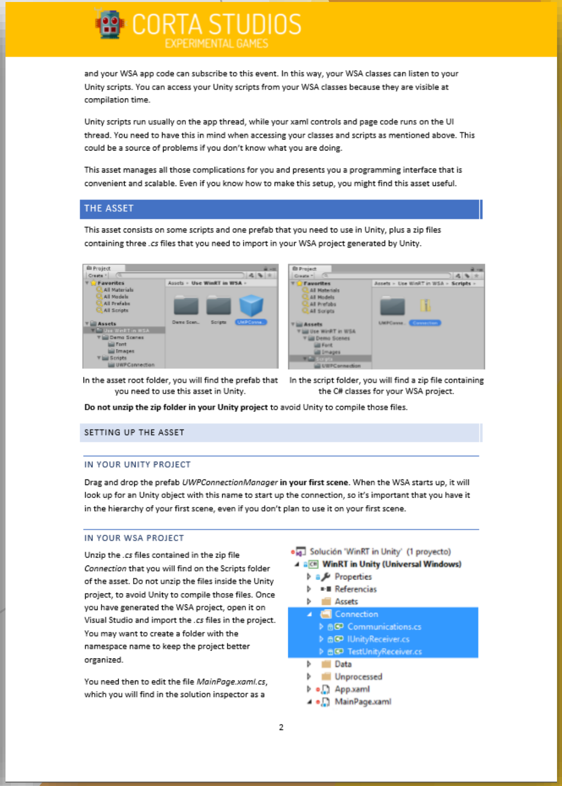
This asset contains scripts for Unity and a package for your WSA project that stablish a connection between your unity scripts and your WSA classes, so you can use it to call UWP APIS and use XAML control that you define with Visual Studio from and back to your Unity scripts.

Unity generated WSA Project
Unity project
WinRT in Unity asset
Your Unity scripts. You cannot access WinRT APIs from here.
Your custom Windows RT clases that you write in Visual Studio.

HOW TO STABLISH A CONNECTION BETWEEN YOUR UNITY SCRIPTS AND YOUR WSA APP

Or how this asset work under the hood, briefly:

You cannot access the classes that you write in the Visual Studio project from Unity because they are not available when Unity generates the WSA project. But you can raise an event from your Unity scripts



CORTA STUDIOS
EXPERIMENTAL GAMES

and your WSA app code can subscribe to this event. In this way, your WSA classes can listen to your Unity scripts. You can access your Unity scripts from your WSA classes because they are visible at compilation time.

Unity scripts run usually on the app thread, while your xaml controls and page code runs on the UI thread. You need to have this in mind when accessing your classes and scripts as mentioned above. This could be a source of problems if you don't know what you are doing.

This asset manages all those complications for you and presents you a programming interface that is convenient and scalable. Even if you know how to make this setup, you might find this asset useful.

THE ASSET

This asset consists on some scripts and one prefab that you need to use in Unity, plus a zip files containing three *.cs* files that you need to import in your WSA project generated by Unity.

In the asset root folder, you will find the prefab that you need to use this asset in Unity.

In the script folder, you will find a zip file containing the C# classes for your WSA project.

Do not unzip the zip folder in your Unity project to avoid Unity to compile those files.

SETTING UP THE ASSET

IN YOUR UNITY PROJECT

Drag and drop the prefab *UWPConnectionManager* **in your first scene**. When the WSA starts up, it will look up for an Unity object with this name to start up the connection, so it's important that you have it in the hierarchy of your first scene, even if you don't plan to use it on your first scene.

IN YOUR WSA PROJECT

Unzip the *.cs* files contained in the zip file *Connection* that you will find on the Scripts folder of the asset. Do not unzip the files inside the Unity project, to avoid Unity to compile those files. Once you have generated the WSA project, open it on Visual Studio and import the *.cs* files in the project. You may want to create a folder with the namespace name to keep the project better organized.

You need then to edit the file *MainPage.xaml.cs*, which you will find in the solution inspector as a

About

- Salvador Romero made this asset and published it under the account of Corta Studios. You can learn more about Corta Studios here: http://cortastudios.com
- If you run into any trouble or want to get in contact with me, please write to salvador@cortastudios.com
- The asset is available for free, but if you like it, or consider that it has saved you some valuable time, you can help me back by:
  - Rating the asset in the Asset Store, or write a review
  - Invite me to a coffee, which I will gladly accept because it will boost my moral – and inspire me to share more code. Thank you!! ☺
    - Paypal (https://www.paypal.me/salvadorjesus)
    - Bitcoin

1NCra5L4yDZHarLarAuWfF5Fs9v6x4Acp1