УДК 00.004.4

Ю.Э. Парфенов, В.Н. Федорченко

Харьковский национальный экономический университет, Харьков

ПОДДЕРЖКА C++ ПРИ РАЗРАБОТКЕ ПРИЛОЖЕНИЙ ДЛЯ WINDOWS PHONE 8

Рассматриваются аспекты использования языка программирования C++ под Windows Phone 8, а также некоторые общие соображения, которыми должен руководствоваться любой C++ разработчик при выборе целевой платформы для разработки приложений. Представлены основные характеристики указанной платформы, преимущества ее использования разработчиками. Обсуждаются возможности отладки приложений под Windows Phone 8, реализации 3D-анимации и графики.

Ключевые слова: *графический пользовательский интерфейс, платформа, интерфейс, класс, Direct3D, DirectX, управляемый код, API, C++/CX, SDK, машинный код, элемент управления, XAML, потоки, события, метаданные, рендеринг, коллекции, COM-объекты, шаблоны, тип данных*

Постановка проблемы

Windows Phone 8 SDK добавляет два новых набора интерфейсов API для разработки приложений, использующих машинный код. Эта статья посвящена использованию C++ под Windows Phone, а также некоторым общим соображениям, которыми должен руководствоваться любой C++ разработчик при выборе целевой платформы для разработки.

Windows Phone 8 SDK API состоит из трех взаимосвязанных частей (Рис.1) [1, 2, 3]:

• . NET - предоставляет доступ к функциональности Windows Phone, например, Live Tiles, отправка SMS;

• Windows Phone Runtime (WinPRT) - промежуточный API, для доступа к низкоуровневой функциональности, такой как голосовые команды, VoIP;

• Машинный код - предоставляет доступ к низкоуровневым API, таким как Socket, DirectX.

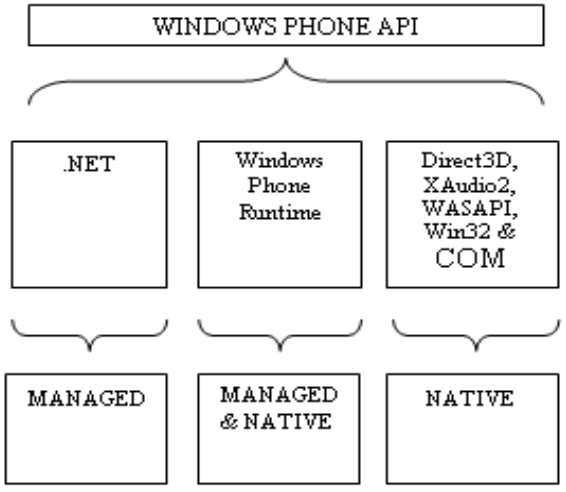

Рис. 1. Windows Phone 8 SDK API

.NET (C# и VB)

. NET API – это оригинальный фреймворк для Windows Phone 8. Он позволяет разработать графический интерфейс пользователя с использованием XAML-технологий, получать доступ к основной функциональности (Live Tiles, задачи, отправка почты, SMS) и XNA. Это самый важный фреймворк для разработки приложений Windows Phone. Для разработки можно использовать языки C# или VB.

Примечание: технология XNA всегда поддерживается, но только для разработки приложений Windows Phone 7. В Windows Phone 8 она заменена на Direct3D.

WinPRT (C#, VB & C++/CX)

Этот API имеет следующую функциональность:

• общий API с Windows 8. Подмножество этого API используется совместно с Windows 8;

• Смешивание управляемого и машинного кода. Управляемый и машинный коды несовместимы по своей природе, но этот API можно использоваться как с одним, так и с другим.

API WinPRT базируется на эволюции технологии COM. Чтобы использовать его с C++ и легко разрабатывать компоненты WinPRT, Microsoft создала Visual C++ Language Reference (C + +/CX) [2, 5].

Это расширение добавляет к C++ возможность использования управляемых элементов, таких как свойства, делегаты, события. Для обеспечения возможности использования в управляемом коде, класс, разработанный на C++/CX, генерирует метаданные. Фактически, набор инструментов переводит этот класс в комплексный код C++ с аннотациями WinPRT:

с C++/CX

public ref class Number sealed

```
{
public:
    Number() : _value(0) { }

    int GetValue()        { return _value;  }
    void SetValue(int value) { _value = value; }
private:
    int _value;
};
```

без C++/CX

```
[exclusiveto(Number)]
[uuid(5b197688-2f57-4d01-92cd-a888f10dcd90)]
[version(1.0)]
interface INumber : IInspectable
{
    HRESULT GetValue([out, retval] INT32* value);
    HRESULT SetValue([in] INT32 value);
}
[activatable(1.0), version(1.0)]
runtimeclass Number
{
    [default] interface INumber;
}
class Number : public RuntimeClass<INumber>
{
    InspectableClass (RuntimeClass_WRLNumberComponent_Number, BaseTrust)
public:
    Number() : _value(0) { }
    virtual HRESULT STDMETHODCALLTYPE GetValue(INT32* value) override {
        *value = _value;
        return S_OK;
    }
    virtual HRESULT STDMETHODCALLTYPE SetValue(INT32 value) override {
        _value = value;
        return S_OK;
    }
private:
    INT32 _value;
};
```

При разработке на C++ вы всегда будете где-то использовать это расширение, потому что это ваш C++ код используется управляемым API, а не наоборот. Например, нельзя использовать объект C++ непосредственно в приложении на C#. Сначала нужно написать компонент Windows Runtime.

Native («Родной» C++)

В Visual Studio 2012 есть очень хорошая поддержка языка C++ и его последней нормализации. Для поиска функциональности, поддерживаемой Windows Phone, полезны следующие ссылки [1, 4, 5]:

• Языки C/C++;

• Справочник по стандартной библиотеки C++;

• Справочник по языку C++.

C++11 добавляет много действительно хороших концепций и возможностей [5].

«Родной» API разработан на C++ и поделен на библиотеки:

• Win 32 и COM API - подмножество Win32 API. Эта библиотека предоставляет доступ к фнкциям для работы с файлами, сокетами и COM-объектами.

• Direct3D 11 - 3D-рендеринг.

• Microsoft Media Foundation API - фреймворк для захвата аудио/видео и рендеринга. Windows Phone реализует подмножество Windows 8.

• Windows Runtime C++ Template Library - вспомогательная библиотека для использования и создания компонентов COM или Windows Runtime на C++. Заменяет старый ATL API. С Windows Phone, эта библиотека, как правило, применяется только для использования компонентов COM и WinPRT.

• Библиотека параллельных шаблонов - высокоуровневая многопоточная библиотека алгоритмов с параллелизмом.

В некоторых случаях можно использовать непосредственно COM-объект:

• Расширенный доступ к камере;

• Расширенный доступ к аудио.

Native application/«Родные» приложения

Windows Phone 8.0 поддерживает две модели приложений [4]:

- DirectX может запускать приложения, которые полностью написаны на C++ для игр или бизнес-логики, графики и ввода данных;

- XAML + C# приложения могут использовать код C++, написанный в DirectX и отображаемый с помощью одного из двух объектов XAML

- XAML + C# приложения могут использовать код C++, скомпилированный в compute-only библиотеки C++ под названием «WinPRT components».

Не поддерживаются приложения, написанные исключительно на C++ и XAML. Считается, что это удовлетворяет потребности 80% сообщества WPDev: полная поддержка родных игр и повторного использования кода для приложений и игр, предназначенных для нескольких платформ.

Чтобы запустить приложение, Windows Phone необходима «точка входа», которую можно использовать. Чтобы выполнить приложение, эта "точка входа" должна быть разработана на C++/CX, а функция main C++ заменена версией C++/CX:

```
[Platform::MTAThread]
int main(Platform::Array<Platform::String^>^)
```

```
{
    auto factory= ref new myFactory();
    CoreApplication::Run
            (direct3DApplicationSource);
    return 0;
}
```

Platform::MTAThread - метаданные о многопоточном окружении приложения.

myFactory () - реализует IFrameworkViewSource. Этот класс является фабрикой, которая используется для создания экземпляра Iframeworkview.

```
ref class myFactory sealed : Windows::
                        ApplicationModel::
            Core::IFrameworkViewSource
{
public:
  virtual Windows::
   ApplicationModel::Core::
            IFrameworkView^ CreateView()
  {
     return ref new myView();
  };
};
```

CoreApplication::Run - функция C++/CX, которая запрашивает Iframeworkview из фабрики IFrameworkViewSource.

IFrameworkView является провайдером для обеспечения рендеринга Direct3D.

Функции интерфейса:

Initialize() - функция инициализации. Принимает CoreApplicationView как параметр. Можно использовать событие CoreApplicationViewActivated для получения уведомления о активации приложения и использовать эту функцию для регистрации своего IFrameworkView с событиями CoreApplication для обработки изменений состояния приложения.

```
void myView::Initialize(CoreApplicationView^
applicationView)
{
    applicationView->Activated +=
            ref new
    TypedEventHandler<CoreApplicationView^,
IActivatedEventArgs^>(this,
&myView::OnActivated);
 //обрабатывать приостановления
//и возобновления состояний приложения.
        CoreApplication::Suspending +=
            ref new
EventHandler<SuspendingEventArgs^>(this,
&myView::OnSuspending);
        CoreApplication::Resuming +=
            ref new
EventHandler<Platform::Object^>(this,
&myView::OnResuming);
}
```

SetWindow() - устанавливает текущий CoreWindow. Используйте его для обработки событий, таких как закрытие, изменение видимости, касание и т.д.:

```
void myView::SetWindow(CoreWindow^
 window)
{
    window->VisibilityChanged +=
    ref new TypedEventHandler<CoreWindow^,
VisibilityChangedEventArgs^>(this,
&myView::OnVisibilityChanged);
    window->Closed +=
    ref new TypedEventHandler<CoreWindow^,
CoreWindowEventArgs^>(this,
&myView::OnWindowClosed);
    window->PointerPressed +=
    ref new TypedEventHandler<CoreWindow^,
PointerEventArgs^>(this,
&myView::OnPointerPressed);
    window->PointerMoved +=
    ref new TypedEventHandler<CoreWindow^,
PointerEventArgs^>(this,
&myView::OnPointerMoved);
    window->PointerReleased +=
    ref new TypedEventHandler<CoreWindow^,
PointerEventArgs^>(this,
&myView::OnPointerReleased);
}
```

Uninitialize() : Инициализация ресурсов.

Run(): Запуск представления. Реализация событий жизненного цикла приложения.

```
void myView::Run()
{
     // отсчет времени
    BasicTimer^ timer = ref new BasicTimer();
    while (!m_windowClosed)
//пока приложение не закрыто
    {
            if (m_windowVisible)// приложение
//видно, обновление Direct3D рендеринга
            {
                    timer->Update();
                    CoreWindow::
GetForCurrentThread()->
Dispatcher->ProcessEvents
(CoreProcessEventsOption::
ProcessAllIfPresent);
// обработки текущих событий системы
                    m_renderer->Update
(timer->Total, timer->Delta);
// обновить ссылки.
                    m_renderer->Render();
// Direct3D рендеринг
                    m_renderer->Present();
// Этот вызов синхронизирует дисплей
  //с частотой кадров.
            }
```

```
            else // приложение не видно
            {
        CoreWindow::GetForCurrentThread()->
    Dispatcher->ProcessEvents
    (CoreProcessEventsOption::
    ProcessOneAndAllPending);
    //ожидание нового события.
            }
        }
    }
```

«Родные» приложения имеют три важных свойства [2, 4]:

• Для отображения данных можно использовать только Direct3D. Родной API не имеет средств для создания GUI;

• Вы должны обрабатывать системные события;

• Не имеют доступа к функциональности .NET. Нельзя использовать живую плитку, отправлять SMS и т.п.

Без функциональности .NET, родные приложения не так совершенны и вы ограничены разработкой Direct3D. Но помните, что вы можете разрабатывать компоненты WinPRT на C++/CX. Так что возможно инкапсулировать код C++ в C++/CX и использовать его с управляемым кодом.

Смешанные приложения

Вы можете разрабатывать смешанные приложения, в которых управляемый код использует компоненты WinPRT. Для разработки на C++, необходимо создать компонент WinPRT, который взаимодействует с частью кода на C++ с помощью общедоступного «запечатанного» класса C++/CX. Важно, прочитать документацию по расширению C++, чтобы понять его особенности [1, 3]:

• класс или структура C++/CX объявляется с ключевым словом ref;

• выделение памяти производится с помощью ref new;

• структуры C++/CX можно использовать как POD;

• на экземпляр класса ссылаются посредством ^ - «умного» указателя с подсчетом ссылок, наподобие std :: shared_ptr:

myclass ^ myClass = ref new myclass ();

• Platform::String Class заменяет std :: wstring. C++/CX испольует строки Unicode;

• коллекции C++/CX совместимы с STL;

• основные типы схожи;

• свойство похоже на getter/setter в C++;

• делегат - функциональный объект. Он может инкапсулировать C++/CX или управляемый код;

• событие является коллекцией делегатов, которые выполняются при возникновении события;

• ключевое слово sealed: «запечатанный» класс или «запечатанная» функция не может быть переопределена.

Для использования в управляемом коде, код C++/CX генерирует набор метаданных. Этот процесс зависит от модификатора доступа (табл. 1).

Таблица 1

Модификаторы доступа

Модификатор	Значение	Добавляется в метаданные?
private	Уровень доступа по умолчанию. Имеет тот же смысл, что и в C++.	Нет
protected	Уровень доступа по умолчанию. Имеет тот же смысл, что и в C++ внутри приложения, компонента или метаданных.	Да
public	Имеет тот же смысл, что и в C++	Да
public protected or protected public	protected - в метаданных, public - внутри приложения или компонента	Да
protected private or private protected	Не видим в метаданных; protected - внутри приложения или компонента	Нет
internal or private public	Не видим в метаданных; public - внутри приложения или компонента	Нет

Метаданные генерируются только для определенных объектов C++/CX. Класс C++/CX может объявить объект C++ только в том случае, если член/ функция имеет уровень доступа private или internal. Пользовательский public-класс/структура должны использовать модификатор sealed, потому что управляемый код не может переопределить его.

```
public ref class MyClass sealed
{
private : // C++ object может использоваться
   std::string aString:
   std::vector<uint32_t> aFunction();
public : // Объект и функции доступны в
//управляемым код. если используется объект
//C++, ошибка.
   property Platform::String ^ anotherString;

Windows::Foundation::Collections::IVector<uint32>
^anotherFunction();
//...
};
```

Уровень доступа public очень строгий, и его могут использоваться только определенные объекты C++/CX: функция, свойство, делегат, событие.

Эти объекты должны соответствовать следующим типам:

• Фундаментальные типы;

• Объекты WinPRT API;

• Ссылка на пространство имен C++/CX с ограничением. Если класс не совместим, то в общем случае он реализует интерфейс, объявленный в WinPRT API;

• Пользовательский public sealed класс или структура.

Как только в управляемом приложения появляется ссылка на ваш компонент, можно использовать его аналогично другому управляемому объекту. Таким образом, можно связать public-свойства с XAML и подключиться к public-событиям.

Предупреждение: Windows Phone реализует только подмножество пространств имен Windows 8 C++/CX. Некоторые объекты не доступны.

Коллекции

WinPRT API не реализует классы коллекций. Для передачи коллекции между управляемым и машинным кодом определен набор интерфейсов коллекций. Эти интерфейсы имеют эквиваленты в управляемом коде. Например IVector становится IList в C#.

Коллекции C++/CX используют элементы C++ и не могут непосредственно использоваться в управляемом коде [5]. Так как эти классы реализуют интерфейс WinPRT, можно использовать преобразование к типу соответствующего интерфейса.

```
Windows::Foundation::Collections::
IVector<int>^ Class1::GetInts()
{
  auto vec = ref new
Platform::Collections::Vector<int>();
   ...
   return vec;
// Неявное приведение к WinPRT интерфейсу.
//Управляемый код может использовать
//коллекции, которые возвращены
}
```

Примечание: коллекции STL могут быть преобразованы в их C++/CX эквивалент. При преобразовании временной коллекции, можно использовать STD :: move, чтобы избежать ненужного внутреннего копирования.

```
Windows::
Foundation::Collections::IVector<int>^
Class1::GetInts()
{
  st::vector<int> vec;
  for(int i = 0; i < 10; i++)
  {
    vec.push_back(i);
  }
  // неявное преобразование в IVector
  return ref new
Platform::Collections::Vector<int>
(std::move(vec));
}
```

Отладчик

Нельзя отлаживать управляемый код и машинный код одновременно. Для выбора отладчика, который вы хотите использовать (Рис. 2):

• Откройте свойства проекта;

• Откройте вкладку отладки и выберите режим отладчика (Рис. 2).

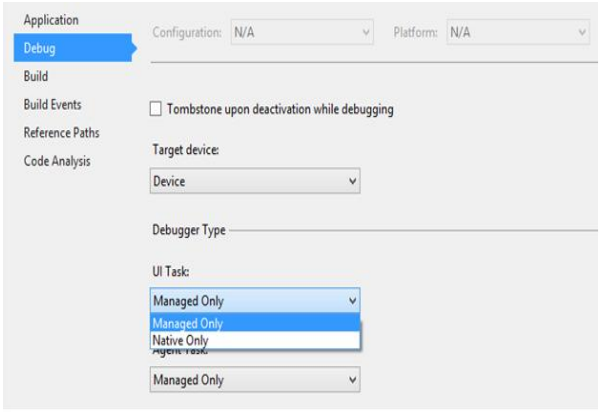

Рис. 2. Режим Debugger

Direct 3D

Управляемый код не имеет непосредственного доступа к Direct 3D. Вы должны разработать компонент WinPRT, использующий Direct3D в коде C++.

Direct3D может отображаться двумя элементами управления XAML [1, 2, 3]:

• DrawingSurface. Direct3D будет воспроизводиться в области DrawingSurface. Этот элемент используется аналогично другие элементам управления пользовательского интерфейса и можно выполнять преобразования, анимацию и т.п.

• DrawingSurfaceBackgroundGrid: Direct3D воспроизводится на фоне вашего приложения. Эта компоновка должна управляться корневым элементом управления в XAML. Можно добавлять элементы управления наподобие Grid.

К сожалению, интеграция с XAML является более сложной. Для доступа к устройству Direct3D необходим слой взаимодействия (см. Direct3DContentProvider.h class in VS project) [1, 3].

SDK содержит два шаблона проекта:

• Visual C# => Windows Phone XAML и Direct 3D: этот проект использует для воспроизведения Direct3D объект DrawingSurface;

• Visual C++ => Windows Phone Direct 3D с XAML: этот проект использует для воспроизведения

Direct3D объект DrawingSurfaceBackgroundGrid.

Эти проекты совместно используют важный код:

• Direct3DContentProvider.h - COM-класс использующий библиотеку WRL и реализующий слой взаимодействия между IDrawingSurfaceContentProvider и COM- интерфейсом IDrawingSurfaceContentProviderNative;

• Direct3DBase.h - Вспомогательный класс, который инициализирует DirectX API-интерфейсы для рендеринга 3D. Можно использовать его в вашем Direct3D-классе;

• XXX::CreateContentProvider() - добавляет слой совместимости и возвращает IDrawingSurfaceBackgroundContentProvider или IDrawingSurfaceContentProvider.

Ваш класс должен реализовывать функции, которые вызываются провайдером Direct3DContentProvider:

```
// IDrawingSurfaceContentProviderNative
HRESULT STDMETHODCALLTYPE Connect
(_In_ IDrawingSurfaceRuntimeHostNative*
host);
void STDMETHODCALLTYPE Disconnect();
HRESULT STDMETHODCALLTYPE
PrepareResources(_In_ const
LARGE_INTEGER* presentTargetTime, _Out_
BOOL* contentDirty);
HRESULT STDMETHODCALLTYPE
GetTexture(_In_ const DrawingSurfaceSizeF*
size, _Out_
IDrawingSurfaceSynchronizedTextureNative**
synchronizedTexture, _Out_
DrawingSurfaceRectF* textureSubRectangle );
```

Этот метод вызывается провайдером Direct3DContentProvider для того, чтобы предоставить Direct3D-контекст для рендеринга.

IDrawingSurfaceBackgroundContentProvider и IDrawingSurfaceContentProvider являются пустыми интерфейсами. Они используются только для ассоциации класса с целевым UI Controler.

Но XXX::CreateContentProvider () является важным:

```
IDrawingSurfaceContentProvider^
Direct3DInterop::CreateContentProvider()
{
    ComPtr<Direct3DContentProvider>
provider =
Make<Direct3DContentProvider>(this);
    return
reinterpret_cast
<IDrawingSurfaceContentProvider^>
(provider.Detach());
}
```

• Make<Direct3DContentProvider>(this); : инкапсуляция экземпляра вашего класса в объект COM Direct3DContentProvider.

• Приведите тип объекта COM к интерфейу IDrawingSurfaceContentProvider. Чтобы ассоцировать ваш рендеринг с UI Controler, необходимо использовать этот возвращаемый экземпляр.

Список литературы

1.Windows Phone developer [Электронный ресурс]. – Режим доступа: http://dev.windowsphone.com/en-us/develop.

2. Nokia developer - WiKi [Электронный ресурс]. – Режим доступа : http://www.developer.nokia.com/Community/Wiki/.

3. MSDN library, Windows Phone developer [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402533 (v=vs.105).aspx.

4.SilverlightShow. Windows Phone 8: Native Code Support [Электронный ресурс]. – Режим доступа : http://www.silverlightshow.net/items/Windows-Phone-8-Native-Code-Support.aspx

5.Справочник по языку C++ (C++/CX) [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/ru-ru/library/windows/apps/hh699871.aspx

Поступила в редколлегию5.03.2013

Рецензент: д-р техн, наук, проф. И.В. Рубан, Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков.

ПІДТРИМКА C++ ПРИ РОЗРОБЦІ ДОДАТКІВ ДЛЯ WINDOWS PHONE 8

Ю.Е. Парфьонов, В.М. Федорченко

Розглядаються аспекти використання мови програмування C++ під Windows Phone 8, а також деякі загальні міркування, якими повинен керуватися будь-який C++ розробник при виборі цільової платформи для розробки додатків. Наведені основні характеристики зазначеної платформи, переваги її використання розробниками. Обговорюються можливості налагодження додатків під Windows Phone 8, реалізації 3D-анімації та графіки.

Ключові слова: *графічний користувальницький інтерфейс, платформа, інтерфейс, клас, Direct3D, DirectX, керований код, API, C++/CX, SDK, машинний код, елемент управління, XAML, потоки, події, метадані, рендеринг, колекції. COM-об'єкти, шаблони, тип даних.*

SUPPORT C++ WHEN DEVELOPING APPLICATIONS FOR WINDOWS PHONE 8

Y.E. Parfyonov, V.M. Fedorchenko

The paper is related to the aspects of using C++ programming language under Windows Phone 8. Some general directions, which any C++ developer should guide to choose the development platform, are considered as well. The paper discusses main characteristics of the platform, the benefits of using it by developers as well as application debugging under Windows Phone 8 and the implementation of 3D animation and graphics.

Keywords: *graphical user interface, the platform, interface, class, Direct3D, DirectX, managed code, API, C++/CX, SDK, native code, control, XAML, streams, events, metadata, rendering, collections. COM-objects, templates, data type.*