Leveraging multi-draw to improve performance

© 2020 Zea Inc. Not for reproduction or distribution.
Anatomy of a ‘Draw Call’

1. Bind a shader
2. Bind a geometry
3. Bind some uniforms
   a. materials
   b. matrices
4. Draw call!
Rendering Performance

- Nb Triangles x **Shaders** x **GPU Perf**
- Nb Draw calls x **Cost per call**
Rendering Optimizations

- Reduce the number of drawn triangles
  - LODs
- Reduce the cost of each draw call
  - Uniform buffer objects (UBOs)
  - Vertex buffer objects (VBOs)
  - Batching calls which share geometries, materials, shaders
- Reduce the number of draw calls
  - Frustum culling
  - Occlusion culling
  - Instancing
  - Multi-draw
**Multi-draw vs Instancing**

**Instancing:** Draw the same geometry multiple times.

**Multi-draw:** Draw the different geometries multiple times.
Warcraft Movie

The Road to One Million Draws

glMultiDrawElementsIndirect
Zea Engine Architecture - Before multi-draw

Scene Tree (Model)

- Scene Tree
  - Root item
    - Tree item
      - GeomItem
        - Material
        - Geometry
    - Asset item
      - GeomItem
        - Geometry
        - Material

Renderer (View)

- Renderer
  - Opaque
  - GLShader
    - GLMaterial
      - GLGeom
      - GLGeomItem
  - Transparent
  - Custom Pass
    - Shader
      - GLMaterial
      - GLMaterial
      - Instanced!

Shared Geometry and Material

Instanced!
- All geometries must be packed into one large vertex buffer
- All element indices must use the same precision
  - We use UNSIGNED_INT
- Uniforms/UBOs cannot be changed between draws
- Instanced attributes are not incremented between draw calls.
  - Instead we have `gl_DrawID`
All geoms in a batch must be packed into a shared set of vertex attributes

- Single large VBO for all geoms
- Element Indices converted to UNSIGNED_INT and offset added
Allocator1D Class

Allocate several blocks

Allocate/Free

Resize elements

Resize container
  - Power of 2 size

Defragment*
Pack material params into textures that can be indexed and read at render time.

Each color value stores 1-more material parameters.

Each Shader specifies a packing layout.
Draw data packing

Draw item data is packed into a texture.

- Flags
- GeomID
- MaterialID
- Model Matrix
- Bounding Box
A ‘Draw Set’ is all items drawn by a given ‘multi-draw’ call.

E.g. geoms assigned a given shader.

Maintains array of counts and offsets which ‘select’ geometry from the library.

The offset and count for a given geometry may occur many times within each draw set. (Instancing)

updateDrawIDsBuffer(renderstate) {
    {
        this.drawElementOffsets = new Int32Array(this.visibleItems.length)
        this.drawElementCounts = new Int32Array(this.visibleItems.length)
        this.visibleItems.forEach((glGeomItem, index) => {
            const offsetAndCount = ...
            this.drawElementOffsets[index] = offsetAndCount[0]
            this.drawElementCounts[index] = offsetAndCount[1]
            this.drawIdsArray[index] = glGeomItem.drawItemIds
        })
        ...
    }
}

multiDraw(renderstate, drawIds, counts, offsets) {
    ext.multiDrawElementsWEBGL(
        gl.TRIANGLES,
        counts, 0, gl.UNSIGNED_INT,
        offsets, 0, counts.length
    )
}
Shader Flow: What happens during drawing...

Vertex Shader

- DrawSet Texture
  - DrawItemld
  - geomld
  - materialld
  - modelMatrix

Fragment Shader

- Draw Items Texture
  - drawItemld
  - geomld
  - materialld
  - modelMatrix

Materials Texture

- baseColor
- roughness
- metallic

Note: gl_Drawld is considered ‘uniformly varying’, so texture lookups will be cached across a given draw.
Most transparent geoms in one large draw set.

Offsets and Counts are sorted using the item distance to camera as key.

Draw with blending.
Results
Results

17k parts
2.5M Triangles
300 Materials
2 Draw Calls
60 Fps
multiDraw(renderstate, drawIds, counts, offsets) {
    if (ext) {
        ext.multiDrawElementsWEBGL(
            gl.TRIANGLES, counts, 0, gl.UNSIGNED_INT, offsets, 0, counts.length
        )
    } else {
        const { drawId } = renderstate.unifs
        for (let i = 0; i < counts.length; i++) {
            gl.uniform1i(drawId.location, drawIds[i])
            gl.drawElements(gl.TRIANGLES, counts[i], gl.UNSIGNED_INT, Offsets[i])
        }
    }
}

Multi-draw Emulation (Safari, Firefox)

17k parts
2.5M Triangles
17k Draw Calls
36 Fps
Limitations of Multidraw

- Non-trivial architecture needed
- All geoms in a batch must share structure
  - Texture coords, normals, etc...
  - Interleaved/non-interleaved
- All geoms in a batch must share precision.
  - All indices converted to UNSIGNED_INT
- Changing materials is tricky
  - Changing textures is impossible
- Multi draw *indirect* calls would be nicer!