Angry Birds on HTML5

Joel Webber <jgw@google.com>

goto conference
aarhus
2011


GET THE WHOLE FLOCK!
CLICK HERE!
ANGRY BIRDS
BETA
PLAY
SD VERSION
HD VERSION
SQUAWK!
Add this game to Chrome and play offline.
Like
8,416,952 people like this.
TRICK OR TREAT!
GET THE ANGRY BIRDS HALLOWEEN TOYS
Report A Bug

Why the web?

No install

Reach

Seamless update

Embeddable and linkable

Goals

- Fast startup
- Smooth 60 frame/s
- Cross-browser support

Moving Parts

Game Loop

Games are simply simulations:

- Get user input
- Update the world
- Render the world

World Model

- Static objects
  - Ground and slingshot
- Dynamic objects
  - Pigs, birds, blocks, and so forth
- Physics simulated using Box2D

- User state
  - Score, birds available, levels played, ...
  - View position, scale
  - Slingshot state

Rendering the world: Backgrounds

Several layers, scaled and repeated

Fill-rate is a challenge

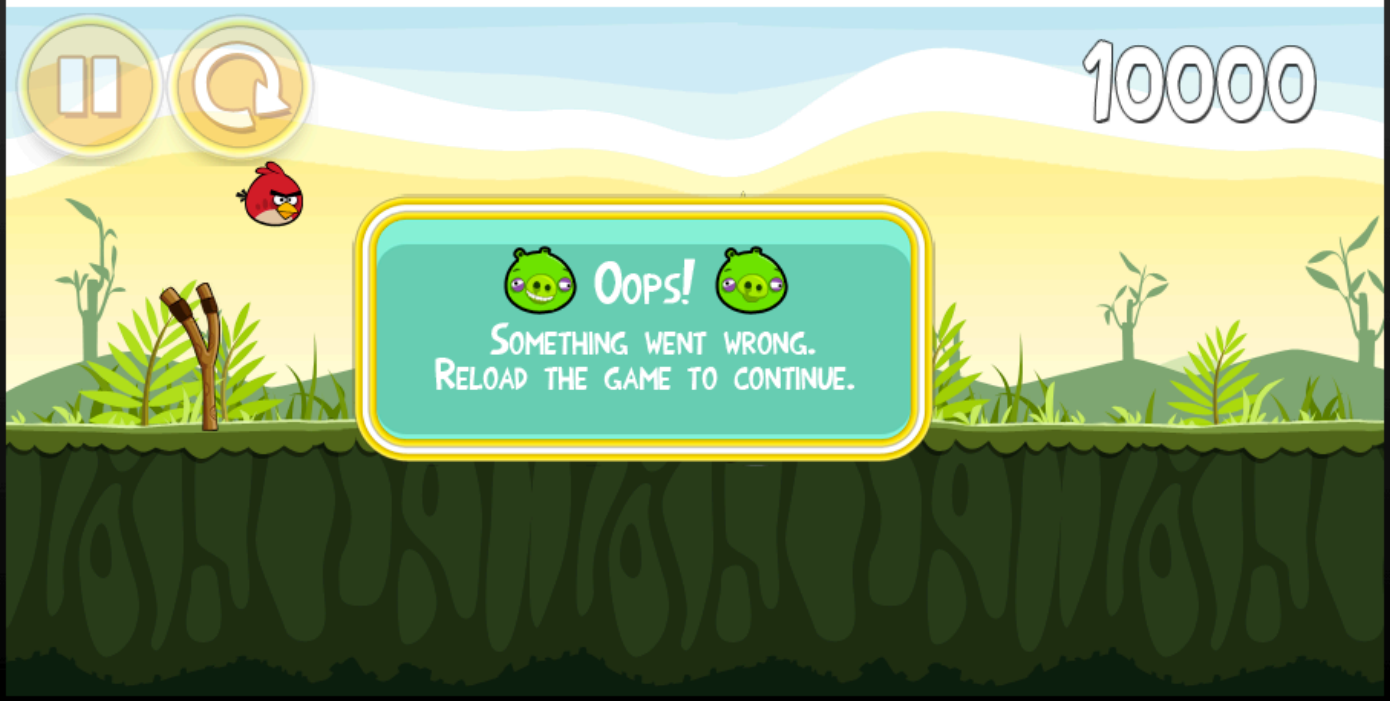
Rendering the world: Sprites

Lots of dynamic objects

- Blocks, blocks, and pigs
- Smoke trails and explosions

Rendering the world: UI

Layered on top of everything else



Resources

Images

Backgrounds, sprites, and UI

Audio

Music, ambient backgrounds, squawks and snorts

Resources

Object definitions and level data

```
{
  theme: "BACKGROUND_CLOUDS",
    world: {
      bird_1: {
        angle:0, id:"BIRD_YELLOW", x:58.472, // ...
```

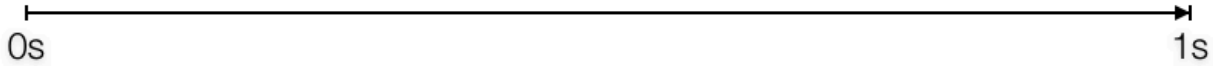
Sprite definitions

```
{
  image: "INGAME_PIGS.png",
  spriteCount: 54,
  sprite_0: { id:"PIG_KING_03", x:2, "y":2, width:131, // ...
```

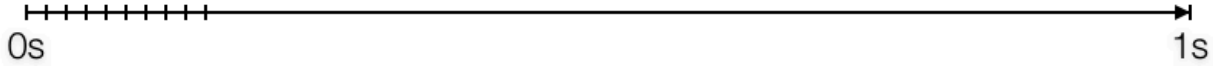
Challenges

Performance

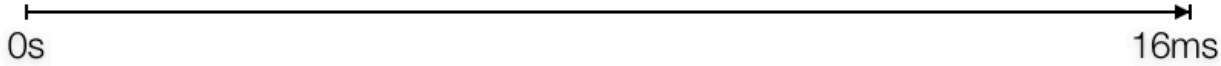
How much time do we have?



60 frames/s



~16 ms



User input, game logic, physics, rendering, garbage collection, ...

And you only get one thread!

Performance: Threads...

... or the lack thereof

Modern game engines separate simulation from rendering

- Take advantage of multiple cores
- Run simulation and rendering at different rates

Javascript is single-threaded

Could use HTML5 Web Workers in theory

- (but this is very complex)

WebGL mitigates this to some extent in Chrome

- GPU babysitting happens in another process

Performance: Rendering

DOM

- Use the browser's built-in element model
- Use CSS to control transforms
- Not as bad as you might expect, if you stay on the rails

```
<div style='
    background: url(pig.png) no-repeat;
    -webkit-transform: matrix3d(
      m00, m10,  0, tx,
      m01, m11,  0, ty,
        0,   0,  1,  0,
        0,   0,  0,  1
    );
  '
/>
```

Performance: Rendering

HTML5 Canvas

- Immediate-mode 2D API, similar to Apple's CoreGraphics
- Hardware-accelerated on many browsers

```
var ctx = canvas.getContext('2d');
ctx.save();
ctx.transform(m00, m01, m10, m11, tx, ty);
ctx.drawImage(pigImg, 0, 0);
ctx.restor();
```

Performance: Rendering

WebGL

- Hardware-accelerated 3D API, modeled on OpenGL ES 2
- By far the fastest approach...
- ...but not supported everywhere yet

```
var positions = new Float32Array([0, 0, 0, 1, 1, 1, 1, 0]);
gl.bufferSubData(gl.ARRAY_BUFFER, 0, positions);

var texCoords = new Float32Array([0, 0, 0, 1, 1, 1, 1, 0]);
gl.bufferSubData(gl.ARRAY_BUFFER, 0, texCoords);

gl.bindTexture(gl.TEXTURE_2D, tex);
gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, pig);

gl.vertexAttribPointer(posAttr, 3, gl.FLOAT, false, 0, 0);
gl.vertexAttribPointer(texAttr, 2, gl.FLOAT, false, 0, 12 * 4);

gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, indexBuffer);
gl.drawArrays(gl.TRIANGLE_STRIP, 0, 4);
```

Performance: Physics

Box2D does enormous amounts of matrix math

```
function mul(A, v, out) {
  out.x = v.x * A.m00 + v.y * A.m10;
  out.y = v.x * A.m01 + v.y * A.m11;
}

var A = new Mat(1, 0, 0, 1);
var v = new Vec(0, 0);
mul(A, v, out);
```

V8 generates 875 instructions for this mul() function

Performance: Physics

TypedArrays to the rescue!

```
function mul(A, v, out) {
  out[0] = v[0] * A[0] + v[1] * A[2];
  out[1] = v[0] * A[1] + v[1] * A[3];
}

var A = new Float32Array(1, 0, 0, 1);
var v = new Float32Array(0, 0);
mul(A, v, out);
```

V8 generates 376 instructions for this mul() function

Translates to roughly 30% performance improvement

Performance: Garbage-collection

- Fast, except when it's not
- Predictability is more important than raw overall speed
  - Remember: 16ms per frame
  - One pause can cause a frame skip
- Strategy
  - Pre-allocate world objects
  - Object pooling
- V8 gets a new garbage collector!

Startup time

The web's unique challenges

- The good news: You can make web apps start nearly instantly
- The bad news: Your users expect them to actually start instantly
- The worse news: The network's always slower than you think it is

Startup time: Caching

The cache can be your friend:

- The browser cache is a good start
  - Careful with those headers!
- The HTML5 AppCache is much more powerful

Much easier to perform incremental updates than with installed packages

Startup time: Sprites

Individual images are natural on the web

```
<img src='bird0.png'>
```

But HTTP requests are expensive!

Solution: sprite sheets

Startup time: Cheating

Pay no attention to the man behind the curtain

- You can't fix the network, but you can cheat!
- Only load what you need
- Whenever possible, hide loading where the user won't notice

Audio

- Two approaches:
  - The one that's deprecated
    - HTML5 <audio> tag is woefully inadequate for games
    - Particularly bad for low-latency audio
  - The one that doesn't work yet
    - The new Web Audio API isn't available everywhere yet
- How Web Audio will fix the problem
  - Explicit buffer management
  - Precise scheduling
  - Convolutions and other effects
  - Spatialization

PlayN

Google

What is PlayN?

- Java library for casual games
  - Introduced as "ForPlay" at Google I/O 2011
- Targets: HTML5, Flash, Android
- Desktop JVM used for development and debugging
- developers.google.com/playn
  - Open source
  - Already seeing significant contributions
  - Early days: contributions welcome!

Stop: Demo Time

PlayN: Goals

Simple

Reductionist

Cross-platform

Focused on the "middle of the bell-curve"

Components: Game Loop

Simply implement playn.core.Game

Ensures update() and paint() happen at the right time

```
public class MyGame implements Game {

  public void init() {
    // initialize game.
  }

  public void update(float delta) {
    // update world:
    //   delta indicates the time-step
  }

  public void paint(float alpha) {
    // render world:
    //   alpha indicates time in the range [0, 1) between world frames
  }
}
```

Components: Input

Simple abstractions for input devices

- Pointer, Mouse, Touch
- Keyboard

```
pointer().setListener(new Pointer.Adapter() {
  public void onPointerStart(Pointer.Event event) {
    // Handle mouse down event.
  }
});

keyboard().setListener(new Keyboard.Adapter() {
  public void onKeyDown(Event event) {
    // Handle key down event.
  }
});
```

Components: Graphics

Two main concepts

- Layers: retained structures (similar to DOM)
- Surfaces: immediate rendering (similar to Canvas)

Implemented using a combination of DOM, Canvas, and WebGL

```
public void init() {
  bg = graphics().createSurfaceLayer();
  graphics.rootLayer().add(bg);

  Layer catGirl = graphics().createImageLayer("catGirl.png");
  graphics.rootLayer().add(catGirl);
}

public void paint(float alpha) {
  Surface surf = bg.surf();
  surf.clear();
  surf.drawImage(cloud, cloudX, cloudY);
}
```

Components: Audio

Simple audio API

```
public void init() {
  Sound music = assetManager().getSound("ambient.mp3");
  music.setLooping(true);
  music.play();

  squawk = assetManager().getSound("squawk.mp3");
}

public void somethingHappened() {
  squawk.play();
}
```

Components: Asset Management

Simple loading methods for images, sounds, and text

```
public void init() {
  Image image = assetManager().getImage("bird.png");
  Sound sound = assetManager().getSound("squawk.mp3");

  // Completion callbacks are available
  image.addCallback(new ResourceCallback<Image>() {
    public void done(Image resource) { imageReady = true; }
    public void error(Throwable err) { imageFailed(); }
  });

  // Text is necessarily async
  assetManager().getText("level.json", new ResourceCallback<String>() {
    public void done(String resource) { loadLevel(json().parse(resource)); }
    public void error(Throwable err) { gameOver(); }
  });
}
```

Components: Network

Some network access already handled by AssetManager

You can also make direct HTTP requests

```
public void saveState() {
  Writer json = json().newWriter();
  json.key("id");    json.value(playerId);
  json.key("score"); json.value(playerScore);

  net().post("/saveState", json.write(), new Callback<String>() {
    public void onSuccess(String result) { }
    public void onFailure(Throwable cause) { tryAgain();}
  });
}
```

Components: Box2D

Box2D baked into the library

Why embedded?

- Somewhat tricky to do it yourself with JBox2D
- We can do some platform-specific optimizations

```
public void init() {
  world = new World(gravity, true);

  Body ground = world.createBody(new BodyDef());
  PolygonShape groundShape = new PolygonShape();
  groundShape.setAsEdge(new Vec2(0, height), new Vec2(width, height));
  ground.createFixture(groundShape, 0.0f);

  world.setContactListener(new ContactListener() {
    public void beginContact(Contact contact) { ... }
    public void endContact(Contact contact) { ... }
    // ...
  }
}

public void update(float delta) {
  // Fix physics at 30f/s for stability.
  world.step(0.033f, 10, 10);
}
```

Future work

- Input
  - Game pads and other input devices
- Rendering
  - 3d graphics API
- Audio
  - Audio effects and spatialization
- Network
  - Streaming sockets
- C (or possibly LLVM) backend
  - Support for iOS and other platforms

The future of web games

- Advanced APIs for game developers
  - WebGL, multiple render targets, deferred shading
  - Low-latency audio, filters, and effects
  - Full-screen, mouse lock
- HTML5 Everywhere?
  - All about performance!

Thanks

Serdar Soganci (Rovio)

Philip Rogers, Seth Ladd, Lilli Thompson (Google)

Michael Bayne (Three Rings)

Questions?