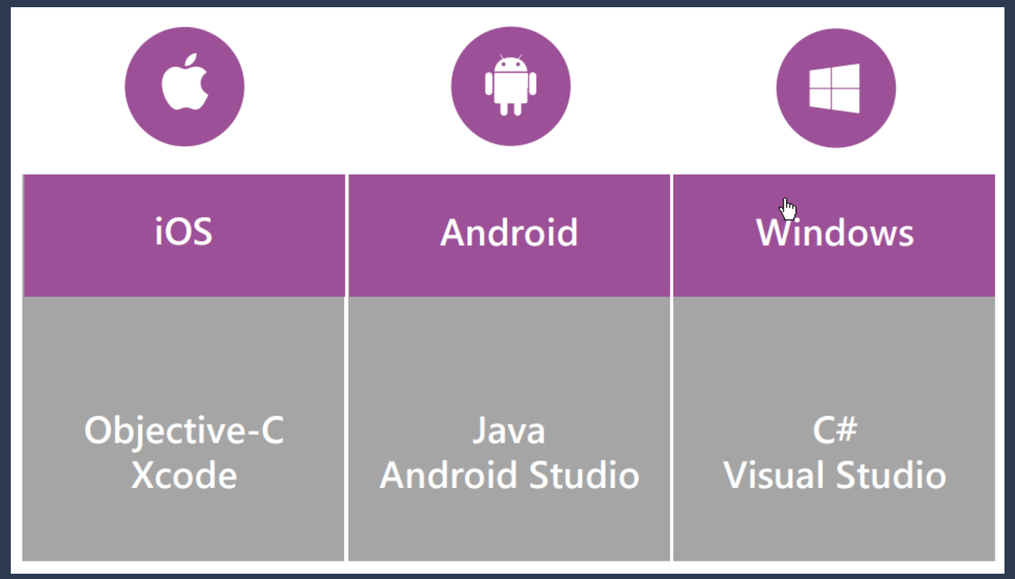
C++ via C#

Egor Bogatov (@EgorBo)
Developer at Xamarin (Microsoft)

Зачем C#-разработчику C++?

- Большое количество хороших библиотек
- До сих пор один из основных способов написания быстрого кроссплатформенного кода



- Когда не хватает скорости в C#

Как?

- COM
- C++/CLI/CX
- IPC, Sockets,…
- [DllImport]

C++ скрещенный с C# (CLI/CX)


```
m_interactionManager->SourcePressed +=
      ref new TypedEventHandler<SpatialManager^, SpatialEventArgs^>(
bind(&SpatialInputHandler::OnSourcePressed, this, _1,_2));


int main(Platform::Array<Platform::String^>^) {}
```

& * ^ . ->

C++ скрещенный с C# (CLI/CX)


```
task<void> PlayAnimationAsync()
{
  auto boxNode = GetBoxNode();
  co_await Rotate(1f, boxNode, Quaternion(0, 90, 0));
  co_await ScaleBy(1f, boxNode, Vector3(2, 2, 2));
  text->SetText("End");
}
```

Как?

- COM
- C++/CLI/CX
- IPC, Sockets,…
- [DllImport]

Как?

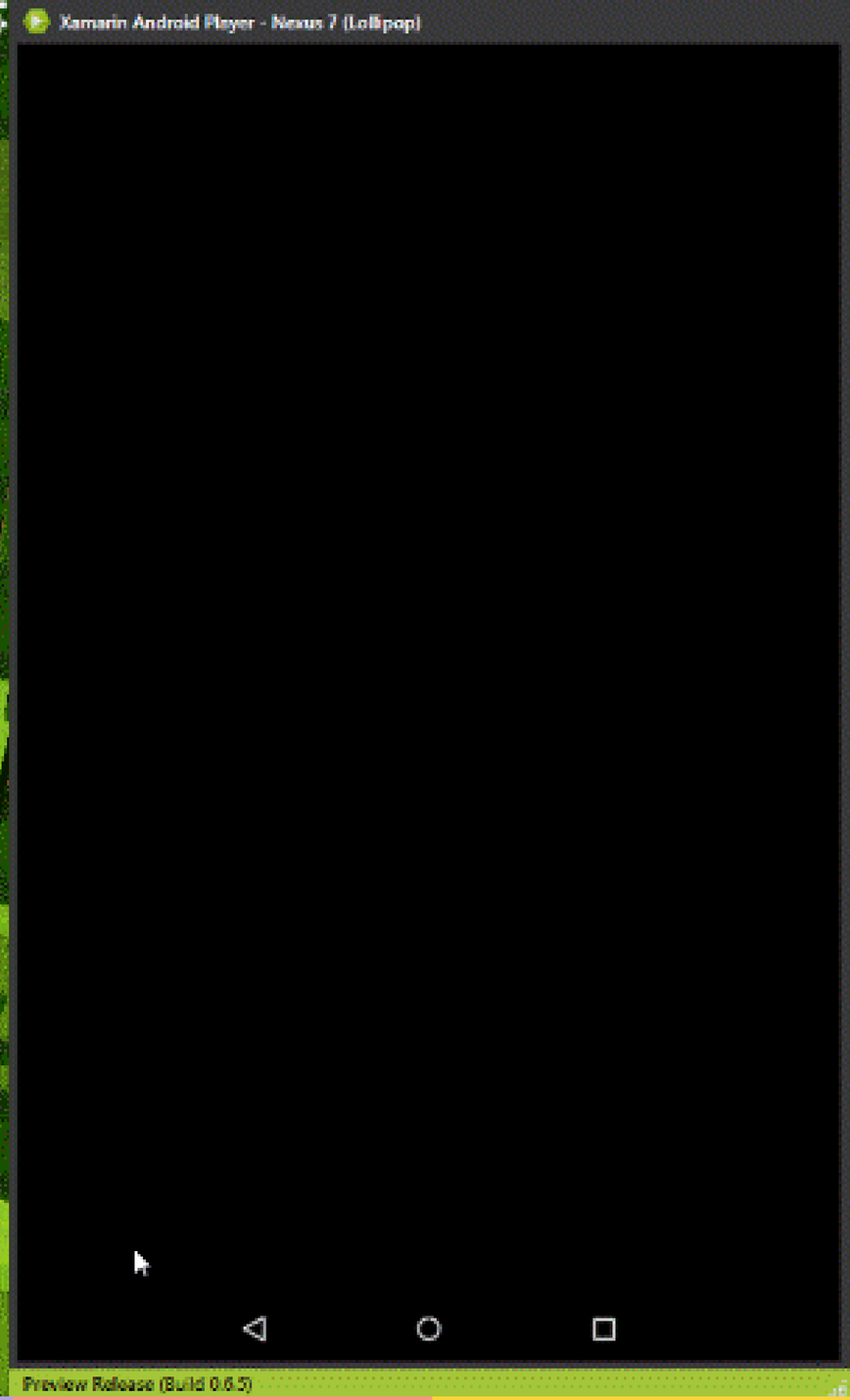
- COM
- C++/CLI/CX
- IPC, Sockets,…
- [DllImport]

Немного небезопасного сишарпа


```
// Да, это валидный C# код:
Matrix4x4 dxm4 = GetProjection();
Matrix4 m4 =  *(Matrix4*) (void*) &dxm4;

Bone* bone = model.GetBone(i);
bone->Animated = false;
```

UrhoSharp

- ~ 300 classes
- ~ 5000 methods
- Same FPS
- +10% memory
- C# for:
  - Windows
  - UWP
  - Mac OS,
  - iOS
  - Android



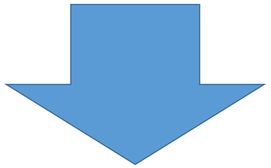
UrhoSharp для HoloLens

UrhoSharp для HoloLens

Пример перевода класса из C++ в C#

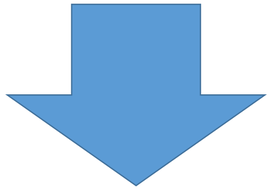

```
class Node : public Animatable
{
public:
    Node(Context* context);

    void SetName(const String& name);
};
```

```
SharedPtr<Node> node(new Node(context));
node->SetName("foo");
rootNode->AddChild(node);
```



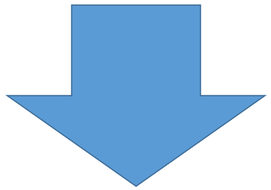
```
Node node = new Node(context);
node.SetName("foo"); //or node.Name =
rootNode.AddChild(node);
```

```
class Node : public Animatable
{
public:
     Node(Context* context);
    void SetName(const String& name);
};
```



```
DllExport void* Node_Node (Context * context) {
    return new Node(context);
}
DllExport void Node_SetName (Node *target, const char * name) {
    target->SetName (Urho3D::String(name));
}
```

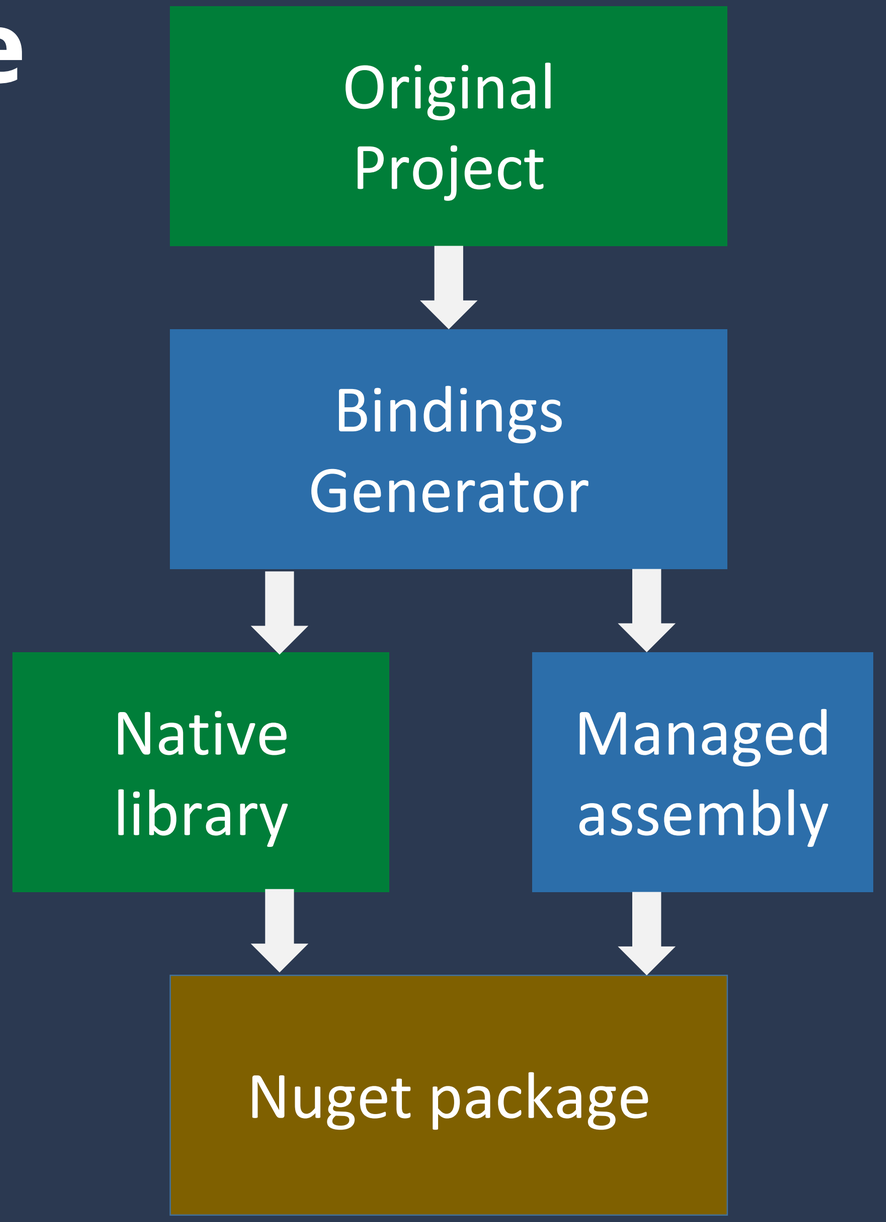
```
DllExport void* Node_Node (Context * context) {
    return new Node(context);
}
DllExport void Node_SetName (Node *target, const char * name) {
    target->SetName (Urho3D::String(name));
}
```



```
[DllImport ("mono-urho")]
static extern IntPtr Node_Node (IntPtr contextPtr);

[DllImport ("mono-urho")]
static extern void Node_SetName (IntPtr handle, string name);
```

```
public class Node : Animatable
{
    [DllImport ("mono-urho")]
    static extern IntPtr Node_Node (IntPtr context);
    [DllImport ("mono-urho")]
    static extern void Node_SetName (IntPtr handle, string name);

    public IntPtr Handle { get; set; }

    public Node (Context context)
    {
       Handle = Node_Node (context.Handle);
    }

    public void SetName (string name) => Node_SetName (Handle, name);
```

Автоматическое создание оберток



Native

Managed

Шаг 1: компиляция всего в PCH

Исходный, нетронутый немытыми руками, C++ код

Составляем pch.cpp (или генерим скриптом на коленке)

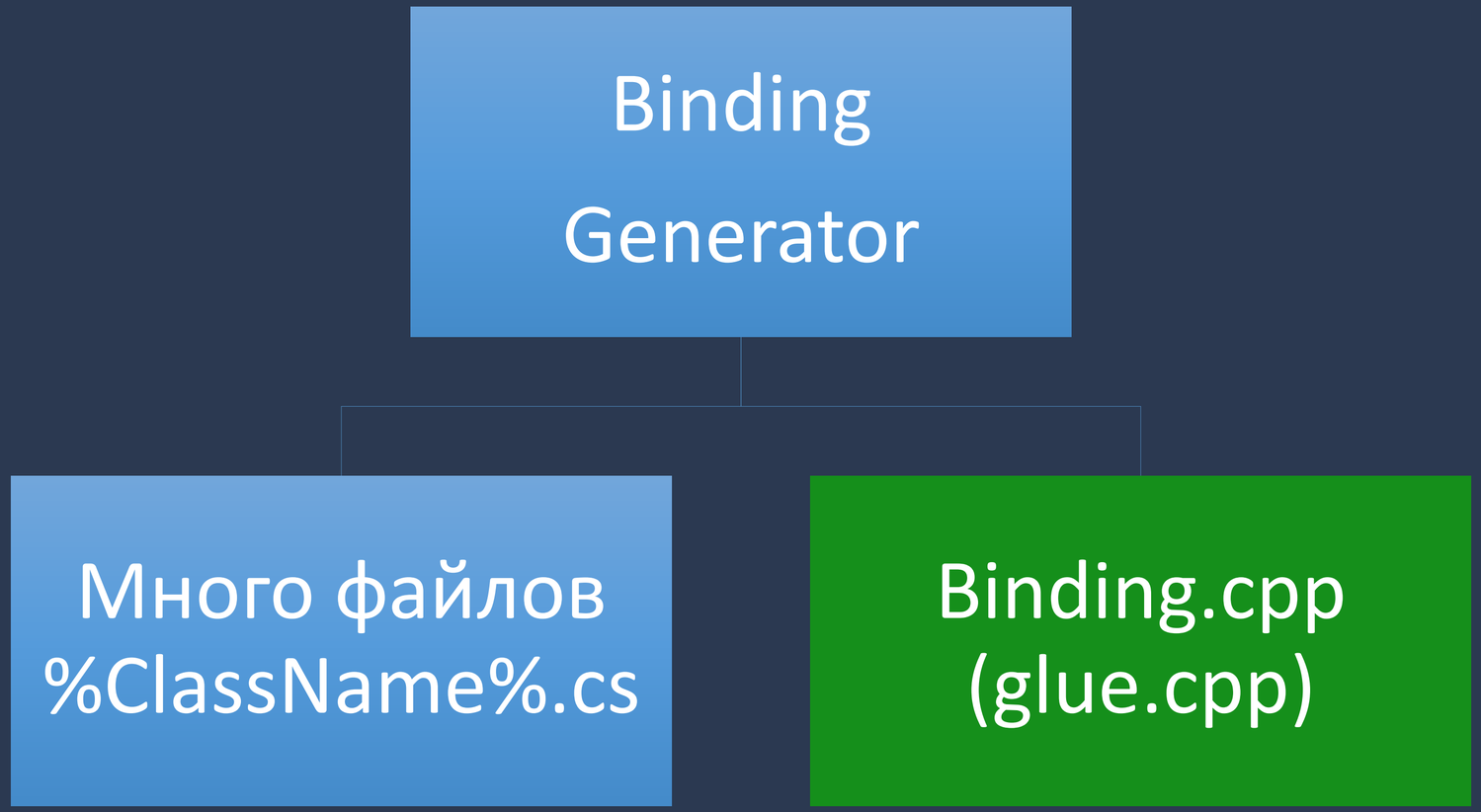
clang -cc1 -emit-pch -o pch.pch pch.cpp

Получаем на выходе .pch файлик.

Шаг 2: C# AST C++ кода


```
// skip __declspec(deprecated)
// skip non-public
// skip destructors
if (decl.HasAttr<DeprecatedAttr>() ||
    decl.Access != AccessSpecifier.Public ||
    decl is CXXDestructorDecl)
    return false;

var returnType = decl.ReturnQualType;
// “const class Urho3D::Ray &”
```

Шаг 3: Генерируем DllImports и C-glue



Шаг 4: Компилируем native library

Binding.cpp

Тот самый, нетронутый немытыми руками, оригинальный C++ код

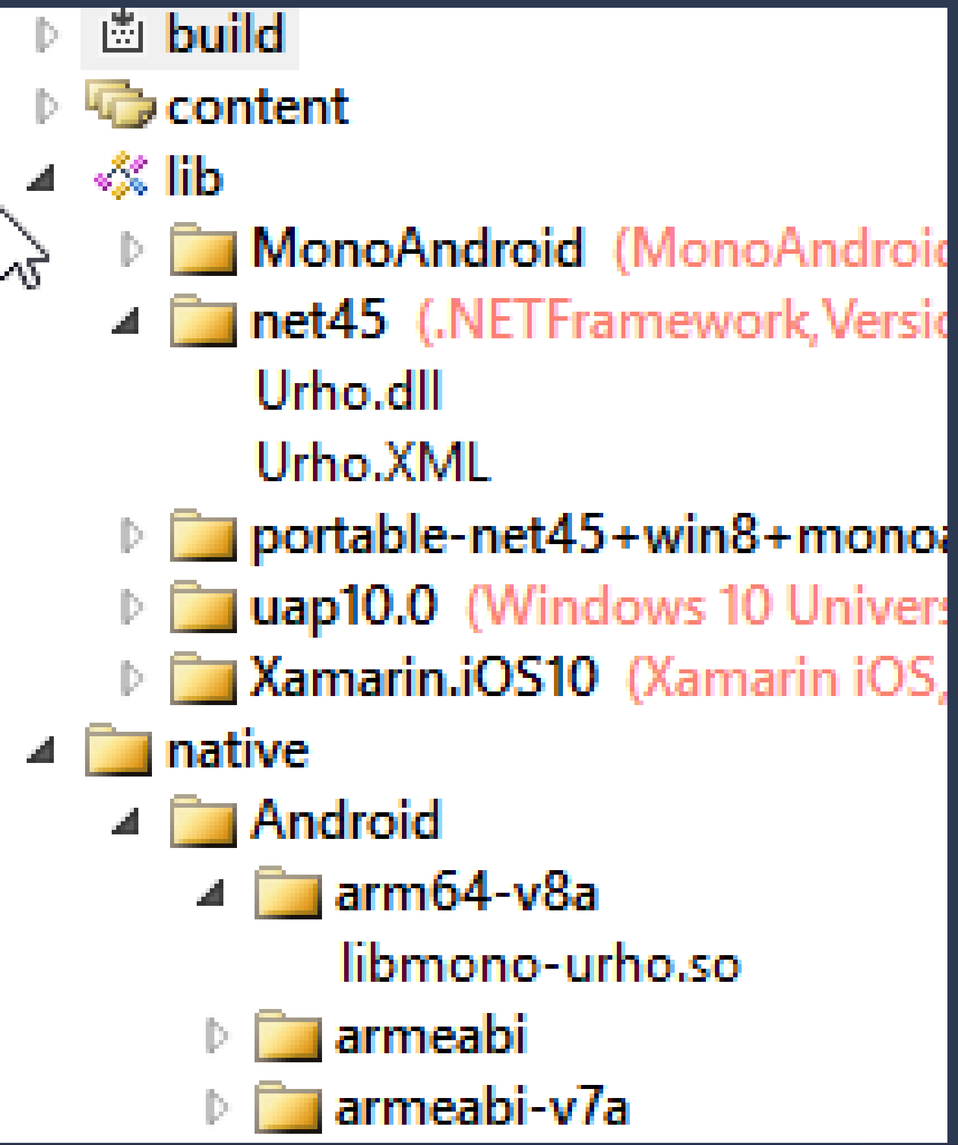
- clang
- gcc
- msvc

Win: mono-urho.dll
Droid: libmono-urho.so
OSX: libmono-urho.dylib
iOS: urho.framework

16 native libs

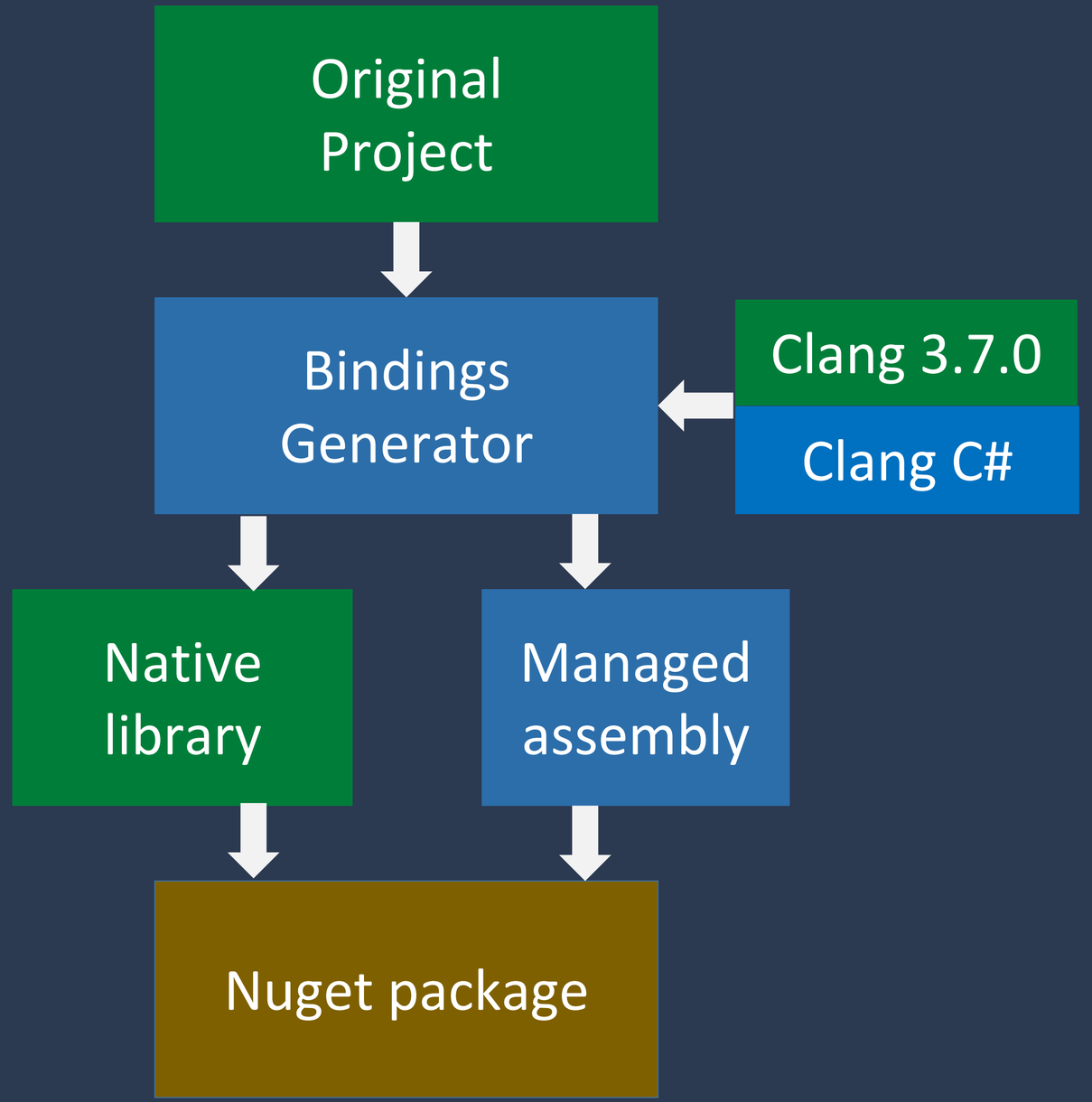
x86, 64, arm, arm64

Шаг 5: Оформляем Nuget пакет

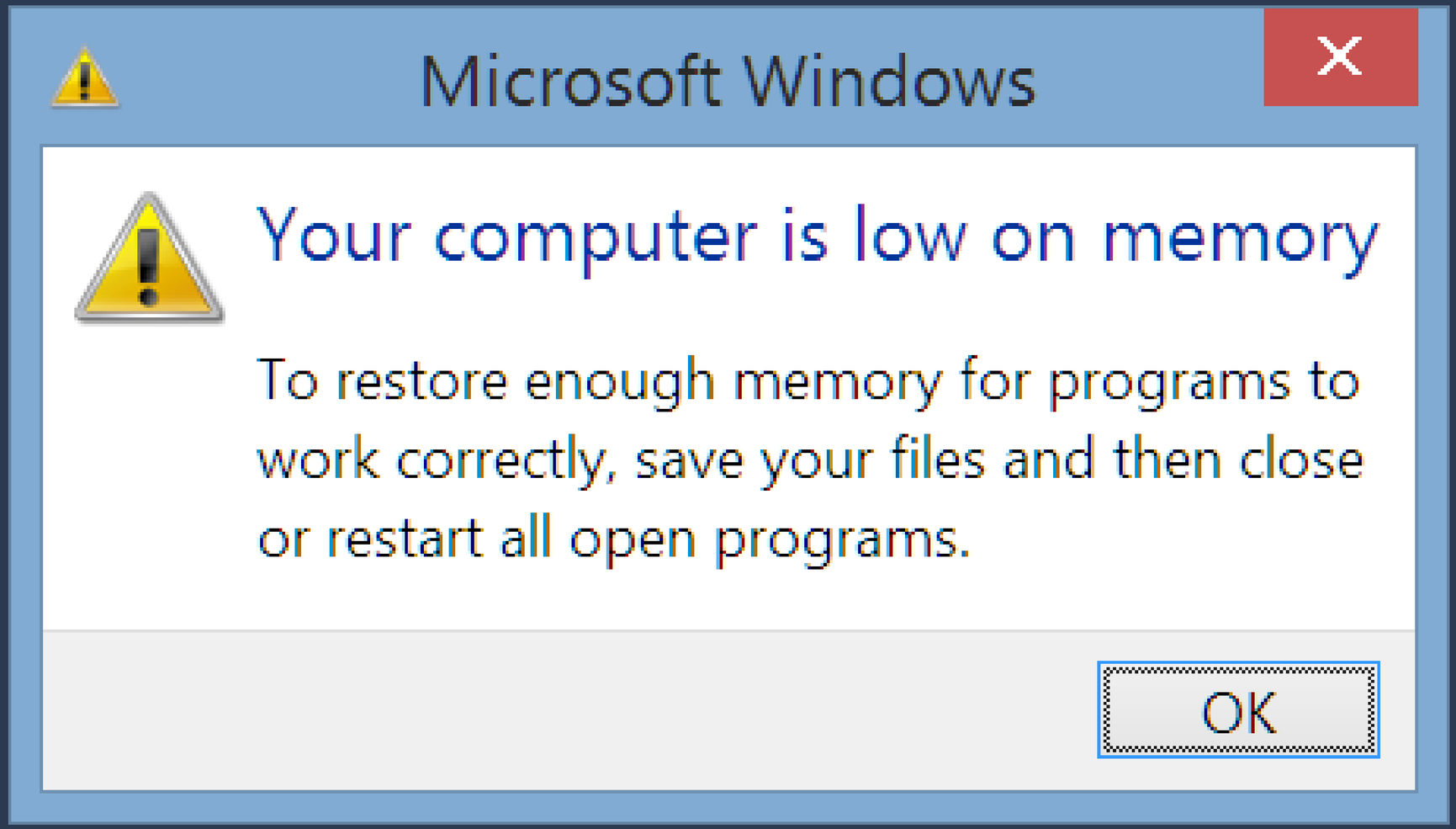



```
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
    <ItemGroup>
        <AndroidNativeLibrary Include="$(MSBuildThisFileDirectory)\..
            <Link>Libs\x86\libmono-urho.so</Link>
        </AndroidNativeLibrary>
        <AndroidNativeLibrary Include="$(MSBuildThisFileDirectory)\..
            <Link>Libs\armeabi-v7a\libmono-urho.so</Link>
        </AndroidNativeLibrary>
        <AndroidNativeLibrary Include="$(MSBuildThisFileDirectory)\..
            <Link>Libs\armeabi\libmono-urho.so</Link>
        </AndroidNativeLibrary>
        <AndroidNativeLibrary Include="$(MSBuildThisFileDirectory)\..
            <Link>Libs\arm64-v8a\libmono-urho.so</Link>
        </AndroidNativeLibrary>
        <AndroidNativeLibrary Include="$(MSBuildThisFileDirectory)\..
            <Link>Libs\x86_64\libmono-urho.so</Link>
        </AndroidNativeLibrary>
    </ItemGroup>
    <ItemGroup>
        <AndroidAsset Include="$(MSBuildThisFileDirectory)\..\..\nati
            <Link>Assets\CoreData.pak</Link>
        </AndroidAsset>
  </ItemGroup>
</Project>
```

Весь процесс



Жизненный цикл нативных объектов



Как не стоит управлять жизненным циклом нативных объектов.


```
GC.Collect();
GC.WaitForPendingFinalizers();

Marshal.FinalReleaseComObject(xlRng);
Marshal.FinalReleaseComObject(xlSheet);

xlBook.Close();
Marshal.FinalReleaseComObject(xlBook);

xlApp.Quit();
Marshal.FinalReleaseComObject(xlApp);
```

RefCounted


```
void RefCounted::AddRef()
{
    (refCount_->refs_)++;
    // callback to C#
    Mono::Callback(RefCounted_AddRef, this);
}
```

Кейс 1: Создаем объект и не используем


```
{
    var node = new Node();
}
```

- Никто в C++ без присмотра не сделал AddRef
- Храним во внутреннем статическом кэше как WeakRef
- В финализаторе C# Node удаляем нативный объект, если underlyingNativeObj.Refs() == 1. (1 т.к. одну ссылку держит C#)

Кейс 2: Создаем объект и используем


```
{
    var node = new Node();
    Scene_AddChild(node.Handle);
}
```

- AddChild где-то внутри сделает AddRef
- Мы ловим Mono::Callback AddRef и повышаем ссылку во внутреннем статическом кэше с Weak до Strong.

Кейс 2: Создаем объект и используем


```
[MonoPInvokeCallback(typeof(NativeCallback))]
void OnNativeCallback(CallbackType type, IntPtr target)
{
    case CallbackType.RefCounted_AddRef:
    {
        var referenceHolder = RefCountedCache.Get(target);
        referenceHolder?.MakeStrong();
    }
```

Кейс 3: Нативный код решает сам избавиться от нашего Node node


```
case CallbackType.RefCounted_Delete:
    {
        var referenceHolder = RefCountedCache.Get(target);
        if (referenceHolder == null)
            return;
        RefCountedCache.Remove(target)
    }
```

Доверяй, но проверяй оффсеты и сайзофы.


```
// Billboard:
static_assert(sizeof(Billboard) == 80, "Billboard has wrong size (80)");
static_assert(offsetof(Billboard, position_) == 0, "Billboard.Position has wrong offset (0)");
static_assert(offsetof(Billboard, size_) == 12, "Billboard.Size has wrong offset (12)");
static_assert(offsetof(Billboard, uv_) == 20, "Billboard.Uv has wrong offset (20)");
static_assert(offsetof(Billboard, color_) == 36, "Billboard.Color has wrong offset (36)");
static_assert(offsetof(Billboard, rotation_) == 52, "Billboard.Rotation has wrong offset (52)");
static_assert(offsetof(Billboard, direction_) == 56, "Billboard.Direction has wrong offset (56)");
static_assert(offsetof(Billboard, enabled_) == 68, "Billboard.enabled has wrong offset (68)");
static_assert(offsetof(Billboard, sortDistance_) == 72, "Billboard.SortDistance has wrong offset (72)");
static_assert(offsetof(Billboard, screenScaleFactor_) == 76, "Billboard.ScreenScaleFactor has wrong offs

// BiasParameters:
static_assert(sizeof(BiasParameters) == 12, "BiasParameters has wrong size (12)");
static_assert(offsetof(BiasParameters, constantBias_) == 0, "BiasParameters.ConstantBias has wrong offse
static_assert(offsetof(BiasParameters, slopeScaledBias_) == 4, "BiasParameters.SlopeScaleBias has wrong
static_assert(offsetof(BiasParameters, normalOffset_) == 8, "BiasParameters.NormalOffset has wrong offse

// FocusParameters:
static_assert(sizeof(FocusParameters) == 12, "FocusParameters has wrong size (12)");
static_assert(offsetof(FocusParameters, focus_) == 0, "FocusParameters.Focus has wrong offset (0)");
```

Маршалинг fixed size array поля


```
struct RenderPathCommand
{
    UrhoString textureNames_[MAX_TEXTURE_UNITS];
```

```
public struct RenderPathCommand
{
    public UrhoString TextureName0;
    ...
    public UrhoString TextureName7;

#if !IOS && !ANDROID
    public UrhoString TextureName8;
    ...
    public UrhoString TextureName15;
#endif
```

Egor Bogatov

Twitter: EgorBo
Mail: yabah@microsoft.com

github:

- xamarin/UrhoSharp
- mono/CppSharp