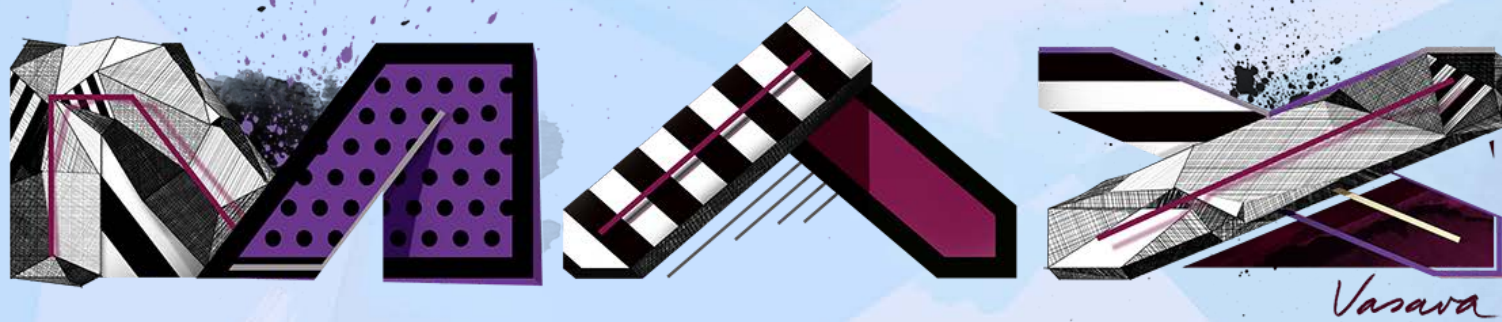




From C++ to Flash: The Power of the Adobe Flash C++ Compiler

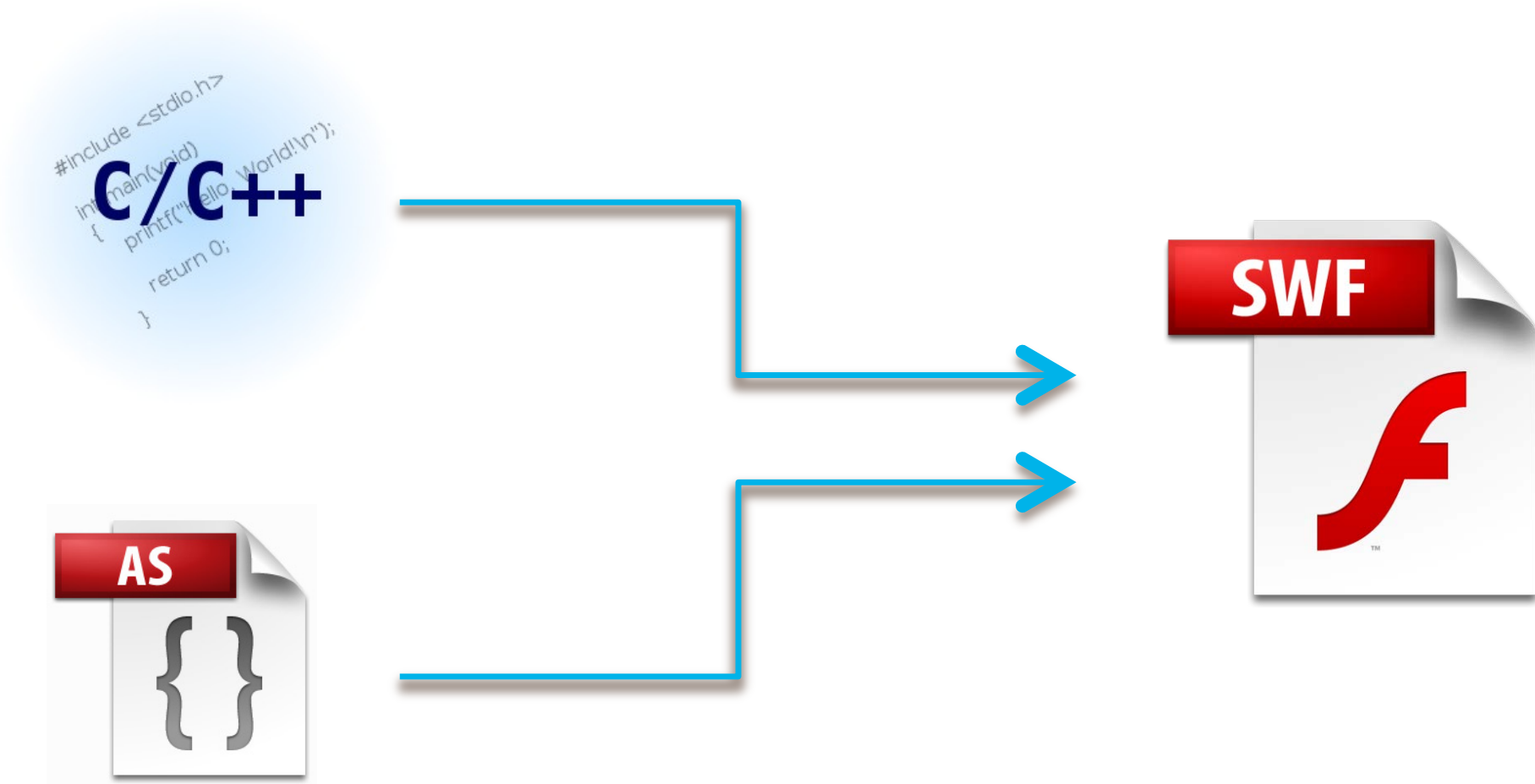
Alexander Macdonald | Sr. Computer Scientist



THE CREATIVITY CONFERENCE



Overview



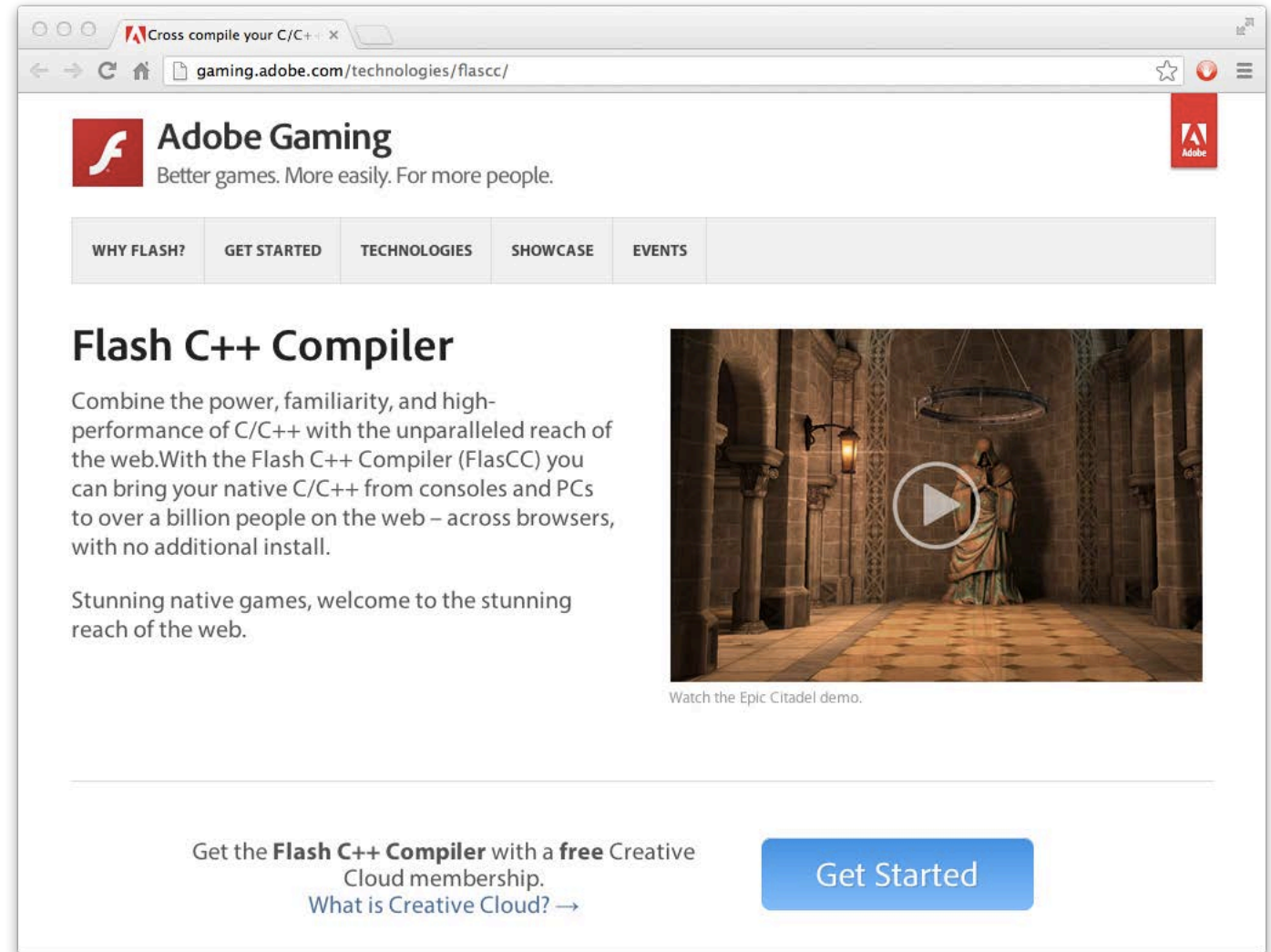
Why C/C++?

<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

| Position Apr 2013 | Position Apr 2012 | Delta in Position | Programming Language | Ratings Apr 2013 | Delta Apr 2012 | Status |
|----------------------|----------------------|-------------------|----------------------|---------------------|-------------------|--------|
| 1 | 1 | = | C | 17.862% | +0.31% | A |
| 2 | 2 | = | Java | 17.681% | +0.65% | A |
| 3 | 3 | = | C++ | 9.714% | +0.82% | A |
| 4 | 4 | = | Objective-C | 9.598% | +1.36% | A |
| 5 | 5 | = | C# | 6.150% | -1.20% | A |
| 6 | 6 | = | PHP | 5.428% | +0.14% | A |
| 7 | 7 | = | (Visual) Basic | 4.699% | -0.26% | A |
| 8 | 8 | = | Python | 4.442% | +0.78% | A |
| 9 | 10 | ↑ | Perl | 2.335% | -0.05% | A |
| 10 | 11 | ↑ | Ruby | 1.972% | +0.46% | A |
| 11 | 9 | ↓↓ | JavaScript | 1.509% | -1.37% | A |

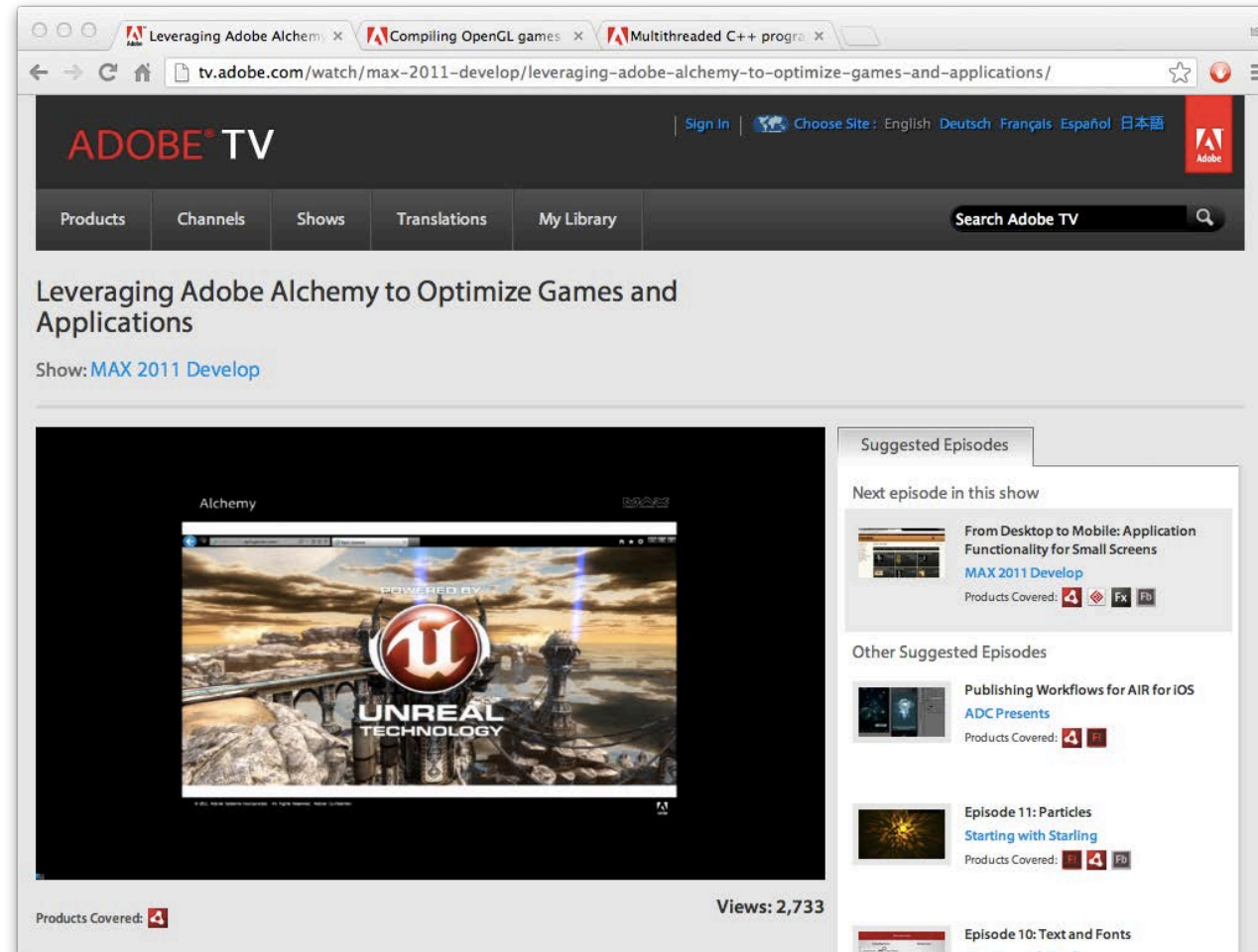
FlasCC

- gaming.adobe.com/technologies/flascc/
- Free to download
- No licensing restrictions on content



Overview

- <http://tv.adobe.com/watch/max-2011-develop/leveraging-adobe-alchemy-to-optimize-games-and-applications/>



Productivity

- Industry standard toolchain
 - GCC is the compiler
 - GDB is the debugger
 - Binutils for the utilities (ar, nm etc)
- Robust code generation
 - LLVM backend
 - FlasCC passes the GCC “torture test” unit tests
 - Handles large codebases like Unreal Engine 3
- Code reuse
 - SWIG auto-generates interop code
 - AS3 developers can use FlasCC SWCs like any other



Multithreading

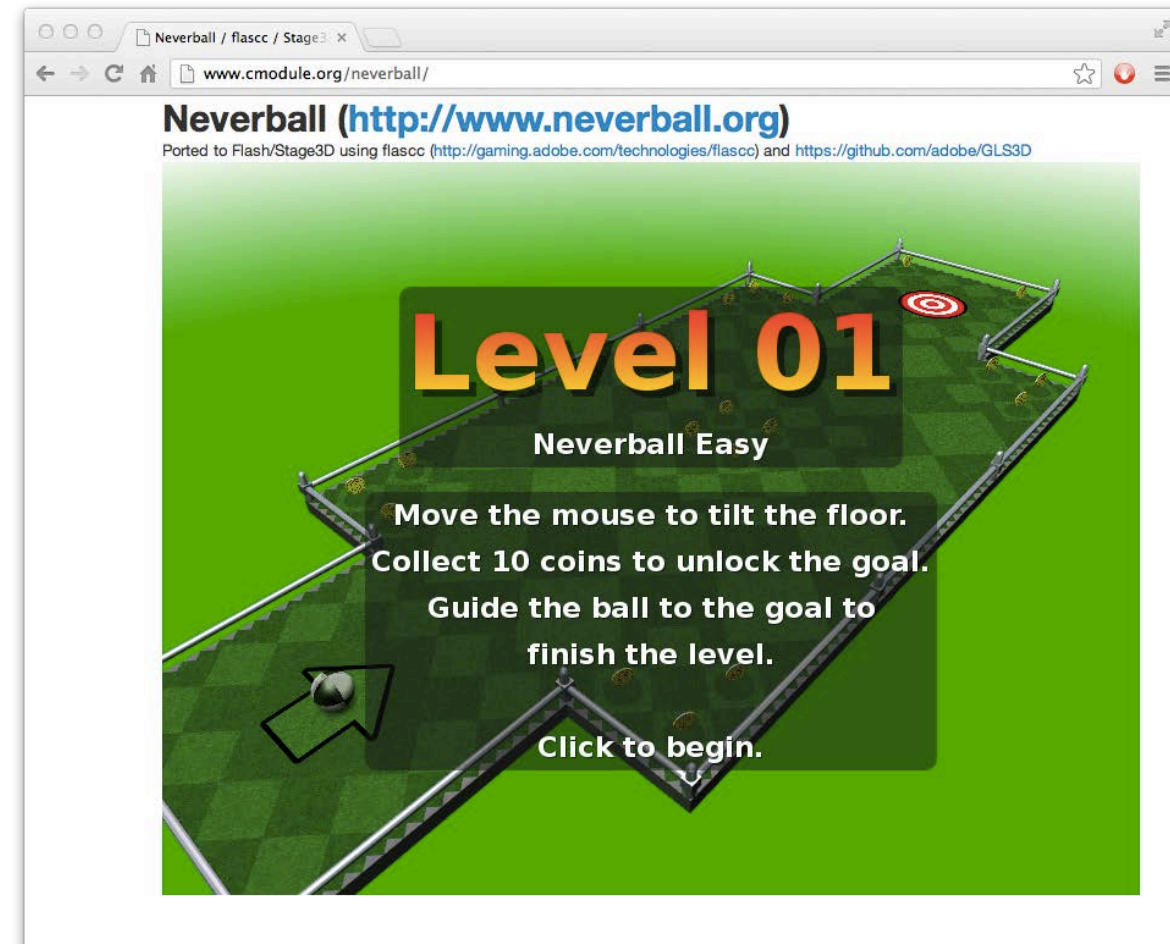
- Flash 11.5 supports multi-threading
 - Worker model (isolated execution environments)
 - AS3 communicates via message passing
 - domainMemory **is** shared across workers
- FlasCC exposes pthreads
 - No compromises, fully functional
 - Mutexes, conditions, rwlocks, semaphores
- Thunking mechanism
 - Similar to “performSelectorOnMainThread”
 - Run code on main (UI) worker
 - Manipulate AS3 objects on main worker

Threads vs Workers

- Workers
 - Each worker runs a whole SWF
 - First SWF to run is the main or “UI” worker → access to privileged APIs
 - All other workers are child workers → NO access to privileged APIs
- Pthreads
 - FlasCC creates workers running the main SWF
 - Each worker can run one pthread at a time
 - A pthread can be impersonated by a different worker
- DEMO

Graphics

- Flash 11 introduced Stage3D
 - GPU accelerated 3D
 - Equivalent to OpenGL ES 2
- FlasCC has experimental OpenGL/GLES support
 - <https://github.com/adobe/GLS3D>
- Gamedev.net “NeHe” OpenGL tutorials
 - <http://www.cmodule.org/nehe/>
 - <https://github.com/alexmac/alcfreeglut>
 - DEMO
- Neverball
 - <http://www.cmodule.org/neverball/>
 - DEMO



Graphics - shaders

- OpenGL → GLSL (source)
- Stage3D → AGAL (bytecode)
- GLSL To AGAL
 - <https://github.com/adobe/glsl2agal>
 - <http://adobe.github.io/glsl2agal/>
- DEMO

Porting libraries

- Two cases:
 - Self contained / middle-ware (e.g. libjpeg)
 - Bottom of the stack / hardware-abstraction (e.g. OpenGL)
- 1:
 - Should compile without modification
 - Need to specify the “target triple” for FlasCC: avm2-unknown-freebsd8
 - Compile to bitcode for performance (-O4) or ABC for quicker link times
- 2:
 - Same as 1, but code needs to be modified
 - Most libraries have some concept of a “backend”
 - Use the `__AVM2__` or `__FLASHPLAYER__` macro to add conditionally compiled code

Extra libraries

- <https://github.com/alexmac/alcextra>
 - SDL_image / SDL_ttf
 - Aalib
 - Eigen
 - Freetype
 - Giflib
 - Gmp
 - Iconv
 - Libogg, Libvorbis
 - Libwebp
 - Ncurses
 - Protobuf
- DEMO

Porting applications

- Pre-Loader
 - FlasCC default preloader can be replaced
 - Code for the default included in the SDK
- Console.as
 - The root sprite class that Flash instantiates.
 - Good place to perform input handling, stage3D setup
 - Register an enterFrame handler

main()

```
#include <stdio.h>
```

```
struct Foo {  
    Foo() { printf("Foo ctor\n"); }  
    ~Foo() { printf("Foo dtor\n"); }  
};
```

```
Foo x;
```

```
int main() {  
    printf("Hello World\n");  
}
```



Foo ctor
Hello World
Foo dtor


main() – AS3_GoAsync()

```
#include <stdio.h>
#include <AS3/AS3.h>
```

```
struct Foo {
    Foo() { printf("Foo ctor\n"); }
    ~Foo() { printf("Foo dtor\n"); }
};
```

```
Foo x;
```

```
int main() {
    printf("Hello World\n");
    AS3_GoAsync();
}
```



Foo ctor
Hello World

Dealing with run loops

```
while(gameRunning) {  
    handleInput();  
    renderFrame();  
}
```

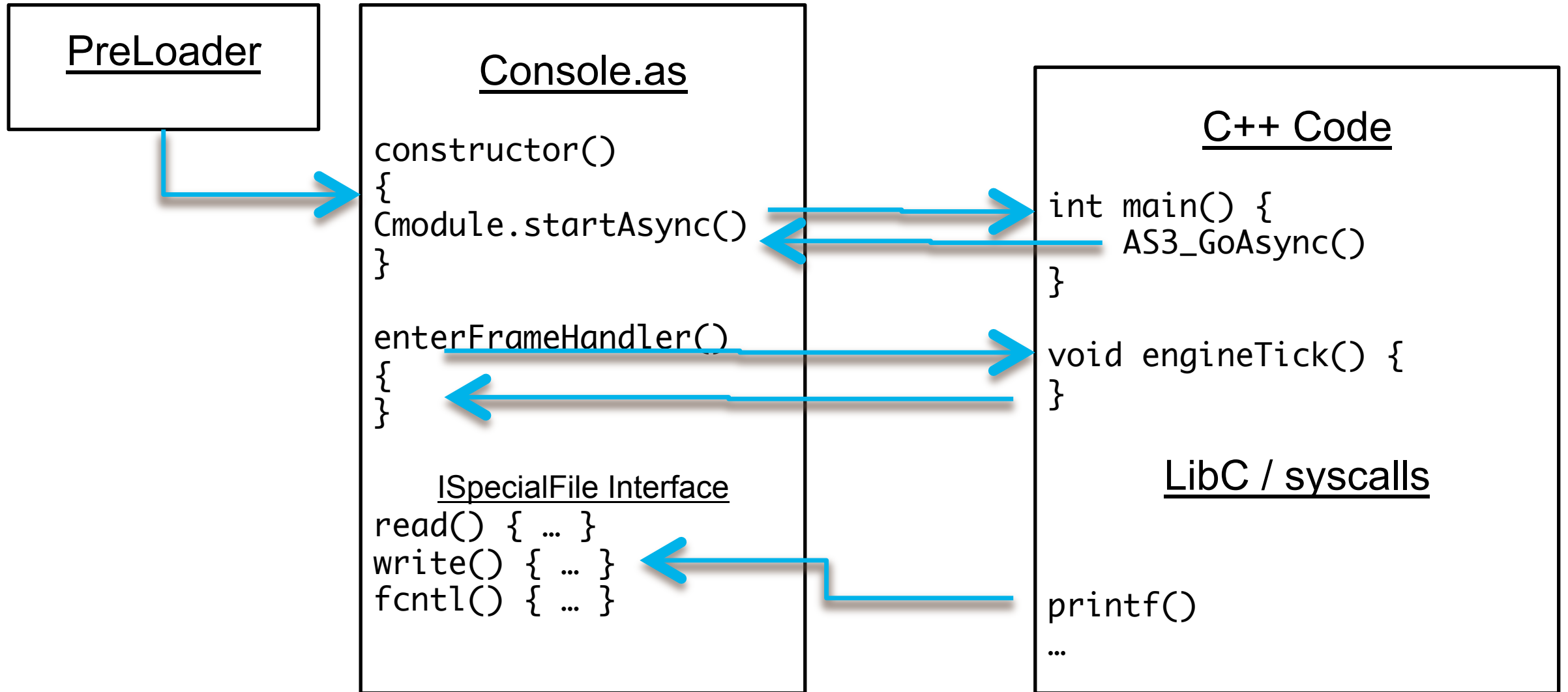
Dealing with run loops

```
while(flashPlayerRunning) {  
    handleInput();  
    //Execute AS3 enterFrame  
    {  
        while(gameRunning) { // This blocks outer loop!  
            handleInput();  
            renderFrame();  
        }  
    }  
    renderFrame();  
}
```

Dealing with run loops

```
while(flashPlayerRunning) {  
    handleInput();  
    //Execute AS3 enterFrame  
    {  
        if(gameRunning) {  
            handleInput();  
            renderFrame();  
        }  
    }  
    renderFrame();  
}
```

High level flow



Dealing with run loops

```
while(gameRunning) {  
    handleInput();  
    renderFrame();  
}
```


Dealing with run loops

```
while(flashPlayerRunning) {  
    handleInput();  
    //Execute AS3 enterFrame  
    {  
        CModule.serviceUIRequests()  
    }  
    renderFrame();  
}  
  
while(backgroundWorkerRunning) {  
    //Execute AS3  
    {  
        while(gameRunning) { // This blocks outer loop, but that's OK.  
            handleInput();  
            renderFrame();  
            avm2_ui_thunk(doStuffOnMainThread);  
        }  
    }  
}
```

High level flow - multithreaded

Console.as (UI Worker)

```
constructor() {  
  Cmodule.startBackground()  
}  
  
enterFrameHandler()  
{  
  Cmodule.serviceUIRequests()  
}
```

ISpecialFile Interface

```
read() { ... }  
write() { ... }  
fcntl() { ... }
```

C++ Code (Background)

```
int main() {  
  while(true) {  
    doStuff()  
    avm2_ui_thunk(doStuffOnMain)  
  }  
}  
  
void doStuffOnMain () {  
  ...  
}
```

LibC / syscalls (Background)

```
printf()  
...
```

High level flow - multithreaded

Console.as (UI Worker)

```
constructor() {  
    Cmodule.startBackground()  
}  
  
enterFrameHandler()  
{  
    Cmodule.serviceUIRequests()  
}
```

ISpecialFile Interface

```
read() { ... }  
write() { ... }  
fcntl() { ... }
```

C++ Code (Background)

```
int main() {  
    while(true) {  
        doStuff()  
        avm2_ui_thunk(doStuffOnMain)  
    }  
}  
  
void doStuffOnMain () {  
    ...  
}
```

LibC / syscalls (Background)

```
printf() ) { avm2_ui_thunk(...) }  
  
...
```

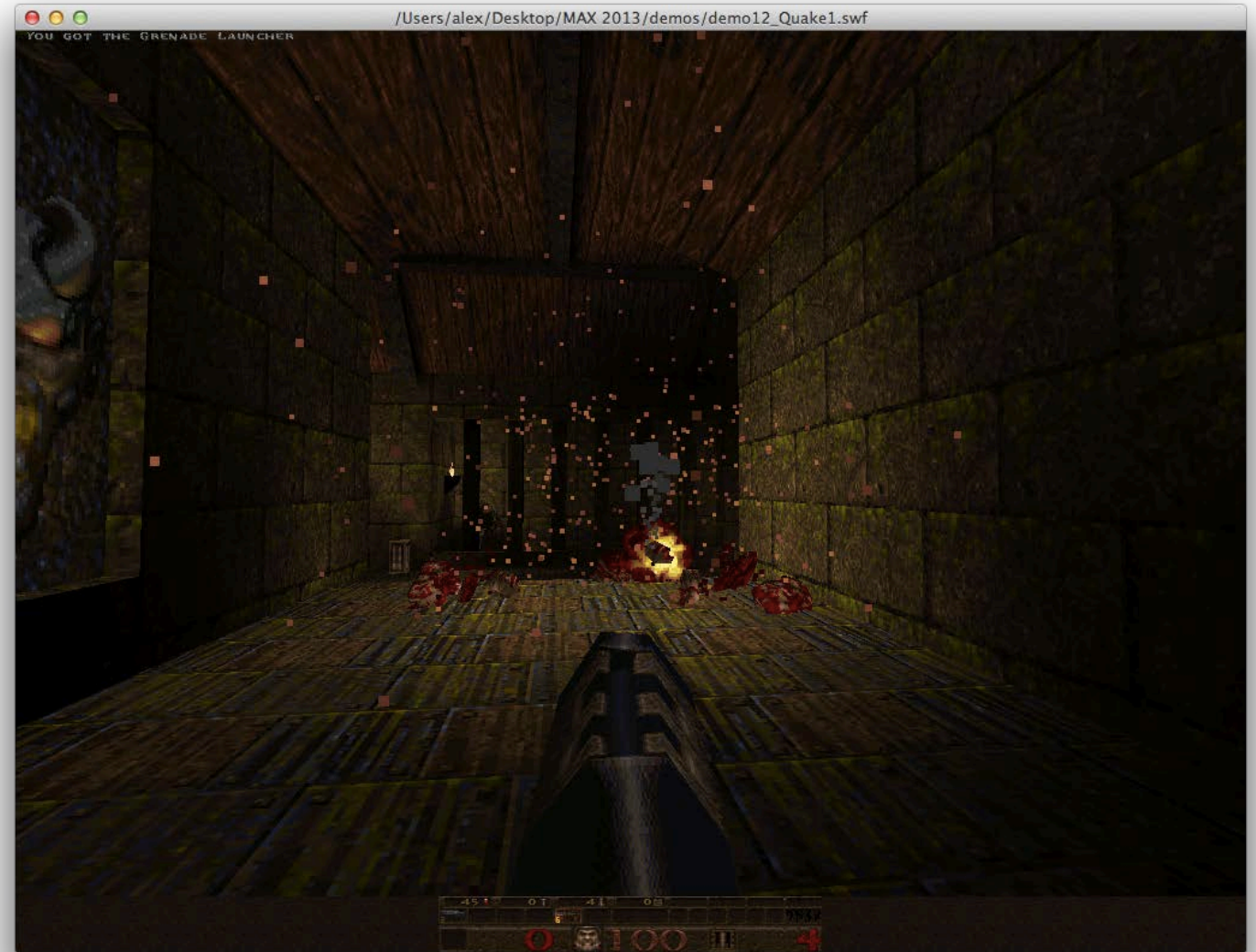


Console.as

```
public function Console()  
{  
    if(CModule.runningAsWorker()) {  
        return;  
    }  
  
    ...  
}
```

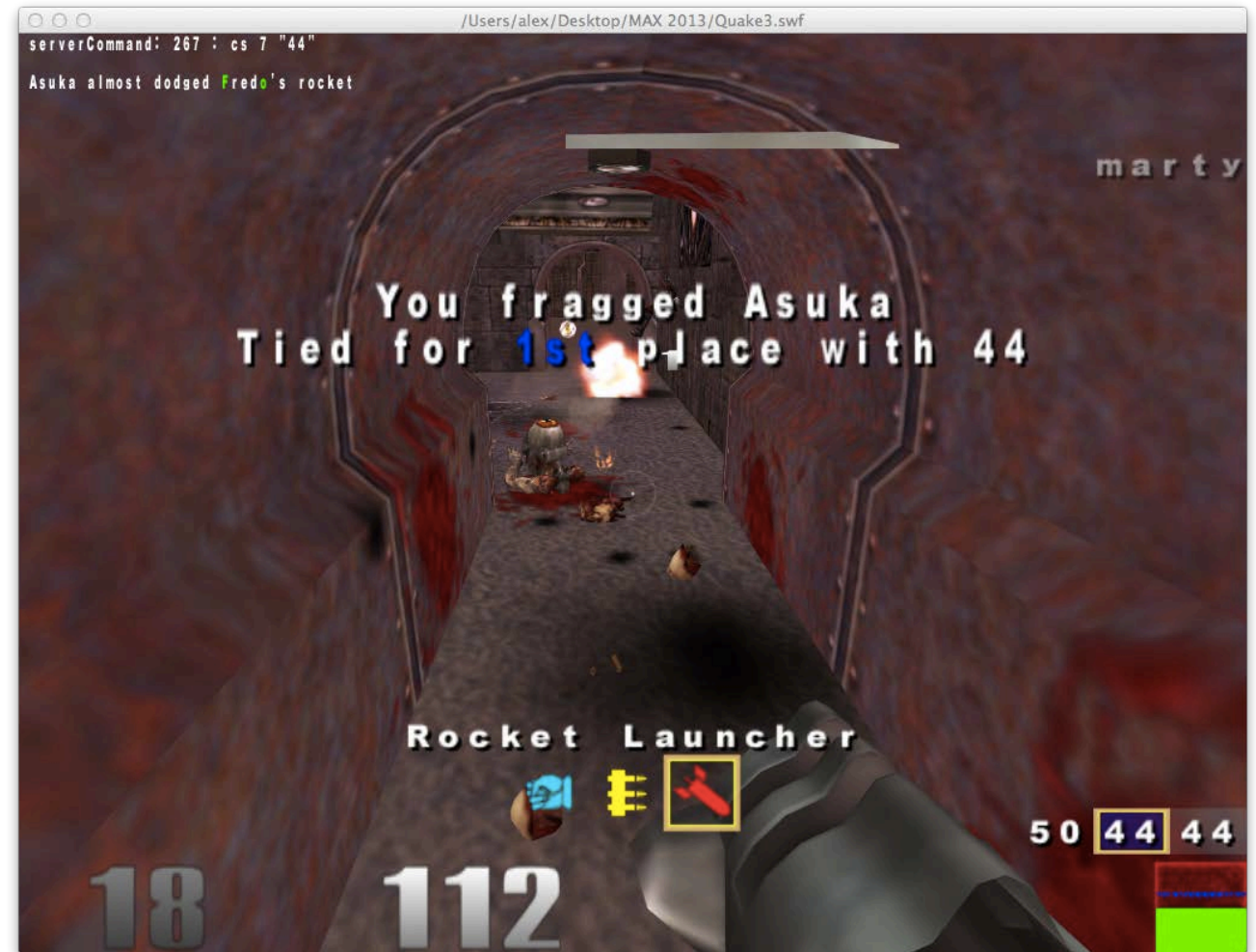
Example ports

- FlasCC SDK
 - Quake 1
 - Broken run loop for Flash < 11.5
 - Background thread for Flash >= 11.5
 - Bullet Physics
 - Box2D
 - Lua
- DEMO



Example ports

- <https://github.com/alexmac/alcexamples>
 - OpenCV
 - Quake3
 - Bochs
 - DosBox
 - NetHack
 - Neverball
- DEMO



Neverball diff

```
1 Binary files ./_DS_Store and /Users/alex/Desktop/neverball-1.5.4/_DS_Store differ
2 diff -uP -r ./gitignore /Users/alex/Desktop/neverball-1.5.4/.gitignore
3 --- ./gitignore 1989-12-31 16:00:00.000000000 -0800
4 +++ /Users/alex/Desktop/neverball-1.5.4/.gitignore 2009-09-20 03:06:00.000000000 -0700
5 @@ -0,0 +1,7 @@
6 +.odj
7 +.sol
8 +.desktop
9 +.locale
10 +/neverball
11 +/neverputt
12 +/mapo
13 Only in .: Console.as
14 Only in .: VFSPreLoader.as
15 diff -uP -r ./ball/main.c /Users/alex/Desktop/neverball-1.5.4/ball/main.c
16 --- ./ball/main.c 2012-11-07 02:45:53.000000000 -0800
17 +++ /Users/alex/Desktop/neverball-1.5.4/ball/main.c 2009-09-20 03:06:00.000000000 -0700
18 @@ -17,7 +17,6 @@
19 #include <SDL.h>
20 #include <stdio.h>
21 #include <string.h>
22 #include <AS3/AS3.h>
23
24 #include "glext.h"
25 #include "config.h"
26 @@ -377,12 +376,11 @@
27 fs_mkdir("Screenshots");
28
29
30 -static int t1, t0, uniform;
31 -
32 int main(int argc, char *argv[])
33 {
34     SDL_Joystick *joy = NULL;
35 -
36 + int t1, t0, uniform;
37 +
38     if (!fs_init(argv[0]))
39     {
40         fprintf(stderr, "Failure to initialize virtual file system: %s\n",
41 @@ -410,9 +408,6 @@
42 config_init();
43 config_load();
44
45 - /* unlock all levels by default */
46 - config_set_cheat();
47 -
48 - /* Initialize the joystick. */
49
50 if (config_get_d(CONFIG_JOYSTICK) && SDL_NumJosticks() > 0)
51 @@ -480,24 +445,6 @@
52 uniform = config_get_d(CONFIG_UNIFORM);
53 t0 = SDL_GetTicks();
54
55 #ifdef __AVX2__
56
57 Console.as will get a pointer to mainLoopTick() and
58 call that every frame instead.
59 - This way we effectively integrate the game loop with the
60 - FlashPlayer event loop (so we don't hang in here as main())
61 - is called from within the FlashPlayer event loop).
62 - */
63 -
64 - /*
65 - Here we throw an exception so that we can break the control
66 - out of main() without returning.
67 - The code in Console.as will take care of calling the
68 - factored out game loop code by calling mainLoopTick() periodically.
69 - */
70 - AS3_GoAsync();
71 #endif
72
73 while (loop())
74 {
75     t1 = SDL_GetTicks();
76     @@ -520,48 +497,5 @@
77     return 0;
78 }
79
80 #ifdef __AVX2__
81 -void mainLoopTick()
82 {
83     loop();
84     t1 = SDL_GetTicks();
85
86     if (uniform)
87     {
88         /* Step the game uniformly, as configured. */
89
90         int u;
91         for (u = 0; u < abs(uniform); ++u)
92         {
93             st_timer(DT);
94             t0 += (int) (DT * 1000);
95         }
96     }
97     else
98     {
99         /* Step the game state at least up to the current time. */
100
101         while (t1 > t0)
102         {
103             st_timer(DT);
104             t0 += (int) (DT * 1000);
105         }
106     }
107 }
108
109 /* Render. */
110
111 st_paint(0.001f * t0);
112 video_swap();
113
114 if (uniform < 0)
115     shot();
116
117 if (config_get_d(CONFIG_NICE))
118     SDL_Delay(1);
119 #endif
120
121 /*-----*/
122 diff -uP -r ./ball/st_title.c /Users/alex/Desktop/neverball-1.5.4/ball/st_title.c
123 --- ./ball/st_title.c 2012-11-07 02:41:53.000000000 -0800
124 +++ /Users/alex/Desktop/neverball-1.5.4/ball/st_title.c 2009-09-20 03:06:00.000000000 -0700
125 @@ -124,7 +124,7 @@
126 if (strcmp(queue, keyphrase) == 0)
127 {
128     config_set_cheat();
129     gui_set_label(play_id, sgettext("menu:Play"));
130     gui_set_label(play_id, sgettext("menu:Cheat"));
131     gui_pulse(play_id, 1.2f);
132 }
133 else if (config_cheat())
134     if ((kd = gui_varray[jd])
135     {
136         if (config_cheat())
137             play_id = gui_start(kd, sgettext("menu:Play"));
138             play_id = gui_start(kd, sgettext("menu:Cheat"));
139             GUI_MED, TITLE_PLAY, 1;
140         else
141             play_id = gui_start(kd, sgettext("menu:Play"));
142     }
143 }
144
145
146 GUI_MED, TITLE_PLAY, 1;
147
148 /*gui_state(kd, sgettext("menu:Replay"), GUI_MED, TITLE_DEMO, 0);*/
149 gui_state(kd, sgettext("menu:Replay"), GUI_MED, TITLE_DEMO, 0);
150 gui_state(kd, sgettext("menu:Help"), GUI_MED, TITLE_HELP, 0);
151 /*gui_state(kd, sgettext("menu:Options"), GUI_MED, TITLE_CONF, 0);*/
152 /*gui_state(kd, sgettext("menu:Exit"), GUI_MED, TITLE_EXIT, 0);*/
153 gui_state(kd, sgettext("menu:Options"), GUI_MED, TITLE_CONF, 0);
154 gui_state(kd, sgettext("menu:Exit"), GUI_MED, TITLE_EXIT, 0);
155 #endif
156
157 gui_filler(jd);
158 Only in ./dist: _DS_Store
159 Only in .: exports.txt
160 Only in ./macosx: _DS_Store
161 diff -uP -r ./putt/main.c /Users/alex/Desktop/neverball-1.5.4/putt/main.c
162 --- ./putt/main.c 2012-11-07 06:50:31.000000000 -0800
163 +++ /Users/alex/Desktop/neverball-1.5.4/putt/main.c 2009-09-20 03:06:00.000000000 -0700
164 @@ -20,7 +20,6 @@
165 #include <stdlib.h>
166 #include <string.h>
167 #include <locale.h>
168 #include <AS3/AS3.h>
169
170 #include "glext.h"
171 #include "audio.h"
172 @@ -180,8 +189,6 @@
173 return d;
174 }
175
176 -static int t1, t0, uniform;
177 -
178 int main(int argc, char *argv[])
179 {
180     int camera = 0;
181     @@ -245,24 +242,6 @@
182
183 init_state(&st_null);
184 goto_state(&st_title);
185
186 #ifdef __AVX2__
187
188 Console.as will get a pointer to mainLoopTick() and
189 call that every frame instead.
190 - This way we effectively integrate the game loop with the
191 - FlashPlayer event loop (so we don't hang in here as main())
192 - is called from within the FlashPlayer event loop).
193 - */
194 -
195 - /*
196 - Here we throw an exception so that we can break the control
197 - out of main() without returning.
198 - The code in Console.as will take care of calling the
199 - factored out game loop code by calling mainLoopTick() periodically.
200 - */
201 - AS3_GoAsync();
202 #endif
203
204 while (loop())
205 {
206     if ((t1 = SDL_GetTicks()) > t0)
207     @@ -290,23 +269,5 @@
208     return 0;
209 }
210
211 #ifdef __AVX2__
212 -void mainLoopTick()
213 {
214     loop();
215
216     if ((t1 = SDL_GetTicks()) > t0)
217     {
218         st_timer((t1 - t0) / 1000.f);
219         st_paint(0.001f * t1);
220     }
221 }
222
223 SDL_GL_SwapBuffers();
224
225 t0 = t1;
226
227 if (config_get_d(CONFIG_NICE))
228     SDL_Delay(1);
229 #endif
230
231 diff -uP -r ./share/audio.c /Users/alex/Desktop/neverball-1.5.4/share/audio.c
232 --- ./share/audio.c 2012-09-16 16:29:38.000000000 -0700
233 +++ /Users/alex/Desktop/neverball-1.5.4/share/audio.c 2009-09-20 03:06:00.000000000 -0700
234 @@ -13,7 +13,6 @@
235 #include <SDL.h>
236 #include <AS3/AS3.h>
237 #include <vorbis/codes.h>
238 #include <vorbis/vorbisfile.h>
239 @@ -194,21 +193,11 @@
240
241
242
243
244
245 -static const int audioBufferLength = 16384;
246 -uint8 *audioBuffer = NULL;
247
248 -void audio_step(void *data, uint8 *stream, int length)
249 -static void audio_step(void *data, uint8 *stream, int length)
250 {
251     uint8 *stream = audioBuffer;
252     int length = audioBufferLength;
253     struct voice *v = voices;
254     struct voice *p = NULL;
255
256     if (audioBuffer == NULL)
257     {
258         stream = audioBuffer = malloc(audioBufferLength);
259     }
260
261     /* Zero the output buffer. */
262     memset(stream, 0, length);
263     @@ -278,16 +267,12 @@
264
265 {
266     /* Start the audio thread. */
267
268     #ifdef __AVX2__
269     audio_state = 1;
270     #else
271     if (SDL_OpenAudio(&spec, NULL) == 0)
272     {
273         audio_state = 1;
274         SDL_PauseAudio(0);
275     }
276     else fprintf(stderr, "ts\n", SDL_GetError());
277 #endif
278 }
279
280 /* Set the initial volumes. */
281 diff -uP -r ./share/config.c /Users/alex/Desktop/neverball-1.5.4/share/config.c
282 --- ./share/config.c 2012-11-07 03:40:22.000000000 -0800
283 +++ /Users/alex/Desktop/neverball-1.5.4/share/config.c 2009-09-20 03:06:00.000000000 -0700
284 @@ -184,7 +184,7 @@
285 { &CONFIG_KEY_RESTART, "key_restart", SDLK_r },
286 { &CONFIG_KEY_SCORE_NEXT, "key_score_next", SDLK_TAB },
287 { &CONFIG_KEY_ROTATE_FAST, "key_rotate_fast", SDLK_LSHIFT },
288 { &CONFIG_CHEAT, "play", 0 },
289 + { &CONFIG_CHEAT, "cheat", 0 },
290 { &CONFIG_STATS, "stats", 0 },
291 { &CONFIG_UNIFORM, "uniform", 0 },
292 }
```

The Virtual file system

- Unix style FS rooted at “/”
- Implement the IBackingStore interface
- Subclass InMemoryBackingStore
 - HTTPBackingStore
 - LSOBackingStore
 - ZipBackingStore (see Neverball example)
- Add the backing store:
 - `CModule.vfs.addBackingStore(yourBackingStore, path)`
 - E.g.
 - “/data” → HTTPbackingStore
 - “/user” → LSOBackingStore

Flash++

- AS3 APIs reflected into C++ APIs
 - AS3 namespace → C++ namespace
 - AS3 property → C++ members
 - AS3 method → C++ method
- Works for arbitrary ABCs
 - AS3WIG.jar is the tool
 - Sample 12 uses it for AGALAssembler.abc

Flash++

```
#include <vector>
#include <AS3/AS3.h>
#include <Flash++.h>

using namespace AS3::ui;

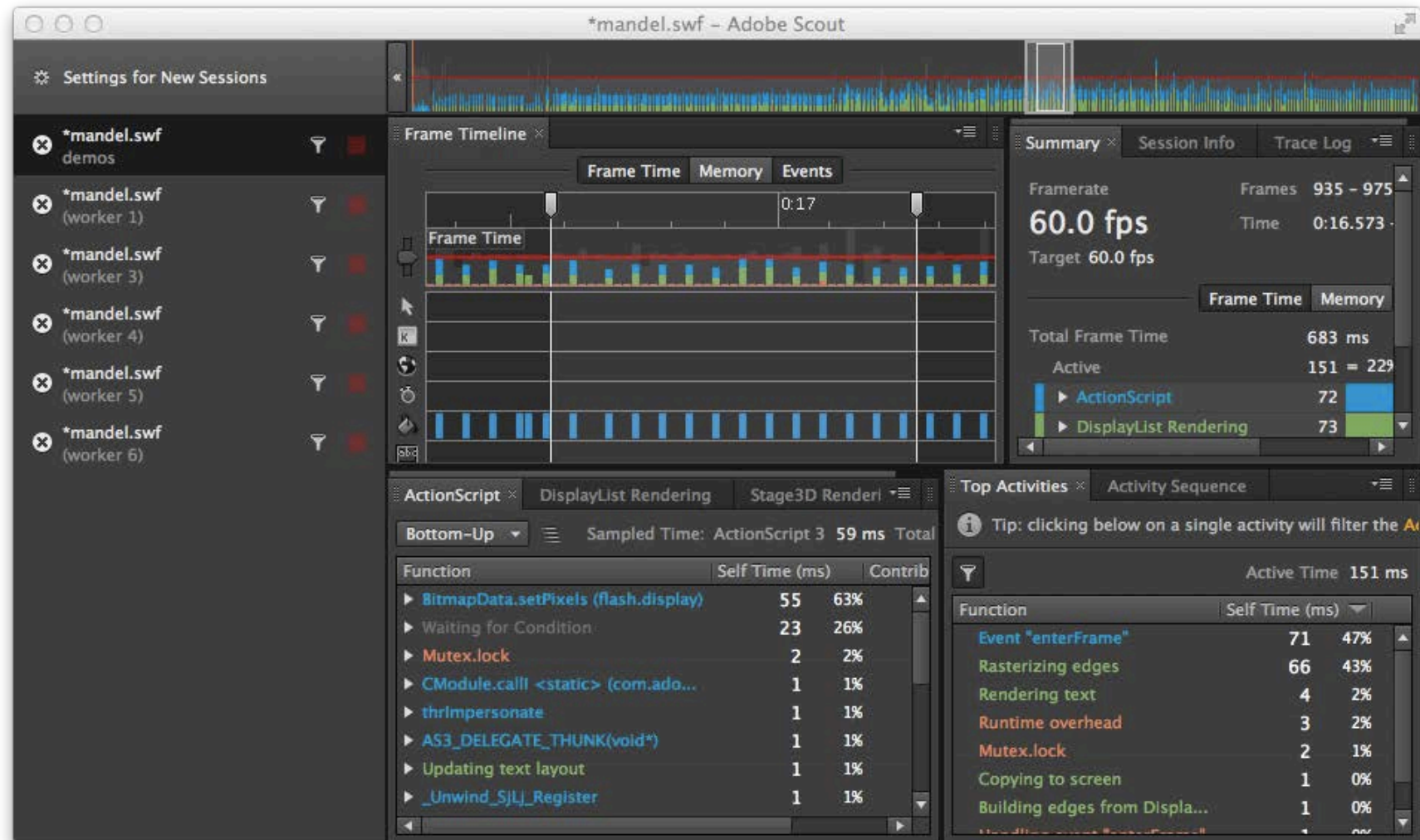
int main()
{
    flash::display::Stage stage = internal::get_Stage();

    flash::display::Sprite mySprite = flash::display::Sprite::_new();
    flash::display::Graphics graphics = mySprite->graphics;
    graphics->beginFill(0xff00ff, 0.5);
    graphics->drawCircle(0.0, 0.0, 30.0);
    graphics->endFill();

    stage->addChild(mySprite);
}
```

Profiling with Scout

- <http://gaming.adobe.com/technologies/scout/>

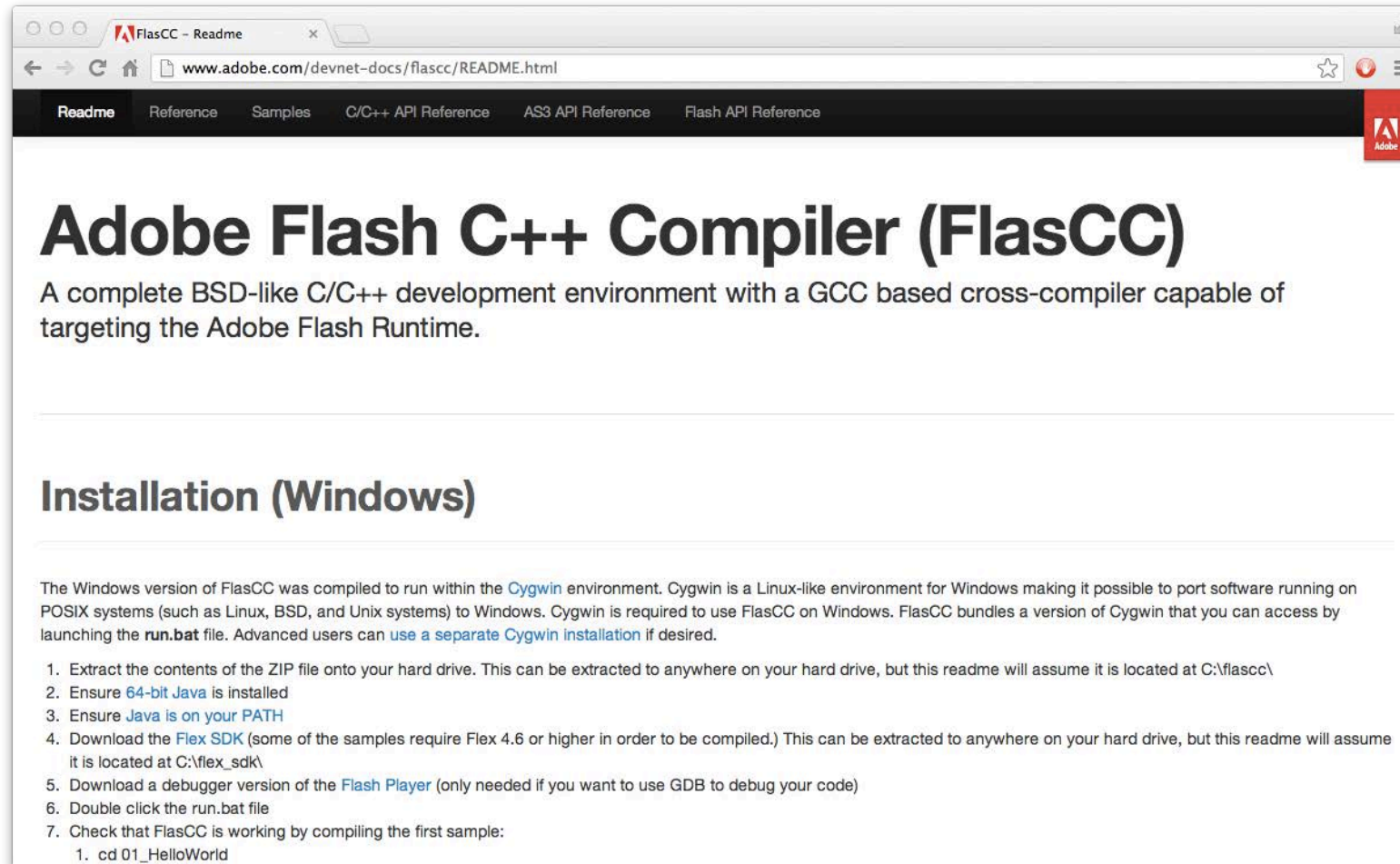


Profiling with Scout

- Demangles C/C++ function names
- Tracks domainMemory usage separately from other ByteArrays
- Experimental support for Threading
- Custom metrics can be generated from C/C++ code

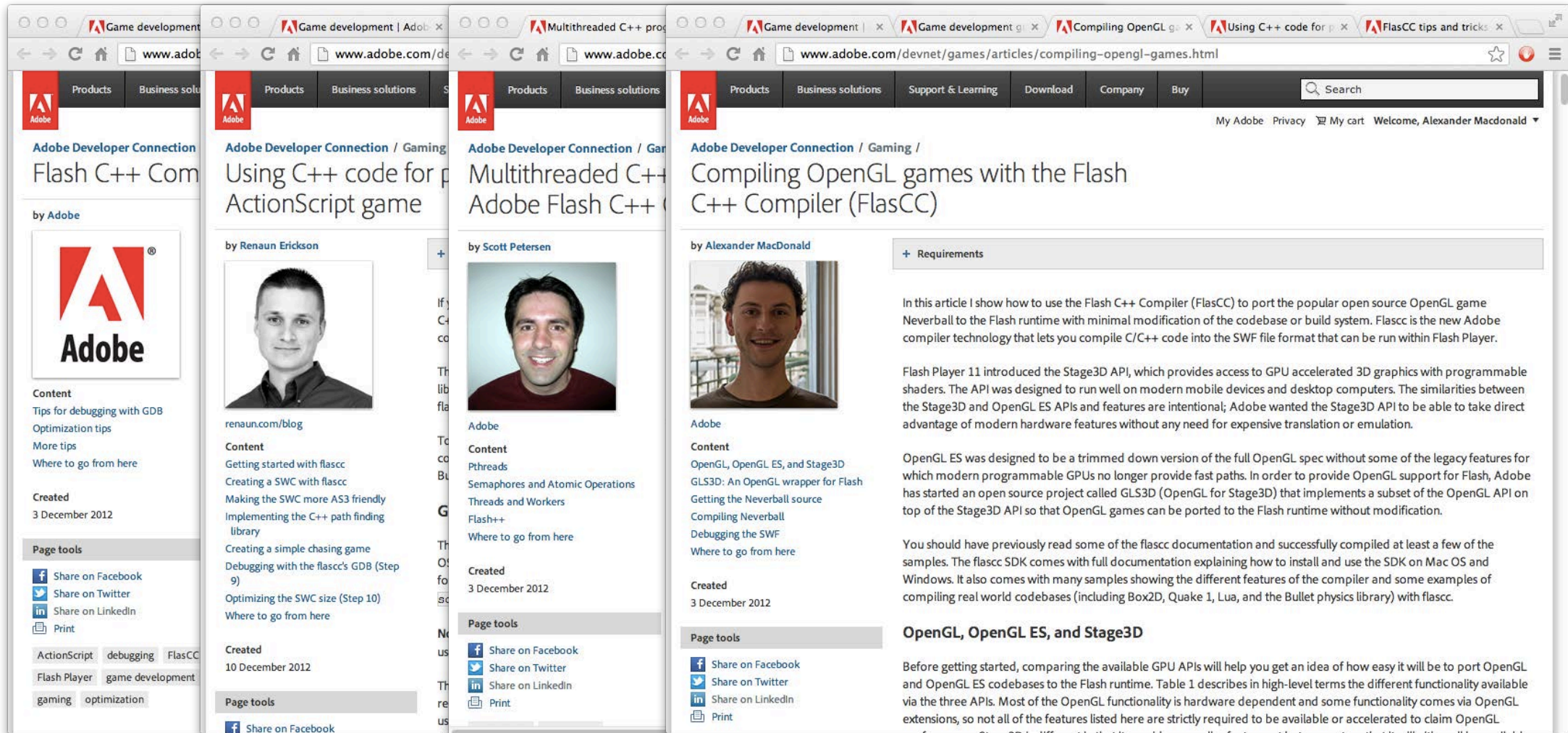
- DEMO

- <http://www.adobe.com/devnet-docs/flascc/README.html>



Adobe Devnet articles

- <http://www.adobe.com/devnet/games/game-development.html#gameflascc>



Future

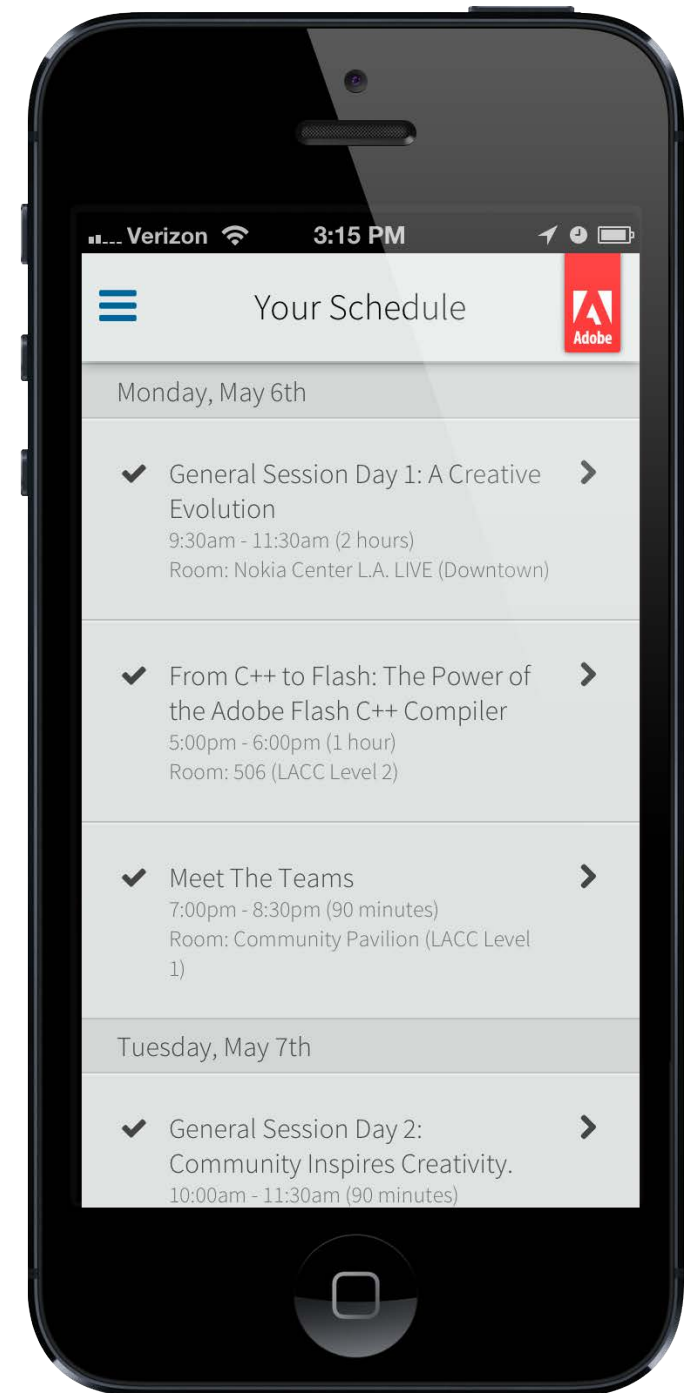
- Whole toolchain being open sourced
- Adobe will continue to support FlasCC
- Alchemy → FlasCC → CrossBridge



Adobe

alexmac@adobe.com

@dralexmac





Adobe