

Game development on Android

Using the NDK



Overview

- Background
- Porting to JNI
 - Differences to iPhone
 - Basic JNI
 - C++ Threads under JNI
- Debugging
 - (OMG This is B***shit!)
- Some OpenGL tips and tricks



Who am I?

- Matthew Clark matt@thevoxelagents.com
- Programmer/Founder of The Voxel Agents
 - We make games - Original IP
 - iPhone games
 - Train Conductor
 - Train Conductor 2
- Currently in the process of porting to Android



Voxel Engine

- Not a 'Voxel' Engine
- OpenGL engine originally for iPhone
 - C++ as much as possible
 - ObjectiveC for:
 - File access
 - Saving / Loading users data
 - Sounds
 - Texture loading
 - Guess what we need to rewrite? :)



There is more!

- No STL (need to compile and link STLPort)
- No Exceptions
 - Various libraries need tweaking/changing
 - TinyXML
 - JSON
- Minor compiler differences
- Our pipeline is heavily integrated with xcode



And more...

- Many devices means different resolutions and aspect ratios
- Even worse... different video cards!!
 - With different subsets of OpenGL ES extensions
 - No standard texture compression!
 - iPhone uses PVR (4bpp)
 - Smallest standard compression on android is RGB565 (16bpp) – no alpha!
 - For a 1024x1024 texture 512K vs 2Mb !



JNI - The Basics

Calling C from Java

- Really Easy
 - In .java:

```
class VoxelGLSurfaceView extends GLSurfaceView
{
    public boolean onTouchEvent(final MotionEvent event)
    {
        touchEvent(event.getAction(), event.getEventTime(), event.getX(), event.getY());
        return true;
    }

    private static native void touchEvent(int action, float time, float x, float y);
}
```

- In .cpp :

```
void Java_com_tva_Voxel_VoxelGLSurfaceView_touchEvent( JNIEnv* env, jobject this,
                                                       jint touchEventType, jfloat time, jfloat x, jfloat y )
{
    Core::Vec2 touchPos(x - backingWidth/2, y - backingHeight/2);
    touchPos = touchPos * Core::Vec2(480.f / backingWidth, 320.f / backingHeight);

    Input::TouchInput input(touchPos, touchPos, time, 0, (Input::TouchInputPhase)touchEventType, 0);
    inputStack.push_back(input);
}
```


Calling Java from C

- Bit trickier

```
{
    // Get the environment
    JNIEnv* env = GetJEnv();

    // find the class
    cls = env->FindClass("com/tva/Voxel/FileReader");

    // register the methods
    ctor = env->GetMethodID(cls, "<init>", "(Ljava/lang/String;)V");
    readMethod = env->GetMethodID(cls, "GetBytes", "[B)I");

    // make a string to pass to java
    jstring mystr = env->NewStringUTF(fullPath);

    // Create the object invoking the constructor
    jobject reader = env->NewObject(cls, ctor, mystr);

    // Call GetBytes
    jbyteArray bytes = env->NewByteArray(numBytes);
    jint bytesRead = env->CallIntMethod((jobject)reader, readMethod, bytes);
    jbyte *bPtr = env->GetByteArrayElements(bytes, JNI_FALSE);

    // do something with bPtr
    |
}
```

GLSurfaceView

```
class DemoRenderer implements GLSurfaceView.Renderer
{
    public void onSurfaceCreated(GL10 gl, EGLConfig config)
    {
        nativeInit();
    }

    public void onSurfaceChanged(GL10 gl, int w, int h)
    {
        nativeResize(w, h);
    }

    public void onDrawFrame(GL10 gl)
    {
        nativeRender();
    }

    private static native void nativeInit();
    private static native void nativeResize(int w, int h);
    private static native void nativeRender();
}
```



Loading textures

- Phil Hassey's blog about porting Galcon to android is awesome!
 - Files in `/assets/`* are directly accessible by filename

```
InputStream stream = app.getAssets().open(filename);  
Bitmap bitmap = BitmapFactory.decodeStream(stream);  
gl.glBindTexture(GL10.GL_TEXTURE_2D, textureID);  
GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmap, 0);
```

- Thats it!
- <http://www.philhassey.com/blog/2010/08/03/porting-galcon-using-the-android-ndk/>



Sounds

- SoundPool (SoundFX)
 - Specify how many sounds you want to play at once
 - When you play too many sounds, it will stop playing the earliest one
 - This is okay behaviour, but sometimes you want the opposite (requires manual tracking)
- AudioManager (Music)
 - `create()`
 - `setLooping()`
 - `start()`



pthread

```
JNIEnv* GetThreadJNIEnv()
{
    int status;
    JNIEnv *env;

    const U32 threadId = pthread_self();

    if (g_JNIEnvMap.find(threadId) == g_JNIEnvMap.end())
    {
        // is this even in a special thread?
        status = gJavaVM->GetEnv((void **) &env, JNI_VERSION_1_4);
        if (status < 0)
        {
            DLog("callback_handler: failed to get JNI environment, "
                "assuming native thread");

            status = gJavaVM->AttachCurrentThread(&env, NULL);
            ASSERT(status == 0);
        }
    }

    return g_JNIEnvMap[threadId];
}
```

C++ threads can't
access the JVM
directly

They need to have a
Java Native
Environment created
for them

AttachCurrentThread()

<http://android.wooyd.org/JNIExample/>



pthread continued...

- Make sure you Detach the thread

```
void EndThread()
{
    JNIEnvMap::iterator iter =
        g_JNIEnvMap.find(pthread_self())

    if (iter != g_JNIEnvMap.end())
    {
        gJavaVM->DetachCurrentThread();
        g_JNIEnvMap.erase(iter);
    }

    pthread_exit(NULL);
}
```

Gotcha

- Because the thread is running outside of the GLContext – we can't do any GL operations
 - Say good bye to threaded texture loading!



Debugging JNI

- No breakpoints in native code
- Poor stack tracing
 - Even worse inside Java!!!!
- Emulator is terrible (1-3 fps)
- Super headache!!!



USELESS INFORMATION*

Time	pid	tag	Message
11-03 11:40:11.331	I	11909	DEBUG signal 11 (SIGSEGV), fault addr deadbaad
11-03 11:40:11.331	I	11909	DEBUG r0 00000000 r1 afd14629 r2 00000027 r3 00000070
11-03 11:40:11.331	I	11909	DEBUG r4 afd42328 r5 00000000 r6 00000000 r7 43700000
11-03 11:40:11.331	I	11909	DEBUG r8 486ddb08 r9 41866f50 10 808a13f4 fp 41866f50
11-03 11:40:11.331	I	11909	DEBUG ip 00001728 sp 486dd6c8 lr deadbaad pc afd11c80 cpsr 60000030
11-03 11:40:11.331	I	11909	DEBUG d0 6472656767756265 d1 6568636e61726273
11-03 11:40:11.331	I	11909	DEBUG d2 736e696172542f74 d3 64696f72646e612e
11-03 11:40:11.331	I	11909	DEBUG d4 2f2e2e2f696e6a2f d5 6372756f532f2e2e
11-03 11:40:11.331	I	11909	DEBUG d6 6968706172472f65 d7 4c747265562f7363
11-03 11:40:11.331	I	11909	DEBUG d8 00000000c1500000 d9 43e0000000000000
11-03 11:40:11.331	I	11909	DEBUG d10 41dfffffff000000 d11 c3e0000000000000
11-03 11:40:11.341	I	11909	DEBUG d12 0000000000000000 d13 0000000000000000
11-03 11:40:11.341	I	11909	DEBUG d14 0000000000000000 d15 0000000000000000
11-03 11:40:11.341	I	11909	DEBUG d16 3fee016a2a0378af d17 3f60603e70e8d7a3
11-03 11:40:11.341	I	11909	DEBUG d18 bf56b3a4d5793dde d19 3f903fe9f6e55711
11-03 11:40:11.341	I	11909	DEBUG d20 3fa5553e1053a42 d21 3fc3746e4fe777a5
11-03 11:40:11.341	I	11909	DEBUG d22 0000000000000000 d23 3ef99342e0ee5069
11-03 11:40:11.341	I	11909	DEBUG d24 3ef99342e0ee5069 d25 3fe7f7c300000000
11-03 11:40:11.341	I	11909	DEBUG d26 be30f58ce0000000 d27 3ef99342e0ee5069
11-03 11:40:11.341	I	11909	DEBUG d28 0000000000000000 d29 3ff0000000000000
11-03 11:40:11.341	I	11909	DEBUG d30 0000000000000000 d31 3ff0000000000000
11-03 11:40:11.341	I	11909	DEBUG scr 80000012
11-03 11:40:11.451	I	11909	DEBUG #00 pc 00011c80 /system/lib/libc.so
11-03 11:40:11.451	I	11909	DEBUG #01 pc 00018f5c /system/lib/libc.so
11-03 11:40:11.451	I	11909	DEBUG #02 pc 000522a0 /data/data/com.tva.Voxel/lib/libVoxelEngine.so
11-03 11:40:11.451	I	11909	DEBUG #03 pc 00050408 /data/data/com.tva.Voxel/lib/libVoxelEngine.so
11-03 11:40:11.451	I	11909	DEBUG #04 pc 00050da0 /data/data/com.tva.Voxel/lib/libVoxelEngine.so
11-03 11:40:11.451	I	11909	DEBUG #05 pc 00149248 /data/data/com.tva.Voxel/lib/libVoxelEngine.so
11-03 11:40:11.451	I	11909	DEBUG code around pc:
11-03 11:40:11.451	I	11909	DEBUG afd11c60 2d00682d e029d1fb b12b68db c05cf8df
11-03 11:40:11.451	I	11909	DEBUG afd11c70 f8442001 4798000c e054f8df 26002227
11-03 11:40:11.451	I	11909	DEBUG afd11c80 2000f88e eee4f7fb f7fd2106 f04fe802
11-03 11:40:11.451	I	11909	DEBUG afd11c90 91035180 460aa901 96012006 f7fc9602
11-03 11:40:11.451	I	11909	DEBUG afd11ca0 a905eb88 20024632 eb92f7fc eed0f7fb
11-03 11:40:11.451	I	11909	DEBUG code around lr:
11-03 11:40:11.451	I	11909	DEBUG deadba8c ffffffff ffffffff ffffffff ffffffff
11-03 11:40:11.451	I	11909	DEBUG deadba9c ffffffff ffffffff ffffffff ffffffff
11-03 11:40:11.451	I	11909	DEBUG deadbaac ffffffff ffffffff ffffffff ffffffff
11-03 11:40:11.451	I	11909	DEBUG deadbab0 ffffffff ffffffff ffffffff ffffffff
11-03 11:40:11.451	I	11909	DEBUG deadbabc ffffffff ffffffff ffffffff ffffffff
11-03 11:40:11.451	I	11909	DEBUG deadbacc ffffffff ffffffff ffffffff ffffffff
11-03 11:40:11.451	I	11909	DEBUG stack:
11-03 11:40:11.451	I	11909	DEBUG 486dd688 00000015
11-03 11:40:11.451	I	11909	DEBUG 486dd68c afd14659 /system/lib/libc.so
11-03 11:40:11.451	I	11909	DEBUG 486dd690 afd425a0 /system/lib/libc.so
11-03 11:40:11.451	I	11909	DEBUG 486dd694 afd4254c /system/lib/libc.so
11-03 11:40:11.451	I	11909	DEBUG 486dd698 00000000
11-03 11:40:11.451	I	11909	DEBUG 486dd69c afd15673 /system/lib/libc.so
11-03 11:40:11.451	I	11909	DEBUG 486dd6a0 afd14629 /system/lib/libc.so
11-03 11:40:11.451	I	11909	DEBUG 486dd6a4 afd14629 /system/lib/libc.so
11-03 11:40:11.451	I	11909	DEBUG 486dd6a8 00000070
11-03 11:40:11.451	I	11909	DEBUG 486dd6ac afd42328 /system/lib/libc.so
11-03 11:40:11.451	I	11909	DEBUG 486dd6b0 00000000
11-03 11:40:11.451	I	11909	DEBUG 486dd6b4 486dd6dc
11-03 11:40:11.451	I	11909	DEBUG 486dd6b8 43700000
11-03 11:40:11.451	I	11909	DEBUG 486dd6bc afd148cb /system/lib/libc.so



*from the perspective of a human

arm-eabi-addr2line

- In the NDK tools
- Converts useless information into *slightly* more useful information
 - Doesn't always have line numbers
 - No object inspection
 - Ouch Ouch Ouch

```
libVoxelEngine.so 00057328 _ZNK8Graphics6Sprite12DrawContentsEv ??:0
libVoxelEngine.so 000297b8 _ZNK8Graphics13DisplayObject4DrawEv ??:0
libVoxelEngine.so 000295e8 _ZNK8Graphics13DisplayObject12DrawChildrenEv ??:0
libVoxelEngine.so 000297c0 _ZNK8Graphics13DisplayObject4DrawEv ??:0
libVoxelEngine.so 000295e8 _ZNK8Graphics13DisplayObject12DrawChildrenEv ??:0
libVoxelEngine.so 000297c0 _ZNK8Graphics13DisplayObject4DrawEv ??:0
libVoxelEngine.so 000295e8 _ZNK8Graphics13DisplayObject12DrawChildrenEv ??:0
libVoxelEngine.so 000297c0 _ZNK8Graphics13DisplayObject4DrawEv ??:0
libVoxelEngine.so 000dbec8 _ZN7Project4Game10InlineDrawEv ??:0
libVoxelEngine.so 000dc144 _ZN7Project4Game4DrawEv ??:0
libVoxelEngine.so 001d8108 Java_com_tva_Voxel_VoxelRenderer_nativeRender ??:0
```



Solution?

- Lots of logging
- Use logcat where possible to filter information
 - `__android_log_print(ANDROID_LOG_INFO, "VoxelEngine", "blah")`
 - `__android_log_print(ANDROID_LOG_DEBUG, "VoxelTextures", "blah")`
 - `__android_log_print(ANDROID_LOG_WARNING, "VoxelFonts", "blah")`
 - Filter using debug levels and tags
 - `Filter: VoxelEngine:W VoxelTextures:I *:S`
- Lots of Panadol / Asprin / Valium
- Don't create bugs?



NDK-GDB

- Finally! Available on android 2.2 phones!
- Line numbers! OMG!!!
 - I haven't used it successfully yet

```
Program received signal SIGSEGV, Segmentation fault.  
0x80300bae in Java_com_example_hello gdbserver_HelloGdbServer_invokeCrash  
(env=0xaa50, clazz=0x4625f0b8)  
  at /home/vilimpoc/android-ndk-r4b/samples/hello-gdbserver/jni/hello-  
gdbserver.c:29  
29          *crasher = 0xdeaddead;  
  
(gdb)
```

<http://vilimpoc.org/blog/2010/09/23/hello-gdbserver-a-debuggable-jni-example-for-android/>



OpenGL

- The first rule of OpenGL development is...
 - Avoid State Changes
- The second rule is...
 - AVOID STATE CHANGES! :)

Open GL



- Use texture sheets
- Don't rebind a texture that is already bound
 - Cache the last bound textureID
 - Fewer texture swaps
- Textures are rarely square anyway...
 - save texture space!

Open GL

- In our game, we have 5 different types of vert lists
 - x,y,u,v,colour (Sprites with vert colours)
 - x,y,z,nx,ny,nz,u,v (3D model)
 - x,y,u,v (Regular Sprites)
 - x,y (super simple shapes – mostly debug only)
 - x,y,size,col (Point sprites (particles))
- Each different type needs a different setup

```

enum DrawState
{
    S_VertexXYUVCol,
    S_Vertex3D,
    S_VertexXYUV,
    S_VertexXYUVShort,
    S_VertexXY,
    S_VertexXYSizeCol,
};

DrawState prevState = S_VertexXYUV;
inline void KillPrevDrawState(DrawState newState)
{
    switch (prevState)
    {
        case S_VertexXYUVCol: glDisableClientState(GL_COLOR_ARRAY); break;
        case S_Vertex3D: glDisableClientState(GL_NORMAL_ARRAY); break;
        case S_VertexXYUVShort:
        case S_VertexXYUV: /*default*/ break;
        case S_VertexXY: glEnable(GL_TEXTURE_2D);
            glEnableClientState(GL_TEXTURE_COORD_ARRAY); break;
        case S_VertexXYSizeCol: glDisable(GL_POINT_SPRITE_OES);
            glDisableClientState(GL_POINT_SIZE_ARRAY_OES);
            glDisableClientState(GL_COLOR_ARRAY); break;
    }
    prevState = newState;
}
VertexXYUVCol::Draw()
{
    glBindBuffer(GL_ARRAY_BUFFER, buff);
    if (prevState != S_VertexXYUVCol)
    {
        KillPrevDrawState(S_VertexXYUVCol);
        glEnableClientState(GL_COLOR_ARRAY);
    }
    glVertexPointer(2, GL_SHORT, sizeof(VertexXYUVCol), VERT_OFFSET(VertexXYUVCol,x));
    glTexCoordPointer(2, GL_FLOAT, sizeof(VertexXYUVCol), VERT_OFFSET(VertexXYUVCol,u));
    glColorPointer(4, GL_UNSIGNED_BYTE, sizeof(VertexXYUVCol), VERT_OFFSET(VertexXYUVCol,col));
    glDrawArrays(drawMode, 0, numVerts);
}

```

- Remember the previous state
- Only change what you need to change

OpenGL

- Use vertex buffers
- Use `GL_SHORT` where possible for vertex data.
 - Shorts are half the size of floats
 - Some androids don't support floating points :(