WebGL tricks, a brief talk

1M LoC, 1 gigapixel, Flash to html5.
4 years of The Household conversion

- Game: [The Household - Home](#)
- 10000 Cows: [WebGL 2D Engine test](#)
- Interaction: [WebGL 2D Engine demo with interaction](#)
- Full result: [Village WebGL](#)
<table>
<thead>
<tr>
<th>Size of project</th>
<th>&amp;</th>
<th>Our audience</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 30M installs</td>
<td></td>
<td>Respectable housewifes,</td>
</tr>
<tr>
<td>2. 40 years wasted each day by players in game.</td>
<td></td>
<td>many with potato PC.</td>
</tr>
<tr>
<td>3. 10 years of development by team of 30 coders and artists</td>
<td></td>
<td>&gt; 95% has WebGL1 available</td>
</tr>
<tr>
<td>4. After conversion probably biggest PixiJS project</td>
<td></td>
<td>1. Instancing extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Float-texture extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. VAO extension</td>
</tr>
</tbody>
</table>

I am going to show a few tricks that made it into production
Project structure

Game Field

Scrollable area with thousands of animated elements.
1. Instancing
2. Raster data packed into compressed textures
3. Animation data packed into atlases of float textures. Shader animation.
4. Low CPU load if not interacted with.
5. PixiJS to hold all this architecture.

Flash UI

Shops, notifications, e.t.c.
1. It sells!
2. Artists went wild and used everything that exists in Adobe Animate. ColorTransform, filters, e.t.c., all packed in SWF
3. Mozilla Shumway derivative for stage tree
4. We decided to implement all features rather than rewrite the UI logic or re-pack resources. Yes, we are crazy.
Game Field tricks
Overdraw

Question: If we put one big quad element in a dense field, how many elements will draw the same single pixel?

Answer: up to 70.
Fight against overdraw

Trick for old PC's borrowed from old 3D games.

Drawing same pixel 70 times => drawing whole screen 70 times.

For old video cards, writing colors to pixel costs a lot, z-buffer is fast.

1. Assign an ID to each element, - it is depth, Z-coord.
2. Turn on z-culling, turn off colors, write only to depth buffer
   ```
   gl.disable(gl.BLEND);
   gl.enable(gl.DEPTH_TEST);
   gl.depthFunc(gl.LESS);
   gl.depthRange( zNear: -1, zFar: 1);
   gl.colorMask( red: false, green: false, blue: false, alpha: false);
   gl.depthMask( flag: false);
   ```
3. If (result.a < 2.5 / 255.0) { Els with alpha less than
   ```
   discard;
   ```
4. Turn on colors
   ```
   gl.enable(gl.BLEND);
   gl.enable(gl.DEPTH_TEST);
   gl.depthFunc(gl.LEQUAL);
   gl.colorMask( red: true, green: true, blue: true, alpha: true);
   gl.depthMask( flag: false);
   ```
5. Draw everything back-to-front, as usual

Good for old integrated video cards (Intel HD 4000), does not affect discrete.
Interaction hell

Each element consists of several quads, quads are regions in atlas.

Do you want to write algo to find selection?

I don't.
Pixel-perfect interaction

JS is slow, and we are lazy coders. How to get which object is under the mouse without checking each bitmap and finding corresponding pixel in it?

1. Assign each element an index, and corresponding color vDepthColor

2. Pass scanPixel - mouse coordinate to uniforms

3. Put an "if" to every shader.

```javascript
if (result.a < 2.5 / 255.0) {
    discard;
}
if (scanPixel.xy == gl_FragCoord.xy) {
    result = vec4(vDepthColor, 1.0);
}
gl_FragColor = result;
```

4. Exactly in the end of frame call readPixels() to read the value from the framebuffer. If you call it in random mouse event - it won't work!

5. Wait for it. Profiler shows that CPU is used, but in reality it is just a blocking operation, it does not consume CPU and other threads of browser and processes on user computer are not hurt

6. Multiple clicks per frame? Put them in queue.

Method was as a joke by the biggest troll in CrazyPanda. He did not think we were really going to put it into production.
Flash trick

Vector Shapes
Shapes problem. The story.

Triangles, squares, circles, pentagrams are paths. Vector image consists of many paths filled or stroked.

To make edges of shapes look like edges and not a pixel ladder, there are a number of anti-aliasing techniques.

Following solutions help to use different AA types.
Canvas 2D FTW!

Each shape is described like with paths, fills: moveTo(), lineTo() and other methods.

Canvas 2D context has those methods! You can even draw ponies on it.

WebGL has no vector support, we have to create mesh representation using special libraries.

However, we can use `texImage2D(canvasWithShape)`
Try to draw a cow on Canvas 2D

Butcher is somewhere nearby!

His name is **CONFLATION**.

Canvas2D uses Analytical AA which has this problem.

Temp solution: prepare the imageData!

```javascript
getImageData();
for (let i = 3; i < data.length; i += 4) {
    if (data[i] > 153) {
        data[i] = data[i] * 10 - 153 * 9;
    }
}
putImageData();
```

Change alpha: $f(0.6) = 0.6$, $f(0.7) = 1.0$
Draw a guy in SVG

```xml
<defs>
  <filter id="filter">
    <feComponentTransfer color-interpolation-filters="sRGB">
      <feFuncA type="table" tableValues="0 1 1"/>
    </feComponentTransfer>
  </filter>
</defs>
<g filter="url(#filter)">
  <!-- SVG code here -->
</g>
```
Conflation. Random facts

Fact #1. Even Google doodle had it (memorial day 2018)

Fact #2. Sometimes Chrome uses MSAA for SVG. Depends on video card (> nvidia pascal ), windows version and phase of moon.
Not everything is recoverable, 2D context is not enough.
Why not draw shapes with WebGL?

1. Use "PIXI.Graphics" or any other graphics implementation.

2. Use earcut or tess2 to separate a shape to triangles.
   (Earcut is faster, tess2 knows how to deal with random unspecified holes.)

3. Holy cow!
WebGL context antialias parameter

Add "antialias: true" in context creation parameters.

1. MSAA algorithm is not affected by conflation… probably
2. Lines are a bit worse than in canvas 2D context, but OK
3. AwayFL loves it!
Shock: "antialias: true" is evil.

1. MSAA has conflation issue if shape is drawn on top of another shape
2. Does not affect shapes drawn in temporary texture or framebuffer
3. Macbooks are on fire and sound like an aircraft taking off
We can pre-render shape into separate texture!

Pre-render shapes into temp texture, might be an atlas

When rendering main frame, draw regions from atlas

Thus, two shapes will not be AA-ed on top of each other

Required: special MSAA renderbuffer storage.

```javascript
var colorRenderbuffer = gl.createRenderbuffer();
gl.bindRenderbuffer(gl.RENDERBUFFER, colorRenderbuffer);
gl.renderbufferStorageMultisample(gl.RENDERBUFFER, 4, gl.RGBA8, FRAMEBUFFER_ATTACHMENT_COLOR_ATTACHMENT0);
```
MSAA framebuffer is not easy to use!

1. Increases memory consumption several times
2. blit() method - need second framebuffer and copy operation
3. available only in WebGL2
4. you know any app that uses it in production?
   Please send me an email about it.

```javascript
gl.bindFramebuffer(gl.READ_FRAMEBUFFER, framebuffers[FRAMEBUFFER.RENDERBUFFER]);
gl.bindFramebuffer(gl.DRAW_FRAMEBUFFER, framebuffers[FRAMEBUFFER.COLORBUFFER]);
gl.blitFramebuffer(
    0, 0, FRAMEBUFFER_SIZE.x, FRAMEBUFFER_SIZE.y,
    0, 0, FRAMEBUFFER_SIZE.x, FRAMEBUFFER_SIZE.y,
    gl.COLOR_BUFFER_BIT, gl.NEAREST
);```
Jimber.io entered the chat

Jimber: PixiJS Graphics is clearly not enough for 2D shapes
Jimber: Skia CanvasKit is the best solution

Pixi: We cannot use PixiJS in same context, also if Pixi uses WebGL2 and Skia is WebGL 1...

Jimber: OK, it works and performance is great!

```
// take shape from skia
const myTexture = PIXI.Texture.from(skiacanvas);
// make new shape
skiaCanvas.stroke(path);
// notify pixi of skia canvas update
myTexture.update();
```

Pixi: Wait, WHAT? What kind of sorcery is this?
How to use MSAA on shape in texture in WebGL 1

1. Create single extra canvas with WebGL 1 context

2. Pack shapes that you need this frame to atlas and draw in this canvas

3. On main context, bind the texture and use

```javascript
gl.texSubImage2D(gl.TEXTURE_2D, level: 0, xoffset: 0, yoffset: 0, gl.RGBA, gl.UNSIGNED_BYTE, extraCanvas);
```

4. Watch how `texImage2D` transfers data in the usual async way, no `readPixels()`

(Here's Linux log)
Pre-rendering shapes before frame

1. Collect shapes for current frame
2. Separate by AA type: canvas2d/webgl
3. Pack into one or more atlases
4. Draw atlas to AA surface
5. Blit AA surface to texture
6. Copy regions or entire atlas to cache if you want to save few shapes for later
7. Draw on screen

```javascript
if (canvas2d) {
    if (antiConflation) {
        const data = canvas2D.getContext().getImageData();
        gl.texSubImage2D( fixImageData(data) );
    } else {
        gl.texSubImage2D(canvas2D);
    }
} else if (webgl1) { // WebGL 1 copy HACK
    gl.texSubImage2D(canvas2WebGL);
} else { // WebGL 2
    gl.bindFramebuffer(); gl.blitFramebuffer();
}
```
Conclusion
Conclusion

1. WebGL is awesome. Learn WebGL, do not wait for WebGPU, start now!

2. If WebGL does not have particular feature - sometimes you can hack it.

3. Please share your low-level tricks! Regardless of rendering engine, everything goes into collective knowledge.