C++ for cross-platform VR

Nicolas Lazareff
nzff.net
@zff_n

cppcon
the c++ conference
SEPTEMBER 20-25 2015
Bellevue, Washington, USA

Imagined Reality by Benjamin Aguillon

Oculus + OTOY: *Render the Metaverse*

Fushimi Inari Forest by Riccardo Minervino

VR development *isn't easy*

“The first challenge of VR is don’t get sick.”

John Carmack, CTO Oculus

Challenges:

Cross-platform (Windows, Linux, Android, PS4)

Challenges:

Cross-platform (Windows, Linux, Android, PS4)

Proliferation of devices, languages, SDKs

Challenges:

Cross-platform (Windows, Linux, Android, PS4)

Proliferation of devices, languages, SDKs

Wildly different and non-standard input devices

Challenges:

Cross-platform (Windows, Linux, Android, PS4)

Proliferation of devices, languages, SDKs

Wildly different and non-standard input devices

Alpha/beta SDKs still riddled with bugs

“Latency – the *sine qua non* of VR.”

Michael Abrash, Chief Scientist Oculus

Today’s devices:

~ 1k per eye @ ~ 75hz

Ideal future devices:

16k per eye @ 1000hz

VR dev is a job well-suited for C++ :

cross-platform …

control over performance …

suitable as a library …

and good interfaces to other languages.

and good interfaces to other languages.

Bonus: Almost all the SDKs are primarily C++.

Food Tip #1:

Minea Farms apple juice (cider)

“Negative results are just what I want.

They’re just as valuable to me as positive results. I can never find the thing that does the job best until I find the ones that don’t.”

Thomas Edison

Progression

Separate codebases

Oculus Mobile

Cardboard

Progression

Separate codebases

Oculus Mobile

Cardboard

Shared code 100% C++

Oculus Desktop

OpenVR

Playstation VR

Progression

Separate codebases

Oculus Mobile

Cardboard

Shared code 100% C++

Oculus Desktop

OpenVR

Playstation VR

C++ with some scripting

```
VRApp :: Init()
         Frame()
         Shutdown()
```

Oculus Mobile: C++
Cardboard: Java

```
VRApp :: Init()
         Frame()
         Shutdown()
```

VRApp :: Init()

Frame()

Shutdown()

Android lifecycle

Scene init

OpenGL init

Loaders, resource managers, etc

```
VRApp :: Init()
         Frame()
         Shutdown()
```

```
Handle Input
Update()
DrawPerEye(eye)
```

```
VRApp :: Init()
         Frame()
         Shutdown()
```

Android lifecycle

Teardown

```
class MyOculusVRApp : public OVR::VrAppInterface
{

}
```

```
class MyOculusVRApp : public OVR::VrAppInterface
{
    void Configure( ovrSettings & settings )
    void OneTimeInit( const char * intentFromPackage, const char * intentJSON,
                      const char * intentURI )
    void EnteredVrMode()
}
```

```
class MyOculusVRApp : public OVR::VrAppInterface
{
    void Configure( ovrSettings & settings )
    void OneTimeInit( const char * intentFromPackage, const char * intentJSON,
                      const char * intentURI )
    void EnteredVrMode()

    Matrix4f Frame( const VrFrame & vrFrame )
    Matrix4f DrawEyeView( const int eye, const float fovDegreesX, const
                          float fovDegreesY, ovrFrameParms & frameParms )

}
```

```
class MyOculusVRApp : public OVR::VrAppInterface
{
    void Configure( ovrSettings & settings )
    void OneTimeInit( const char * intentFromPackage, const char * intentJSON,
                      const char * intentURI )
    void EnteredVrMode()

    Matrix4f Frame( const VrFrame & vrFrame )
    Matrix4f DrawEyeView( const int eye, const float fovDegreesX, const
                          float fovDegreesY, ovrFrameParms & frameParms )

    void LeavingVrMode()
    void OneTimeShutdown()
}
```

```
public class MyCarboardVRApp extends CardboardActivity
                                implements CardboardView.StereoRenderer
{

}
```

```
public class MyCarboardVRApp extends CardboardActivity
                            implements CardboardView.StereoRenderer
{

    void onSurfaceCreated( EGLConfig config )









}
```

```
public class MyCarboardVRApp extends CardboardActivity
                              implements CardboardView.StereoRenderer
{

    void onSurfaceCreated( EGLConfig config )


    void onNewFrame( HeadTransform headTransform )

    void onDrawEye( Eye eye )




}
```

```
public class MyCarboardVRApp extends CardboardActivity
                            implements CardboardView.StereoRenderer
{

    void onSurfaceCreated( EGLConfig config )


    void onNewFrame( HeadTransform headTransform )

    void onDrawEye( Eye eye )


    void onRendererShutdown()

}
```

```
typedef struct ovrPosef_
{
        ovrQuatfOrientation;
        ovrVector3fPosition;
} ovrPosef;
```

```
typedef struct ovrPosef_
{
        ovrQuatfOrientation;
        ovrVector3fPosition;
} ovrPosef;
```

Orientation of user’s head

```
typedef struct ovrPosef_
{
        ovrQuatfOrientation;
        ovrVector3fPosition;
} ovrPosef;
```

Position of user’s head

```
Matrix4f MyOculusVRApp :: Frame( const VrFrame & vrFrame )

{

    // VrFrame has head pose state
    // Here we update our scene and store the pose …

    //
    // …
    //

}
```

```
Matrix4f MyOculusVRApp :: Frame( const VrFrame & vrFrame )

{

    // VrFrame has head pose state
    // Here we update our scene and store the pose …

    //
    // …
    //

}
```

Gets called twice (two eyes… two calls)

```
Matrix4f MyOculusVRApp :: DrawEyeView( const int eye, const float fovDegreesX,
                                       float fovDegreesY, ovrFrameParms & frameParms )
```

```
Matrix4f MyOculusVRApp :: DrawEyeView( const int eye, const float fovDegreesX,
                                       float fovDegreesY, ovrFrameParms & frameParms )

{

    // Use pose to calculate model view matrix
    // Calculate model view projection matrix
    // Draw per eye

    // OpenGL / drawing commands

    //
    // …
    //

}
```

Food Tips #2 & #3:

Facing East (Taiwanese – *try the pork burger*)
Mediterranean Kitchen (Lebanese)

Oculus Mobile: C++

Cardboard: Java

```
VRApp :: Init()
         Frame()
         Shutdown()
```

Entirely C++

```
VRApp :: Init()
         Frame()
         Shutdown()
```

VRApp :: Init()

Frame()

Shutdown()

Sometimes let SDK init window, OpenGL, eye buffers

Init our scene, textures, loaders, etc

VRApp :: Init()

Frame()

Shutdown()

SDK handles buffer swap, distortion, timewarp, etc

Get head pose from SDK to draw our scene

Respond to user input

```
VRApp :: Init()
         Frame()
         Shutdown()
```

Let SDK shutdown

Do our own cleanup

```
class VRApplication
{

}
```

```
class VRApplication
{
    void Configure( Settings & settings )
    void Initialize( LaunchConfig & config )
    void EnteredVrMode()

}
```

```
class VRApplication
{
    void Configure( Settings & settings )
    void Initialize( LaunchConfig & config )
    void EnteredVrMode()

    void Frame( const Frame & frame )
    void DrawPerEye( const Eye & eye )



}
```

```
class VRApplication
{
    void Configure( Settings & settings )
    void Initialize( LaunchConfig & config )
    void EnteredVrMode()

    void Frame( const Frame & frame )
    void DrawPerEye( const Eye & eye )

    void HandleInput( const Input & input )




}
```

```
class VRApplication
{
    void Configure( Settings & settings )
    void Initialize( LaunchConfig & config )
    void EnteredVrMode()

    void Frame( const Frame & frame )
    void DrawPerEye( const Eye & eye )

    void HandleInput( const Input & input )

    void LeavingVrMode()
    void Shutdown()
}
```

VRApplication.h

```
#if defined ( OCULUS_MOBILE )
   void Frame( const VrFrame & vrFrame )
#endif
```

VRApplication_OculusMobile.cpp

```
void VRApplication :: Frame( const VrFrame & vrFrame )
```

VRApplication_Cardboard.cpp

```
void Java_com_cppcon_VRApplication_nativeOnCreate( JNIEnv * jni, jclass clazz, … )

void Java_com_cppcon_VRApplication_nativeOnFrame( JNIEnv * jni, jclass clazz, … )

void Java_com_cppcon_VRApplication_nativeOnDraw( JNIEnv * jni, jclass clazz, … )

void Java_com_cppcon_VRApplication_nativeOnDestroy( JNIEnv * jni, jclass clazz, … )


//
// Endless JNI
//
```

VRApplication_Cardboard.cpp

JNI … if you are brave

Java API or plans for one?

QUOTE

by **wwwtyro** » December 8th, 2014, 6:51 pm

Is there a java API for the mobile SDK, or do we need to write our apps mostly in C++?

If there is, where can I find the documentation for it? If there isn't, are there plans for one?

Thanks!

Re: Java API or plans for one?

QUOTE

by **chrispruett** » December 9th, 2014, 12:01 am

There is not, but you can call from Java down into the C++ API via JNI if you are brave.

VRApplication_OpenVR.cpp

```
void VRApplication :: MainLoop ()
{

    //
    // while( running )
    //

}
```

VRApplication_OpenVR.cpp

vr::IVRSystem

- Tracking
- Distortion
- Events (e.g. controller)

VRApplication_PlaystationVR.cpp

```
void VRApplication :: MainLoop ()
{

    //
    // while( running )
    //

}
```

VRApplication_PlaystationVR.cpp

```
void VRApplication :: MainLoop ()
{

    //
    // while( running )
    //

}
```

libSceHmd
libSceVrTracker

} All the usual VR stuff

VRApplication_OculusDesktop.cpp

** curiously quite different to Oculus Mobile*

LibOVR:

```
ovr_Initialize()

    ovr_GetTrackingState()
    ovr_SubmitFrame()

ovr_Shutdown()
```

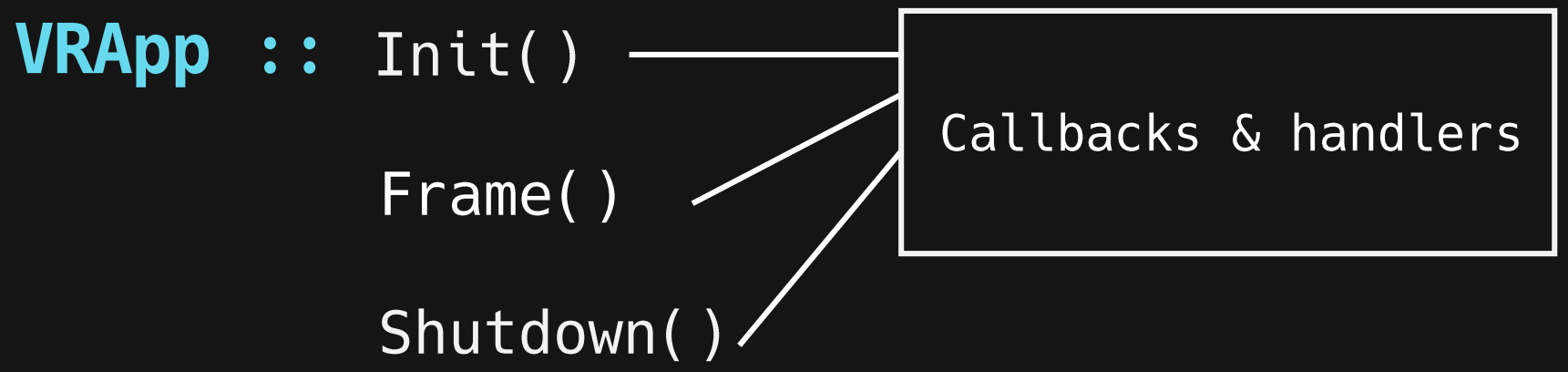
Every platform is the *same*, *but different*

```
VRApp :: Init()
         Frame()
         Shutdown()
```

Primarily C++

Additional scripting layer

```
VRApp :: Init()
         Frame()
         Shutdown()
```


VRApp ::
Init()
Frame()
Shutdown()
Callbacks & handlers

What’s next?

What’s next?

* Uniformity

What’s next?

* Uniformity
* Further scripting

What’s next?

* Uniformity
* Further scripting
* Utilization of platform ‘nice-to-haves’

What’s next?

* Uniformity
* Further scripting
* Utilization of platform ‘nice-to-haves’
* Attempt to tame input


For pizza, try **Serious Pie**
(Buffalo Mozzarella Pizza)

Thanks for listening,

Questions?

Apple juice/cider: Minea Farms
Lebanese: Mediterranean Kitchen
Taiwanese: Facing East
Pizza: Serious Pie