Google
Developer
Day2009

Native Client: Accelerating Web Applications

Henry Bridge
2009 Jun 05

Google Developer Day 2009

Why Native Code?

- Close the gap between desktop and web apps...
  - Performance
  - Choice of programming language
  - Leverage legacy code
- … but do not sacrifice
  - Portability
  - Safety

What we mean by “Performance”

- Key performance features include
  - POSIX-like thread support
  - Straightforward access to vector instructions
  - Hand-coded assembler

Provide performance features as desktop applications

What does it mean for the Web?

- Desktop CPU performance will enable Web apps with:
  - Safer multimedia codecs
  - Real-time audio and video synthesis
  - Real-time physics simulations
  - Local audio/video analysis and recognition
  - Multimedia editors
  - Flexible, high-throughput cryptography
  - Application-specific data compression
- Together with O3D we will enable:
  - High quality games
  - CAD applications

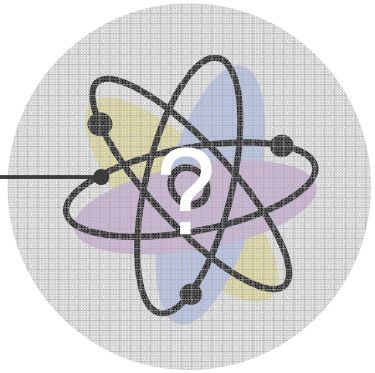
The Life of a NaCl-Enabled Web App

The Life of a NaCl-Enabled Web App

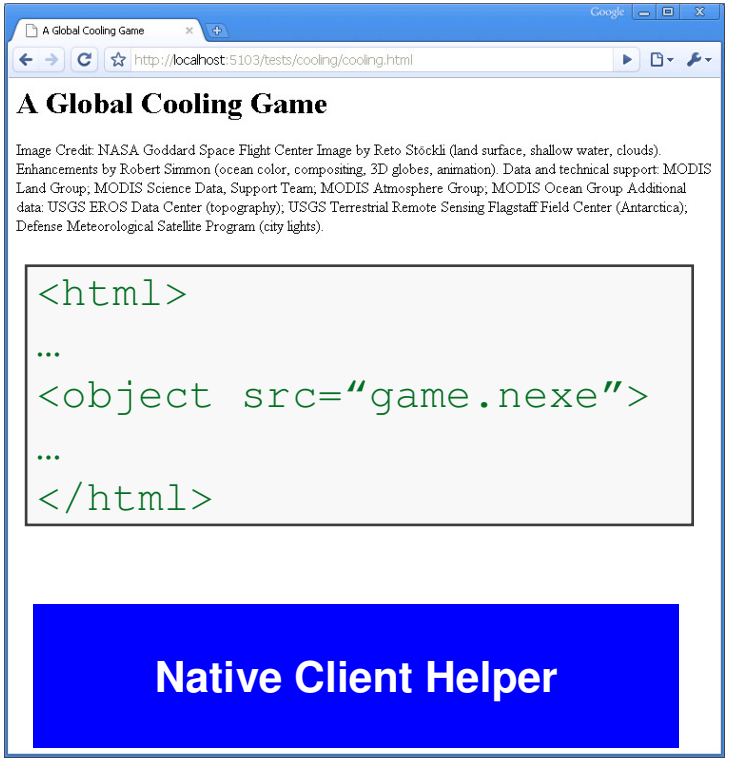
The Life of a NaCl-Enabled Web App

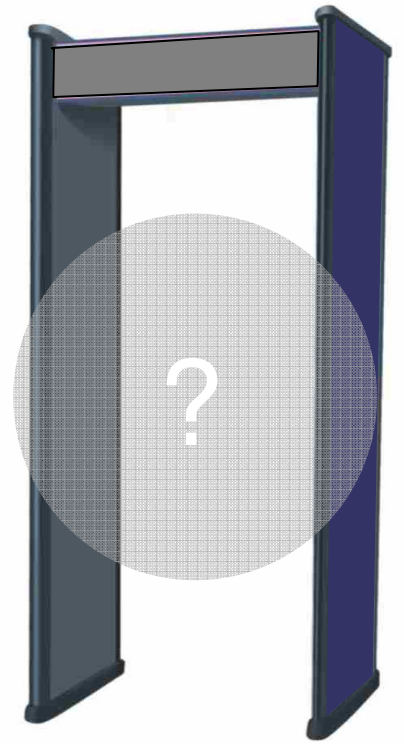
The Life of a NaCl-Enabled Web App

The Life of a NaCl-Enabled Web App

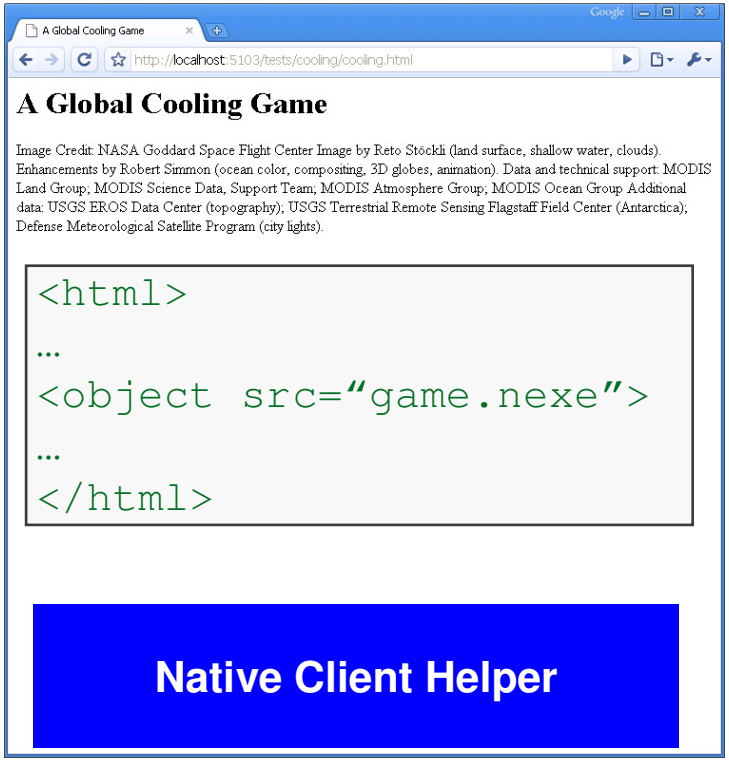


The Life of a NaCl-Enabled Web App

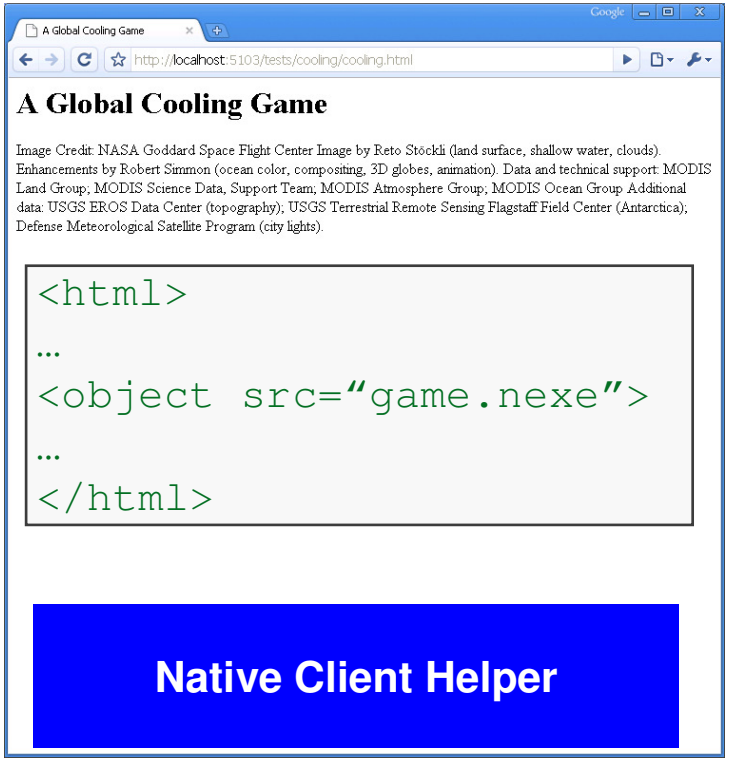


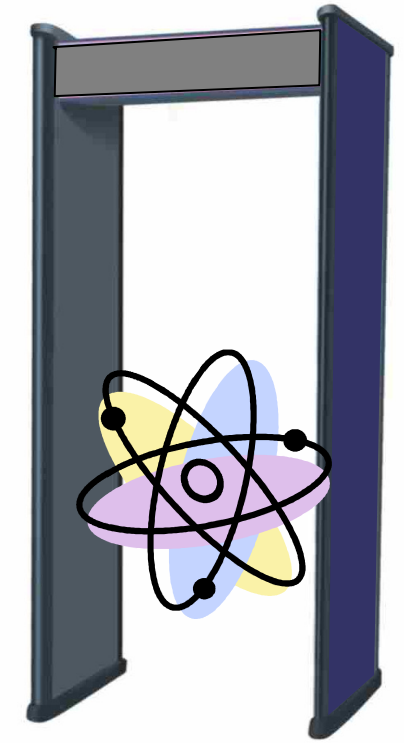


The Life of a NaCl-Enabled Web App



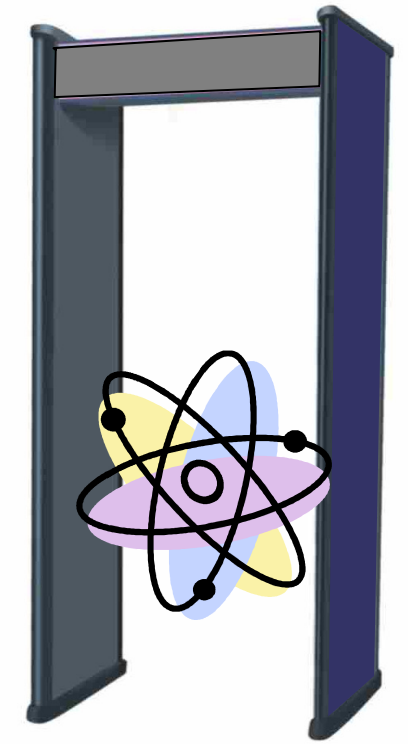
The Life of a NaCl-Enabled Web App



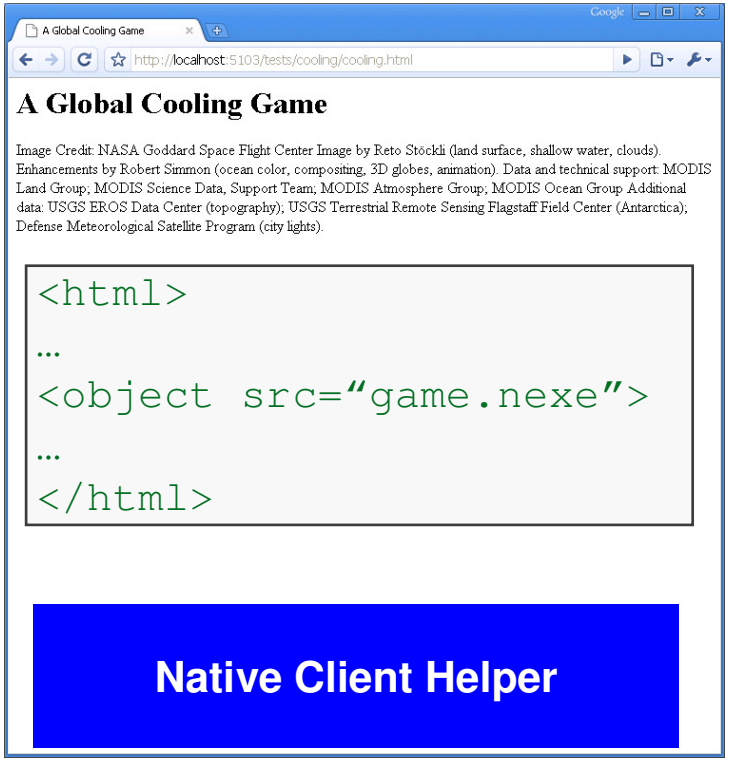


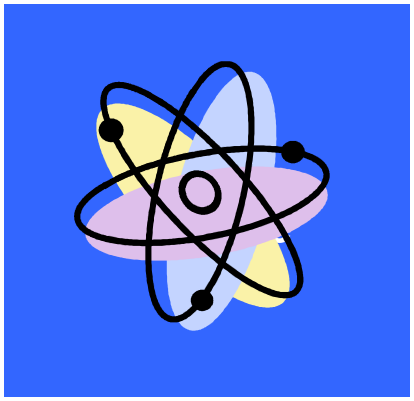
The Life of a NaCl-Enabled Web App

NaCl Runtime

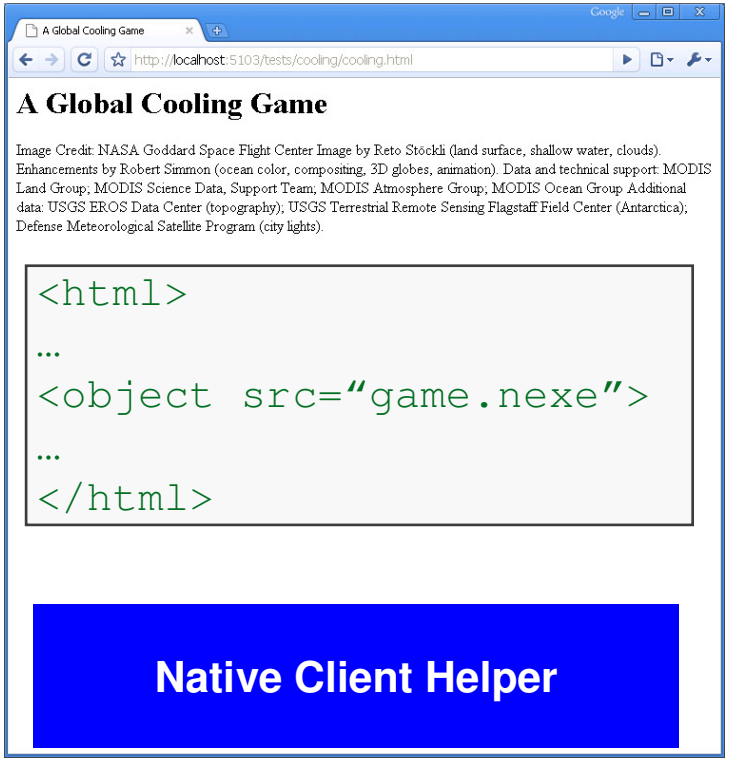


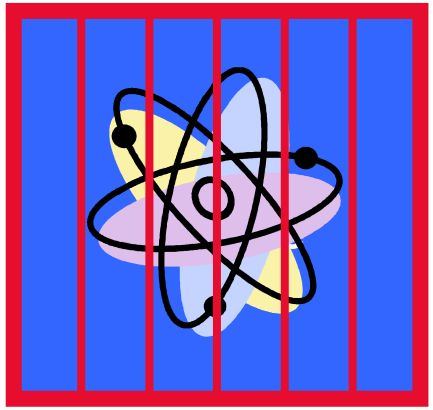
The Life of a NaCl-Enabled Web App



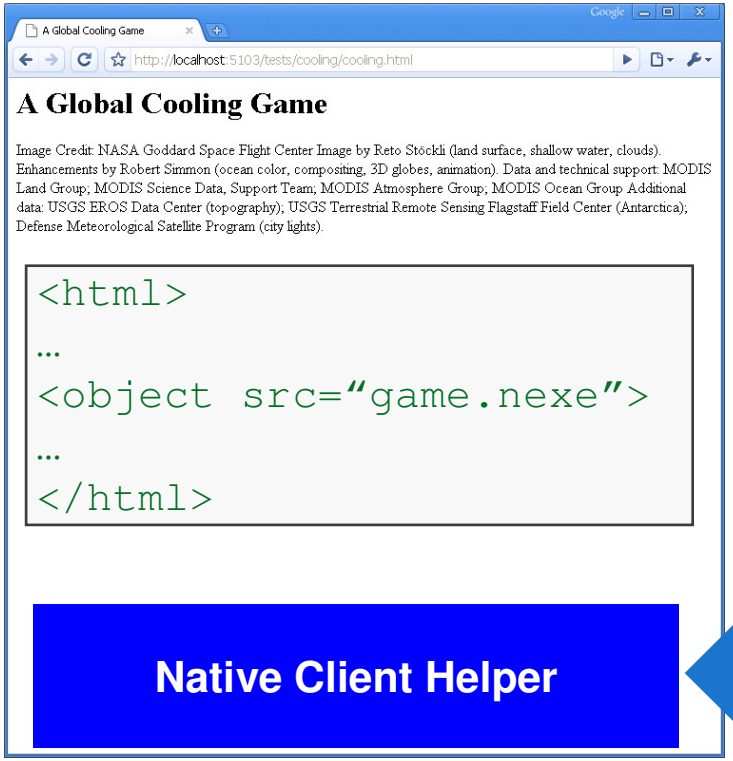


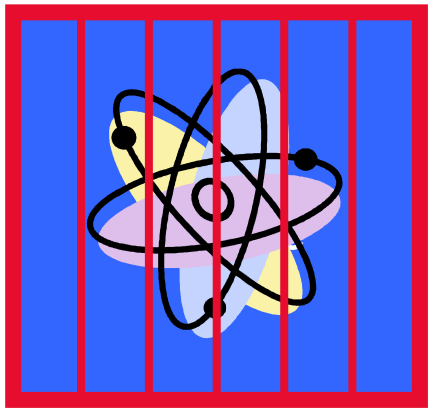
The Life of a NaCl-Enabled Web App



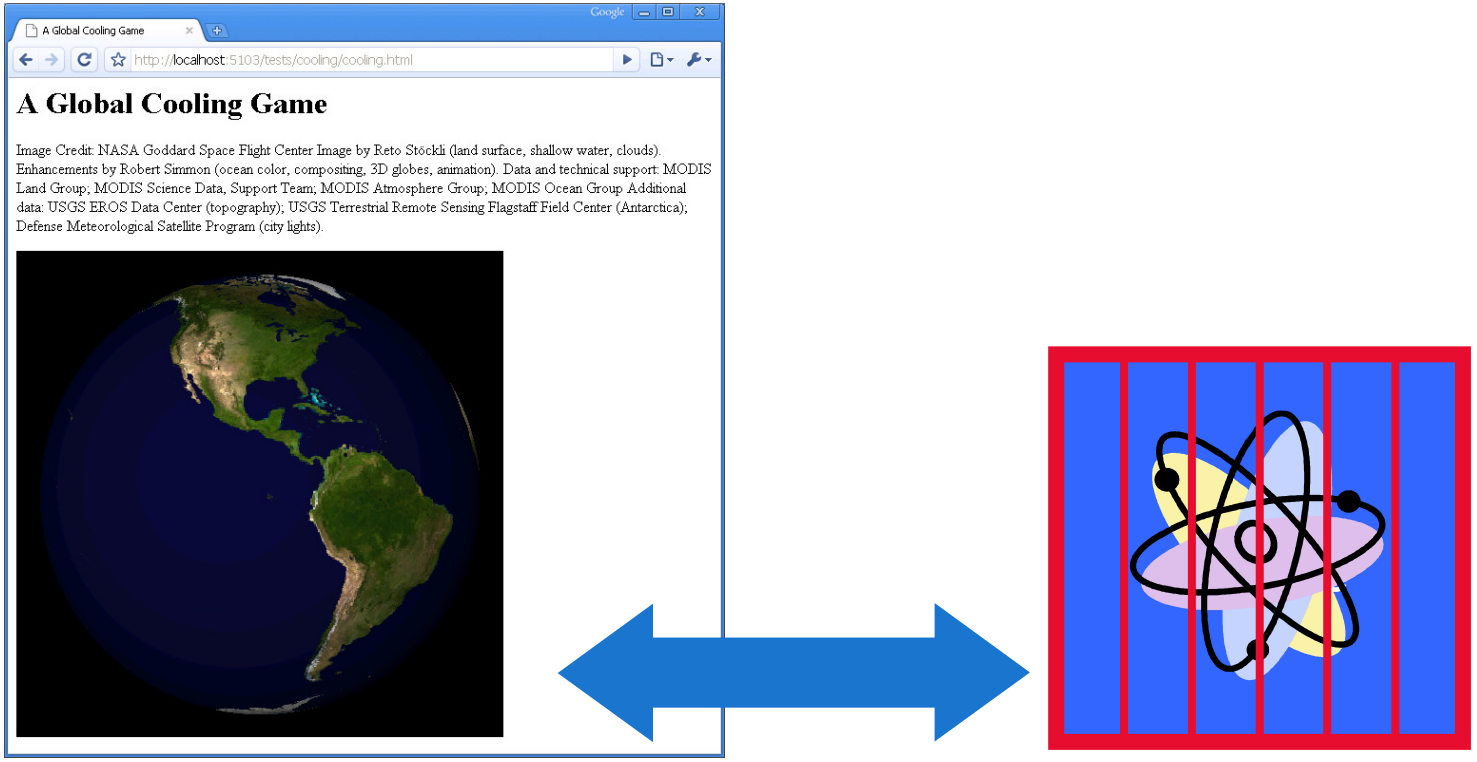


The Life of a NaCl-Enabled Web App





The Life of a NaCl-Enabled Web App



Native Client Security

Native Client Security

- Our goal: make native code at least as safe as JavaScript.
- Steps we've taken include:
- Multiple internal security reviews
- Open sourced our system; encouraged critical public review
- Published a peer reviewed technical paper in the *IEEE Symposium on Security and Privacy*
  - See **http://oakland09.cs.virginia.edu**
- Held an Security Contest

Native Client Security Contest

- 25 February to 5 May 2009
- Over 400 teams and 600 individuals participated
- 22 valid issues submitted
- Profile of valid issues:
  - Inner sandbox (1 + 1 prior to contest)
  - Outer sandbox (not yet enabled)
  - Binary module loader
  - Trampoline interfaces (1 – direction flag)
  - IMC communications interface
  - NPAPI interface (3 – including same origin issues)
  - System calls (1 – unmap / map)
  - Browser integration (8)

NaCl Today and Tomorrow

Google
Developer
Day2009

Native Client Research Release

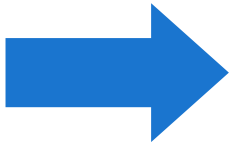
Research

- NPAPI plugin
- x86-32 only
- Raster graphics
- Mirrored public SVN

Native Client Developer Release…

Research

- NPAPI plugin
- x86-32 only
- Raster graphics
- Mirrored public SVN



Developer

- Built into browser
  - Web Workers
  - Revised NPAPI
- x86-32, x86-64, ARM
- O3D integration
- Develop off public SVN

Preview: Native Web Workers

- Web Workers: Simple threading model for the browser
  - No shared data, no DOM access
  - postMessage, XMLHttpRequest, openDatabase
  - See specification at **http://whatwg.org/ww**
- Goals of Native Web Workers:
  - Support workers in C, C++, Ruby, ...
  - Maintain the simplicity of the Web Worker model
  - Support 'low frequency' applications

Demo: Native Web Workers

Preview: Revised NPAPI

- Plugin use today is very limited
  - Well known security issues
  - Pop-up boxes asking unreasonable questions
  - API is under-specified
  - Web portability falls apart
- Creating a brighter future for plugins
  - Address known misfeatures of NPAPI, ActiveX
  - Avoid limitations of Web Workers
    - High frequency applications
    - Real-time applications
    - Synchronous DOM access

Example: H.264 Video Player

Porting a H.264 transcoder from Linux

- Based on a Google internal H.264 decoder
- Original test code decoded H.264 into raw frames
- 20-line change to create simple video player
- 230-lines to add audio and frame-rate control

Porting a Linux application to
Native Client can be very simple.

g264_unittest.c

```
int main(int argc, char *argv[]) {
 ...
#ifdef __native_client__
  int r = nacl_multimedia_init(NACL_SUBSYSTEM_VIDEO);
  if (-1 == r) {
    printf("Multimedia system failed to initialize!  errno: %d\n",
errno);
    exit(-1);
  }
  r = nacl_video_init(NACL_VIDEO_FORMAT_RGB, image_width,
image_height);
  if (-1 == r) {
    printf("Video subsystem failed to initialize!  errno; %d\n",
errno);
    exit(-1);
  }
  write_file_ptr = NULL;
#else
  write_file_ptr = fopen("output.yuv", "wb");
#endif
```

g264_unittest.c

```
#ifdef __native_client__
    YV12toRGB24_generic(img->luma_sample, img->luma_width,
                        img->chroma_sample[0], img->chroma_sample[1],
                        img->chroma_width, RGB24_out,
                        img->luma_width, img->luma_height,
                        img->luma_width);

    r = nacl_video_update(RGB24_out);
    if (-1 == r) {
      printf("nacl_video_update() returned %d\n", errno);
    }
#else
    fwrite(img->luma_sample,      frame_size,    1, write_file_ptr);
    fwrite(img->chroma_sample[0], frame_size>>2, 1, write_file_ptr);
    fwrite(img->chroma_sample[1], frame_size>>2, 1, write_file_ptr);
#endif
```

Demo: H.264 Video Decoder

Demo: Native Client Darkroom

Contribute

- Please visit us at http://code.google.com/p/nativeclient
  - Write new apps
  - Port existing C/C++ libraries
  - Help us test

Questions?

On the web: http://code.google.com/p/nativeclient

Related projects:

- Chromium: http://dev.chromium.org
- O3D: http://code.google.com/p/o3d

Appendix

Example Security Contest Issues

- #50: **`data16`** prefix with two-byte control flow instructions
- We had assumed **`data16`** only applied to data arithmetic, and was safe with all two-byte instructions
- Problem: **`data16`** also impacts some address calculations
- Solution: disallow **`data16`** for most two-byte instructions
- Solution: protect bottom 64KB of the address space

- #51: stack-smashing attack via **`eflags`** direction flag
- **`eflags`** state was preserved across trusted runtime calls
- Problem: Some Windows APIs use **`rep movs`** without checking flag direction
- Solution: use **`cld`** to clear flags during trusted runtime calls