Java & Games

A rivalrous case-study from porting Doom 3
Nothing is true, Everything is permitted
Prologue
INTRO

What is...djoom3?

✘ Cool "temporary" name
  ○ https://github.com/blackbeard334/djoom3

✘ Why?
  ○ Java...because?

✘ Lustrum

✘ Ingredients:
  ○ Java
  ○ OpenGL/OpenAL(LWJGL)

✘ Questions/remarks/heckles
  ○ @blackbeard0x14e
What is the Java?

- Interpre-piled-ish©
- Open source-ish
- Signed math
- Pass by (reference) value :(  
  - Supports Null Pointer Exceptions ;)
- Operator Overloading not supported
- final is like const, except not really
The JVM is a treasure just sitting there waiting to be rediscovered. It's the result of man-centuries (man-millennia?) of work, yes, and as a result is really, really impressive. But imagine if that much work had been put into something fundamentally great, like Smalltalk or Lisp, rather than something fundamentally okay like Java.

It really is a shame that there is so much noise and unnecessary complexity around using it.

Very true. Java itself is not a terrible language (hence why I wrote 'fundamentally okay' above); it's the arcane, rococo, lunatic levels of ritual which surround it. Every time I look at Java code I'm reminded of some insane fantastic court in which one has to plead with the Minister of Small Affairs, make a sacrifice to the God of Bureaucracy, pay for a Token of Token-Paying, appeal to the good graces of the Wise Undersecretary of Vice-Small Affairs and finally spend a weekend climbing an Escher Staircase before an audience with the King, all just to get a scrap of toilet paper.
INTRO

Seriously though...what is the Java?

✘ JLS – Spec
✘ JDK – Language
✘ JVM
  ○ GC
  ○ JIT
  ○ JMM(middleware)
Light it up
Ah, light it up
Another hit erases all the pain
Environment setup

- OpenGL dll/linker/macro bullshite
  - 64bit vs 32bit
- Visual studio project compatibility
- 6 pages [doom3.gpl] Compiling Doom 3/idtech 4 source code
- 19 pages DOOM3 COMPILATION INSTRUCTIONS FOR MAC OS X.
- Money money money...must be funny...in the rich man’s world!
- Scone Scons fiasco...T_T
  - Or chromium...

The original release from TTimo compiles well with Visual Studio 2010 Professional. Unfortunately Visual Studio 2010 "Express" lacks MFC and hence cannot be used. This was disappointing upon release but some people have since removed the dependencies.
Environment setup

✖ Wizardry
✖ Open source based
✖ Paths...bloody paths
✖ Subtle 32bit vs 64bit
  ○ Especially for native code
Build tools

Maven vs xmake

✘ Single pom of failure
  ○ Dependencies
  ○ Profiles
  ○ Testing
Build tools

- Project version
  - SNAPSHOT
- Dependencies
  - Transitive dependencies yay!
- Natives
- Testing
- Ecosystem
- Build servers
Need for speed!

The new /d2ReducedOptimizeHugeFunctions flag in Visual Studio 2019 v16.4 brought our TensorFlow 2.1.0 release builds down from over 48 hours (VS 2017) to under 4.

확실히 이는 큰 승리입니다! @visualc 팀의 흐름을 위해 감사드립니다.

```
- This is a huge win! Thanks to the @visualc team for their work on this.
```
### Compile times

**Clean & Build:**

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Build Type</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS12 – Win 7</td>
<td></td>
<td>25 minutes?</td>
</tr>
<tr>
<td>VS12 – Win 10</td>
<td></td>
<td>10 minutes</td>
</tr>
<tr>
<td>VS12 – Win 10 /MP</td>
<td></td>
<td>3.5 minutes (100% cpu)</td>
</tr>
<tr>
<td>GCC 7.4</td>
<td></td>
<td>10.5 minutes</td>
</tr>
<tr>
<td>GCC 7.4 -O2</td>
<td></td>
<td>13 minutes <strong>error</strong></td>
</tr>
<tr>
<td>GCC 7.4 -O2 redux</td>
<td></td>
<td>12.7 minutes?</td>
</tr>
<tr>
<td>GCC 7.4 -O3</td>
<td></td>
<td>13.5 minutes</td>
</tr>
<tr>
<td>Java compile time (&lt;0&gt;)</td>
<td></td>
<td>25 seconds</td>
</tr>
</tbody>
</table>
Start-up times

Jvm warmup + load vs native load

Pre-warmup?
1. Warmup + intro
2. Loading till video
3. Start second intro
4. Play!
MULTITASKING AT ITS FINEST
CPU

- CPU threading bonanza
- Threading is hard
- Thread Safety is expensive (think null checks everywhere)
  - Usually safe/unsafe flavors
  - Atomicity
- The bla bla bla question
  - Do games need threads?
    - Amdahl vs Moore
    - Who knows...maybe we’ll get better at programming...again
It's interesting that many games can afford a constant 10x interpretation overhead for scripts, but not a spikey 1% for garbage collection.
Garbage Collection

✘ Is GC evil?
  ○ Well...you kind of already do it manually delete/free
  ○ Memory fragmentation vs virtual memory?
✘ Unity/Unreal/MK...etc
✘ Embrace the GC... Is It Time to Rewrite the Operating System in Rust?
✘ Concurrency is hard without GC.
✘ Manual memory management in the GC?
  ○ If you're gonna do it anyways.... Wouldn't you rather get moar bang for your buck?
Memory footprint (thus far)

SHAME. SHAME. SHAME.
Memory footprint (thus far)

❌ CPU overhead is negligible

❌ Memory overhead... is unavoidable could be better
  ○ Depends on the GC algorithm
Core language features (e.g. pointers)

Pointers
Macros
Operator overloading T_T
Type safety
Standard libs

Java vs STL
Intrinsics FTW!
Thread safety
Instrumentation API
Debuggability

Conditional bp C++(fast) /data bp
Perfmon/Valgrind
Instrumentation redux
WORA

Write Once...now Run Arsehole Anywhere

Caveat(s):
LWJGL = win + mac + linux
~Console

None...
~Although...
What else?

- Dev involvement in core language
  - JSR
  - JEP
- Once you go JVM, you never go...
- Untapped potential
  - Easy, but not simple problems (R. Hickey)
We have time for 1 Question...

Other questions may be directed at @blackbeard0x14e