Compiling C++ and C# Games to the Web

Alon Zakai
Researcher, Mozilla

Kevin Gadd

Quick Overview

- Demo several games ported to **HTML5**
- Discuss the **porting process**
- Talk about two **compilers** to JavaScript, for **C/C++** and **C#**

First Demo!

Why is this important?

The Web

Huge market: 100s of millions with HTML5 game-capable browsers, and growing

Games on the Web!

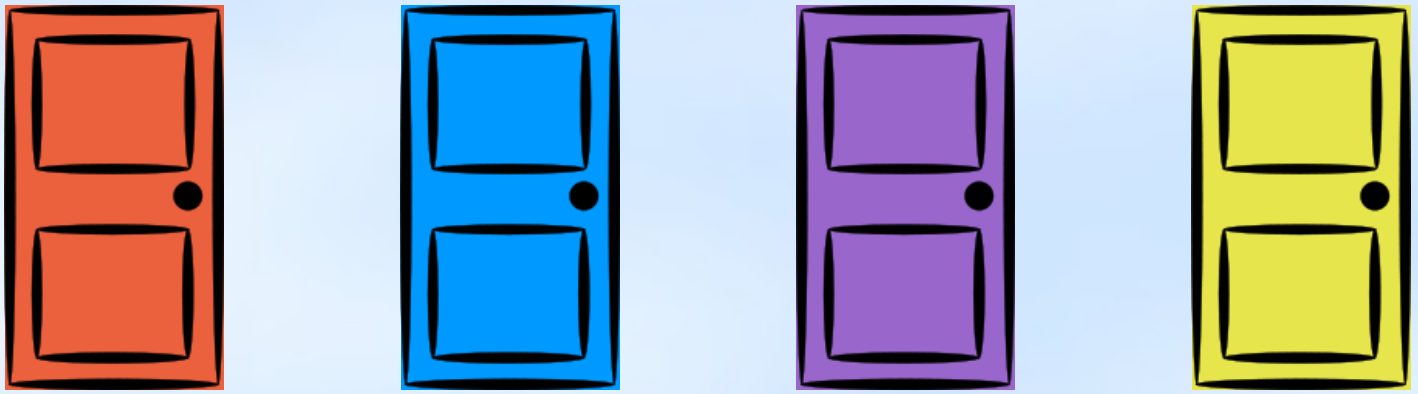
Access users with **minimal friction**, lower **customer acquisition costs**

enters actual game!

Games on the Web!

More options for reaching users

- Facebook, Kongregate, etc., with a fee
- Run your own website yourself



Browser Plugins

- **Flash**: 9% tax on fast 3D games
- **Unity**: Either Flash 9% tax, or no-cost plugin but limited reach
- **NaCl**: Chrome only, Chrome Store only, 30% tax

Browser Plugins

Browser plugins go **against the industry trend**

- **No** plugins in mobile versions of Safari, Chrome, Internet Explorer (IE)

But Wait!

Don't plugins give advantages too?

But Wait!

Don't plugins **fix browser API inconsistencies/limitations?**

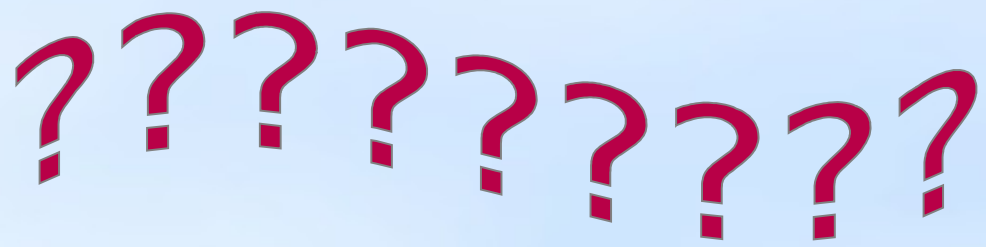
- **Audio** - WebAudio API almost standardized
- **Sockets** - WebRTC will provide raw UDP/TCP

But Wait!

Don't plugins let you **protect your code**?

- **No more and no less** than JavaScript can:

```
j=s[vh>>2]|0;f=rE(j)&7;s[c]=0;if(2>(f-1|0)>>>0))
{g=k;k+=28;h=g+12;i=g+24;gn(g,j);j=s[g>>2];m=s[g
+4>>2];n=g+8|0;p=h+8|0}
```



But Wait!

Don't plugins run **even in Internet Explorer?**

- **2D** is fine
- **3D** – **WebGL** - is indeed an issue in IE

But Wait!

Options for WebGL and Internet Explorer

- Use a **plugin** on IE (yuck)
- **Ignore** IE

But Wait!

Plugins let you write in **languages other than JavaScript**

- C++, C#, Java, ActionScript, etc.

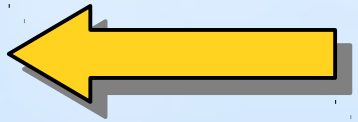
Compiling to JavaScript

The best of **both worlds**

- Use your **language and tools of choice**
- Generated JavaScript **runs in all modern browsers** without plugins

Compiling to JavaScript: Options

- Emscripten: **C**, **C++**
- JSIL: **C#**



We'll talk about these two

- Mandreel: **C**, **C++**, **Objective-C**
- GWT: **Java**

Porting C++ Games with Emscripten

Emscripten

- Compiles **C** and **C++** to JavaScript
  - Utilizes LLVM
- **Open source** and **free** to use
- **Stable and mature**, used to port many codebases

http://emscripten.org

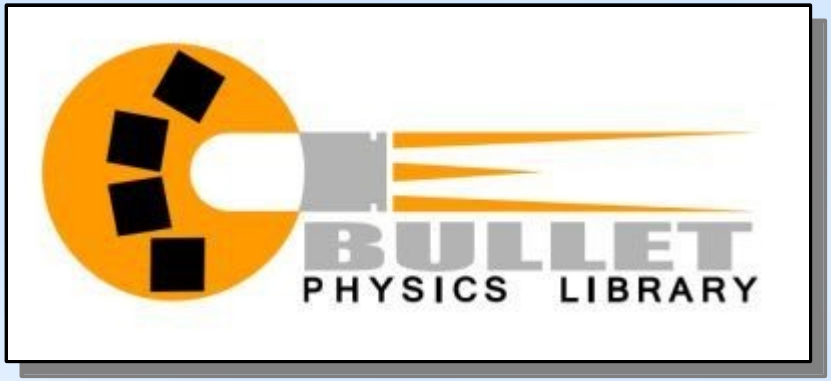
Emscripten – Ported Projects

- **Cube 2**
- **Heriswap**
- **SuperTux**
- **Me & My Shadow**
- **Ceferino**
- **Transport Tycoon Deluxe**
- Bullet
- Box2D
- Python
- Lua
- Ruby
- Poppler
- FreeType
- eSpeak (TTS)
- SQLite
- OpenJPEG
- zlib
- lzip (LZMA)
- libharu (PDF)
- etc.

Second Demo!

https://github.com/kripken/meandmyshadow.web

Third Demo!



https://github.com/kripken/ammo.js/

Porting that first person shooter

Emscripten: 3D FPS Example

BananaBread – Port of the Sauerbraten/Cube 2 game engine

Emscripten: 3D FPS Example

BananaBread – Port of the Sauerbraten/Cube 2 game engine

- **C++** compiled to **JavaScript**
- **OpenGL** compiled to **WebGL**
- **Full game**: Physics, AI, in-game editor, etc.
- **SDL audio** compiled to use **HTML Audio**

Emscripten: 3D FPS Example

BananaBread – Port of the Sauerbraten/Cube 2 game engine

- Startup uses up to **3 CPU cores**:
  - Uses **crunch** to decompress DXT images
  - Uses **zlib** to decompress levels
  - Uses **browser decoders** for PNGs, JPGs

Emscripten: 3D FPS Example

BananaBread – Port of the Sauerbraten/Cube 2 game engine

- 100% **open source** – free to learn from the code or use it in your own projects

https://github.com/kripken/BananaBread

Emscripten: Porting Process

emcc is a drop-in in replacement for gcc or clang

- In many cases can use your normal build system, just plug in emcc

```
emcc -O2 project.cpp -o project.html
```

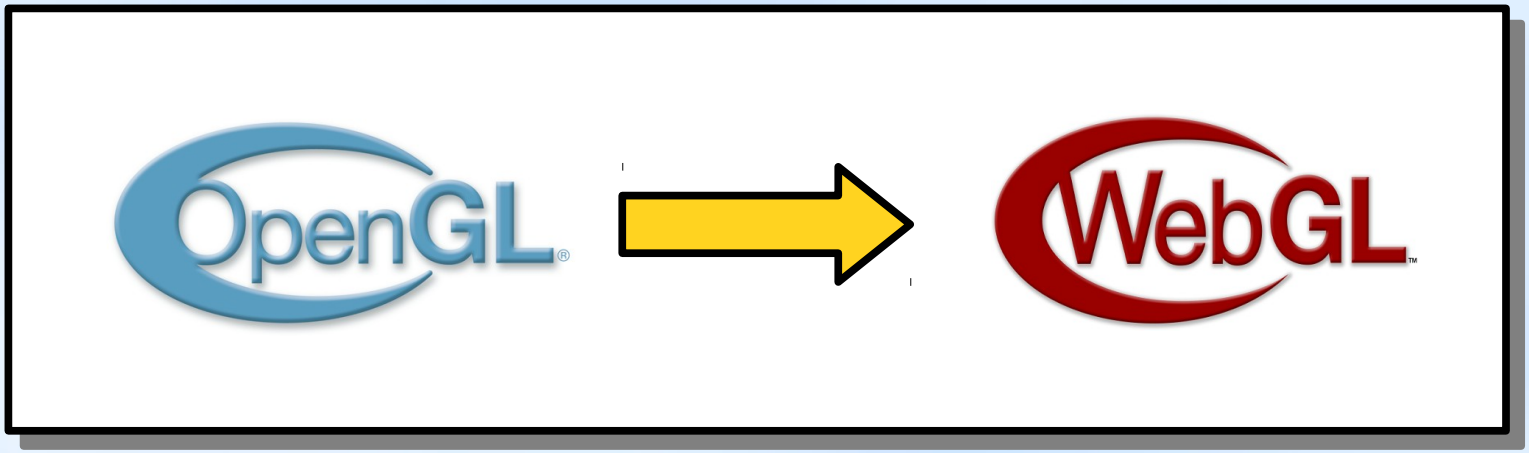
Emscripten: Features

Supports **familiar libraries** like libc, C++ std::, SDL, etc.

Emscripten: Features

Supports all OpenGL code that **maps directly to WebGL** (very close to GLES 2.0)

- And also some non-WebGL features too



Emscripten: Limitations

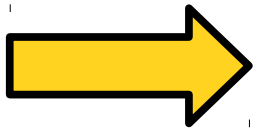
Supports **practically all C/C++ code**, except:

- **Nonportable** code (x86 asm, crazy stack tricks, etc.)

Emscripten: Limitations

No infinite loops on the web

```
while (1) {
  getInput();
  simulate();
  render();
  wait();
}
```



```
void frame() {
  getInput();
  simulate();
  render();
}
[..]
  addHandler(frame);
```

Emscripten: Limitations

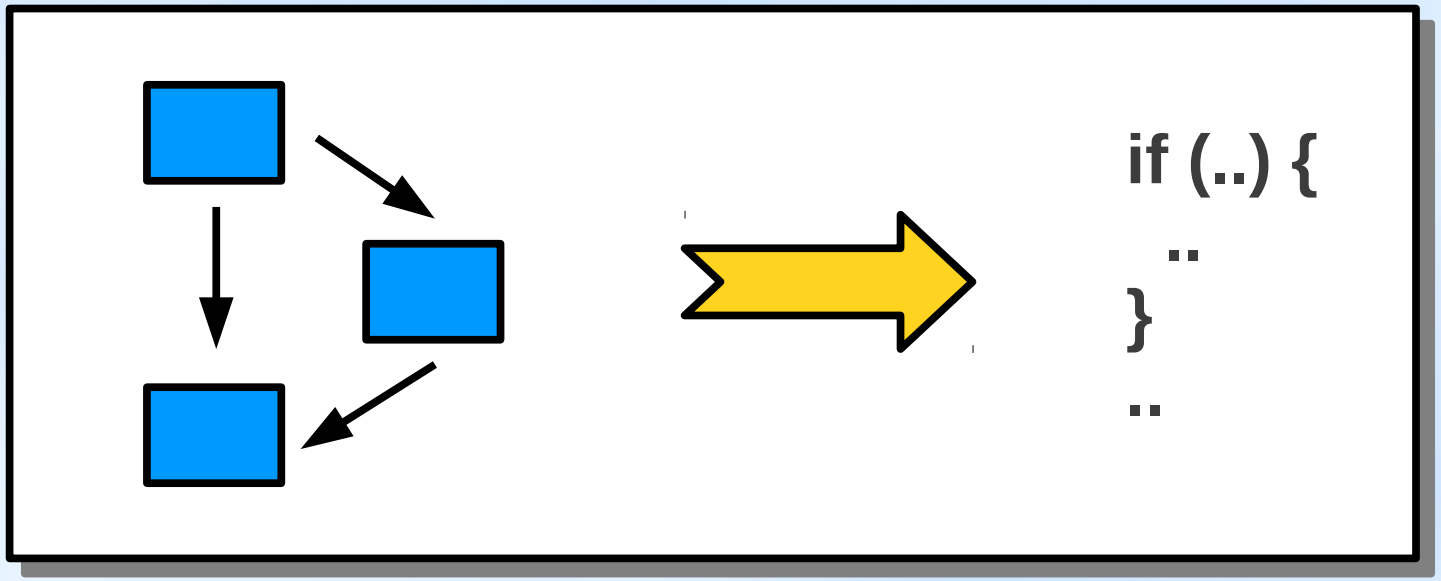
- 64-bit integer math
- No multithreading with shared state
- No Direct3D support, only OpenGL

Compiled C/C++ Performance

- Small benchmarks typically **1.5-6x slower** than natively compiled C/C++
    - Large codebases can hit problems with startup compilation
- Not quite native speed yet – but **improving fast**, and already ok even for 3D games!

Compiled C/C++ Performance

Relooper algorithm generates high-level native JS control flow from LLVM basic blocks



Still, how does JavaScript run a first person shooter...?

Compiled C/C++ Performance

Example code:

```
var x = func(y);
HEAP8[(x + 1)|0] = 10;
var z = (x+10)|0;
```

Compiled C/C++ Performance

Example code:

```
var x = func(y);
HEAP8[(x + 1)|0] = 10;
var z = (x+10)|0;
```

Force C-like integer behavior using |0 etc.

Compiled C/C++ Performance

Example code:

```
var x = func(y);
HEAP8[(x + 1)|0] = 10;
var z = (x+10)|0;
```

Typed array reads/writes easy to optimize

Compiled C/C++ Performance

Example code:

```
var x = func(y);
HEAP8[(x + 1)|0] = 10;
var z = (x+10)|0;
```

No **garbage collection** or **property accesses**

Compiled C/C++ Performance

Example code:

```
var x = func(y);
HEAP8[(x + 1)|0] = 10;
var z = (x+10)|0;
```

Not code you'd write by hand – but **good to compile to!**

Compiling C++ to the Web: Summary

- **Reuse** existing C/C++ code
- Results can be **surprisingly fast**
- Your game runs **on the web**

We've seen C++, now for C#!