Optimised Effects With Mobile Graphics

Stacy Smith
Senior Software Engineer, ARM

In the next 45 minutes...

- What is batching and why bother?
- How to construct and use batches?
- Batch splitting
- Batch cleverness

- What's the deal with bandwidth?
- Skin and bone
- Procedural motion
- Texture manipulation

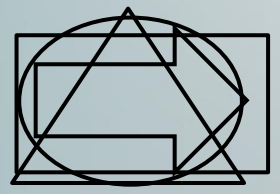
Batching

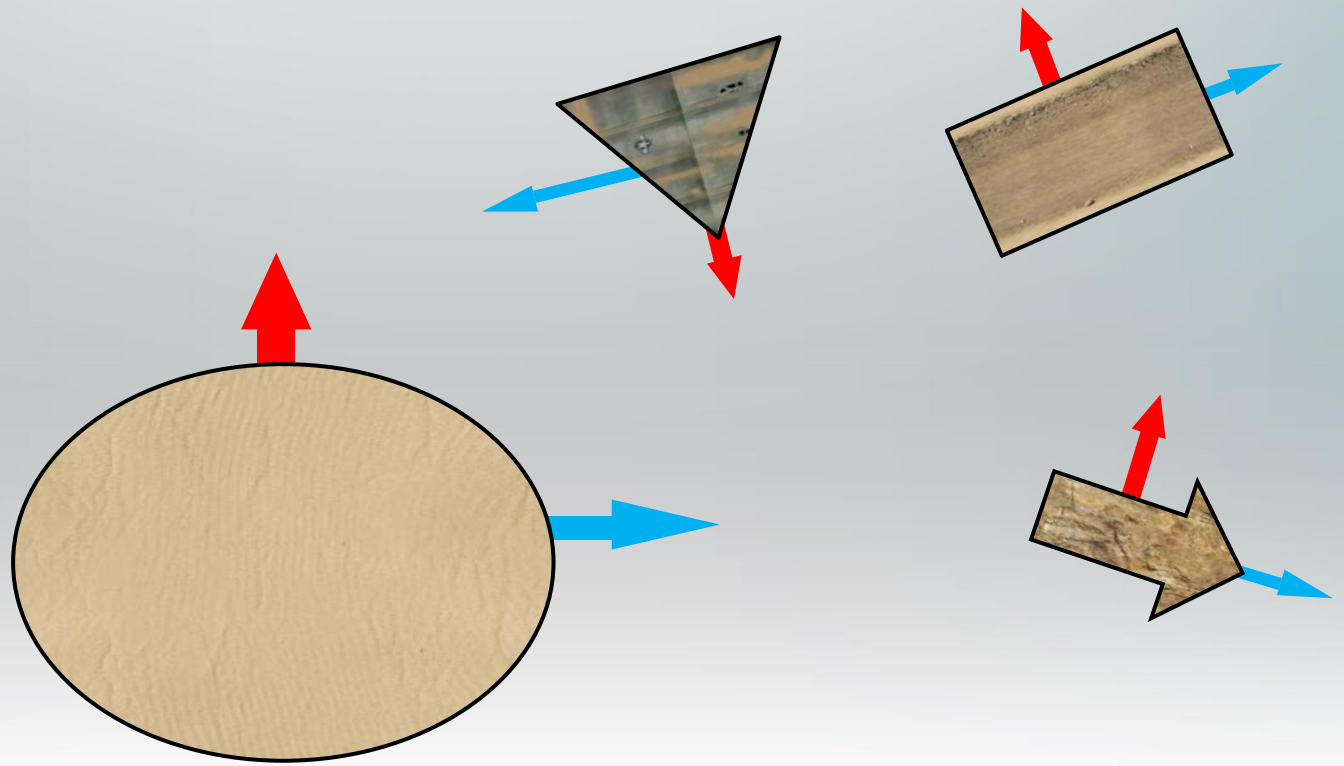
- Deferred immediate mode rendering
- glDrawElements and glDrawArrays have an overhead
- Less draw calls, less overhead.
- DrawCall class stitches multiple objects into one draw
- Macro functions in shaders make batching as simple as:

```
vec4 pos=transform[getInstance()]*getPosition();
```

Batching

Batching





uniform mat4 transforms[4];

Mixing up a Batch

Mesh 1 [(1,1),(0,1),(0,0),(1,0)]
Mesh 2 [(1,2),(0,2),(0,0)]
Mesh 3 [(2,2),(2,1),(1,1)]
Index1: [0,1,2,0,2,3]
Index2: [0,1,2]
Index3: [0,1,2]

Attrib1:[(1,1),(0,1),(0,0),(1,1),(1,2),(0,0),(1,0),(2,2),(2,1),(1,1)]
Attrib2:[0,0,0,0,1,1,1,2,2,2]

Index: [0,1,2,0,2,3,4,5,6,7,8,9]

Serving up a Batch

Vertex shader:

```
uniform mat4 transforms[3];
attribute vec4 pos;
attribute float id;
void main(){
    mat4 trans=transforms[(int)id];
    glPosition=trans*pos;
}
```

GL code:

```
float transforms[16*instanceCount];
.
. /* Load matrices into float array */
.
glUniformMatrix4fv(transID,4,false,transforms);
```

Object Instancing

- Multiple geometries, or single object instances:

```
for(int i=0;i<50;i++)
  drawbuilder.addGeometry(geo1);
drawbuilder.Build()
```

- Can implement LOD switching, when objects are sorted front to back and correctly culled.

- Seen in TrueForce

Scene batching

- Multiple geometries drawn in a single drawcall:

```
drawbuilder.addGeometry(geo1);
drawbuilder.addGeometry(geo2);
drawbuilder.addGeometry(geo3);
drawbuilder.Build();
```

- Always draws in the same order from a start and end index unused objects can be scaled out to zero

- Can lead inefficiency of vertex shaders

A view to a cull

Scene batching

Timbuktu has 22 objects per environment layer

S= Skipped R=Rendered X=Scaled out

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
		R	R	R							R	R						R		R	

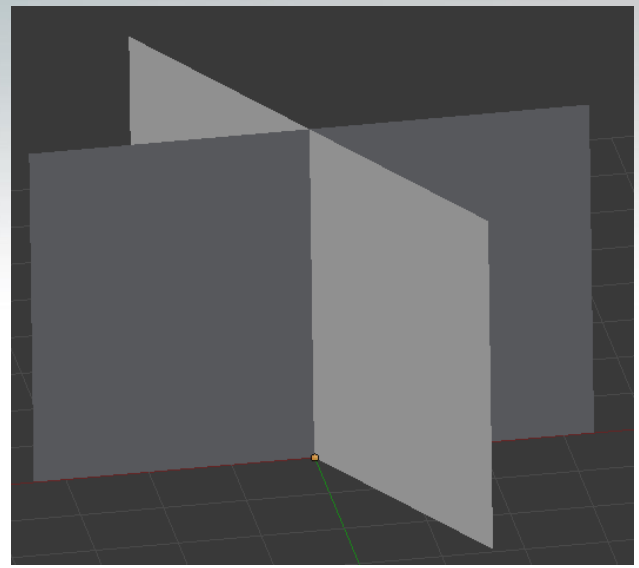
Imposterable!

- Timbuktu's Foliage:
  - 4 types of alpha blended geometries, depth sorted
  - Unpredictable interleaving of Grass, Tree, Bush and Shrub
- First suggestion:
  GTBSGTSBGTSBGTSBGTBSGTBSGTSBGTSBGTS
  - No more than 3 dead models in a row
  - Clumping of similar entities makes worst case more common
- Second suggestion
  GGTTBBSSGGTTBBSSGGTTBBSSGGTTBBSS
  - Allows consecutive pairs more cheaply
  - Worst case still arises frequently, and is now even worse

Imposterable!

- Solution:

- Foliage only varies in texture and scale
- All models topologically identical
- Zero skips
- Extra vec4 per instance



How wide is your band?

- Mobile GPUs (especially ours) have heaps of processing capability
- The limiting factor is bandwidth, and that's not about to change
- Bandwidth is:
  - Geometry Attributes
  - Indices
  - Textures
  - Uniforms

Still Life

- The bad old days of OpenGL® ES 1.1 had no concept of programmable shaders
- What you put in was what you got out
- Anything which is animated needed to have all geometry re-sent
- Bandwidth restrictions became animation restrictions

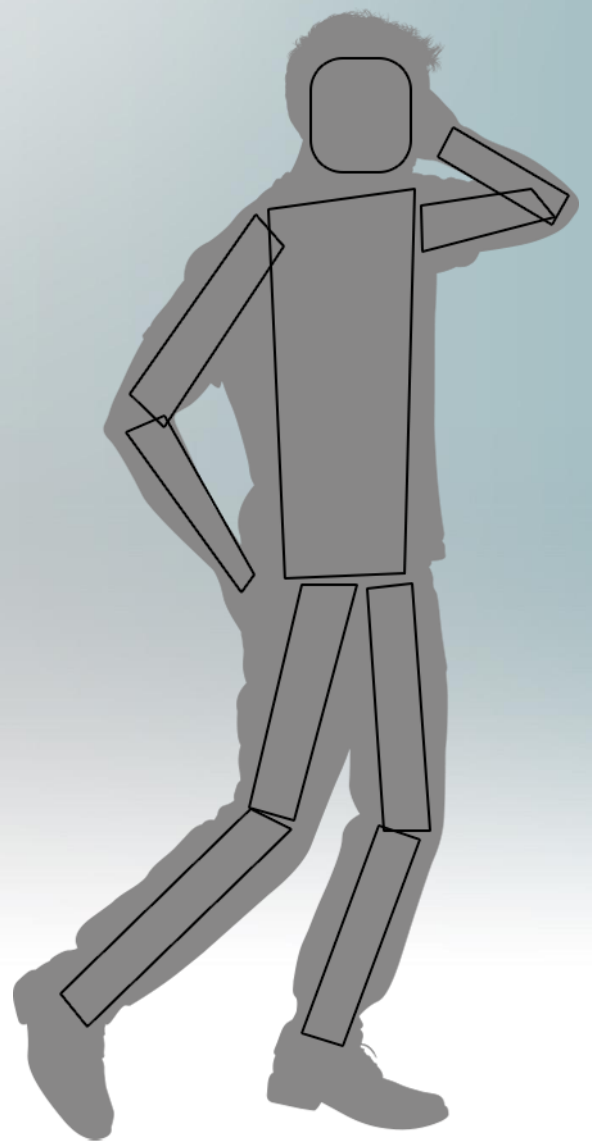
The trouble with Skeletons

- OpenGL ES 2.0 enabled shader side Skeletal Animation

- Vertices attached to bones, bone matrices sent in uniforms

- A basic skeleton is 10 bones
- Each bone is a matrix of 4x3 floats
- 120 floats is 480 bytes
- Per animated model
- Per frame

- For complex, designed animations, this is still the best solution we have



Skinless and Boneless

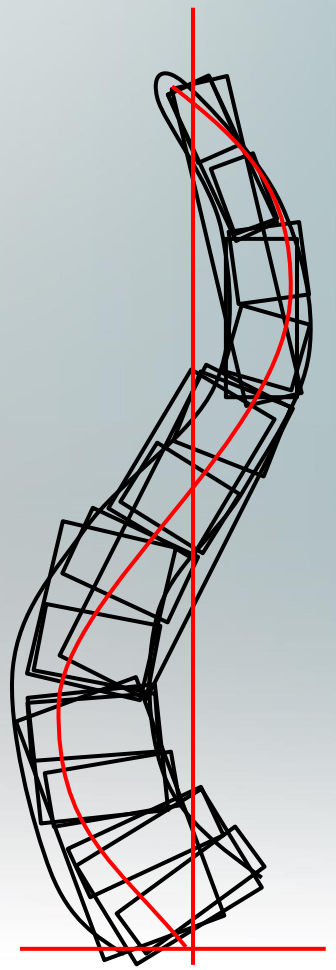
- What about a simpler form? Like a tentacle?
- 3 Bones : 144 bytes per frame

- Bone interpolation for smoother forms
- Multiple bone weight adds to geometry size

"Infinite complexity can be defined by simple rules"

~~-Benoit Mandelbrot~~

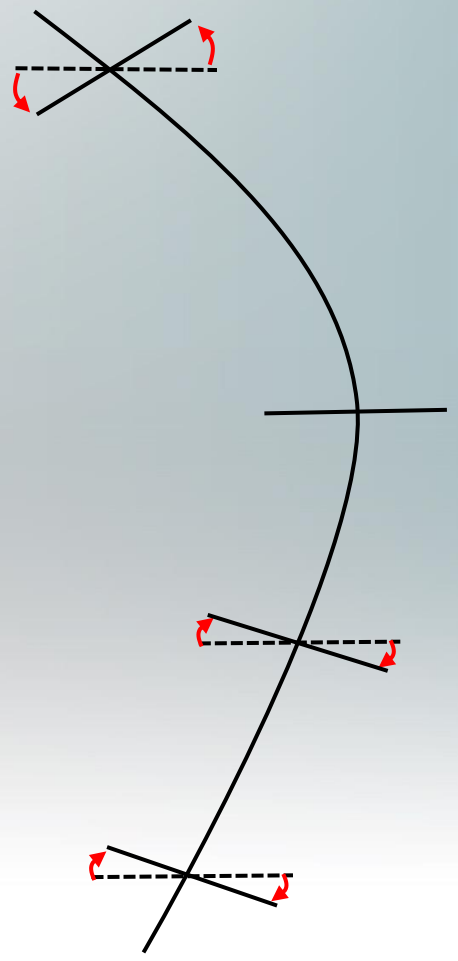
-Jonathan Coulton



```
x+= sin(y+t);
```

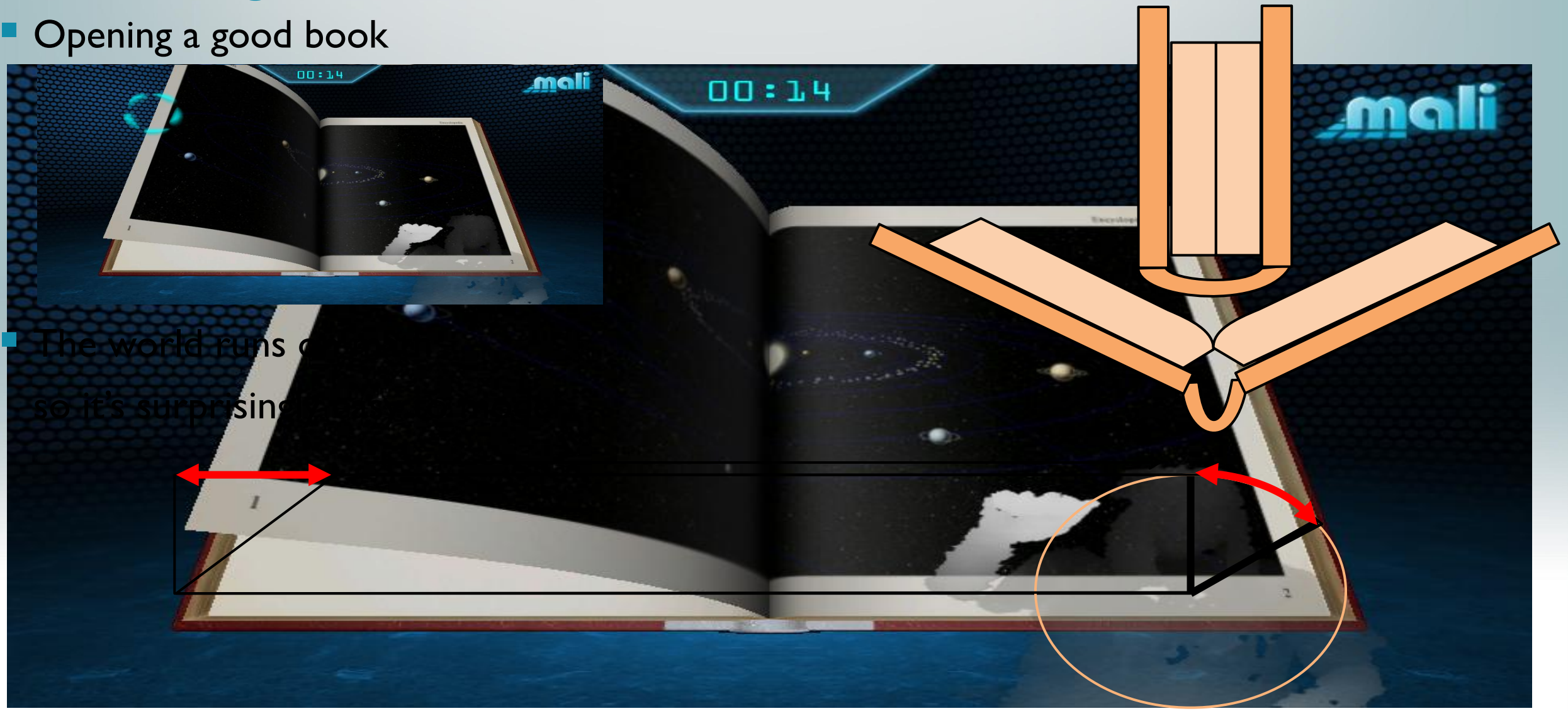
Following Procedure

- The actual maths is a little harder
- Nudging to the side 'works' but needs:
  - Vertices moved to factor in new angle
  - Similar adjustments to normals and tangents
- Any continuous function can be adjusted to calculate these other values
- If all else fails, throw trig at it



Following Procedure

- Opening a good book



- The world runs o
  so it's surprisin

Following Procedure

- It doesn't end at sines cosines and tangents
- Mathematics contains a plethora of curvature functions
- ARM® Mali™-T604 and the entire Mali-T600 series have unified shader core architecture
- This means you can do texture reads in the vertex shader
- Useful for terrain deformation

Moving Pictures

- What's good for the vertex is good for the fragment
- Time values, offsets, sine waves all usable on the fragment side
- Simple applications:
  - Plasma effects
  - Wavey surface normals
  - Texture distortion

Particle Lighting

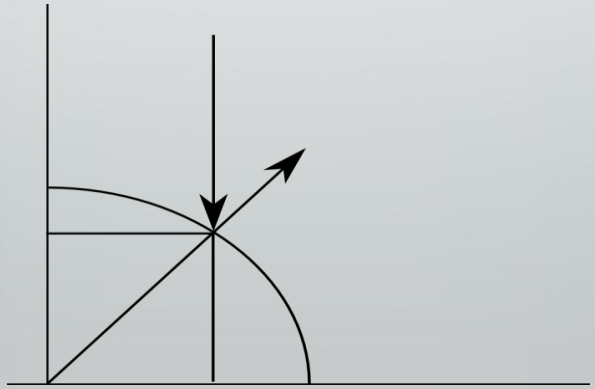
Pseudo-raytracing:

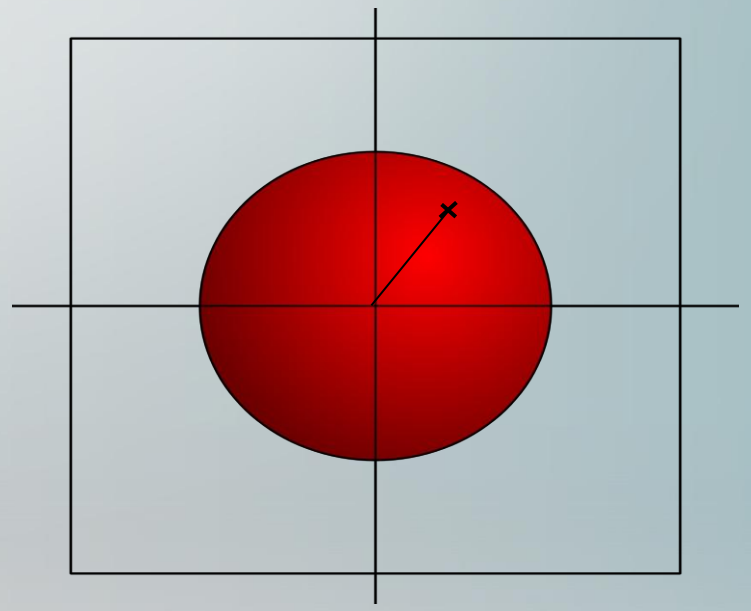
$a^2+b^2=c^2$

$a^2=c^2-b^2$

$a=\sqrt{(1-b^2)}$

$b^2=x^2+y^2$

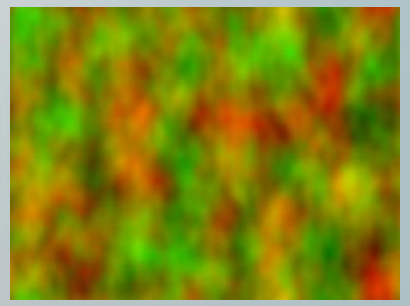


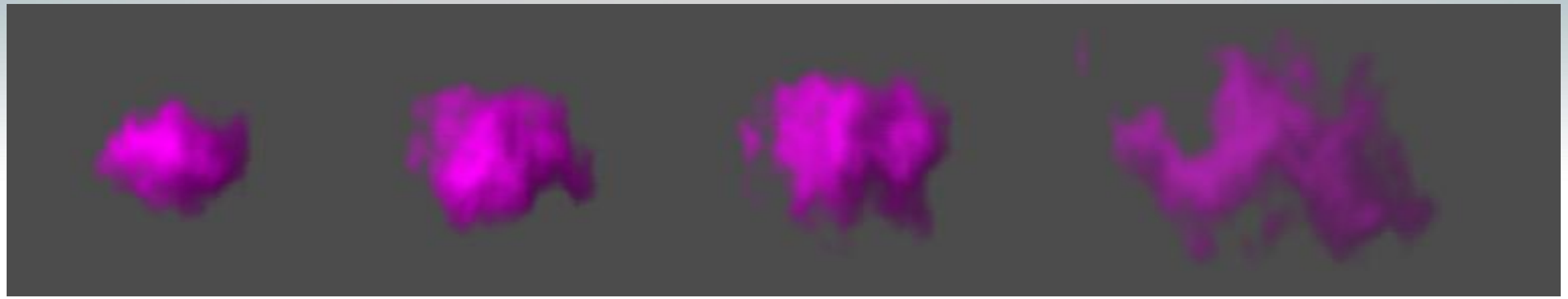


```
float r=tex.x*tex.x+tex.y*tex.y;
if (r<1){
  viewNormal=vec3(tex.x,tex.y,sqrt(1-r));
}else discard;
```

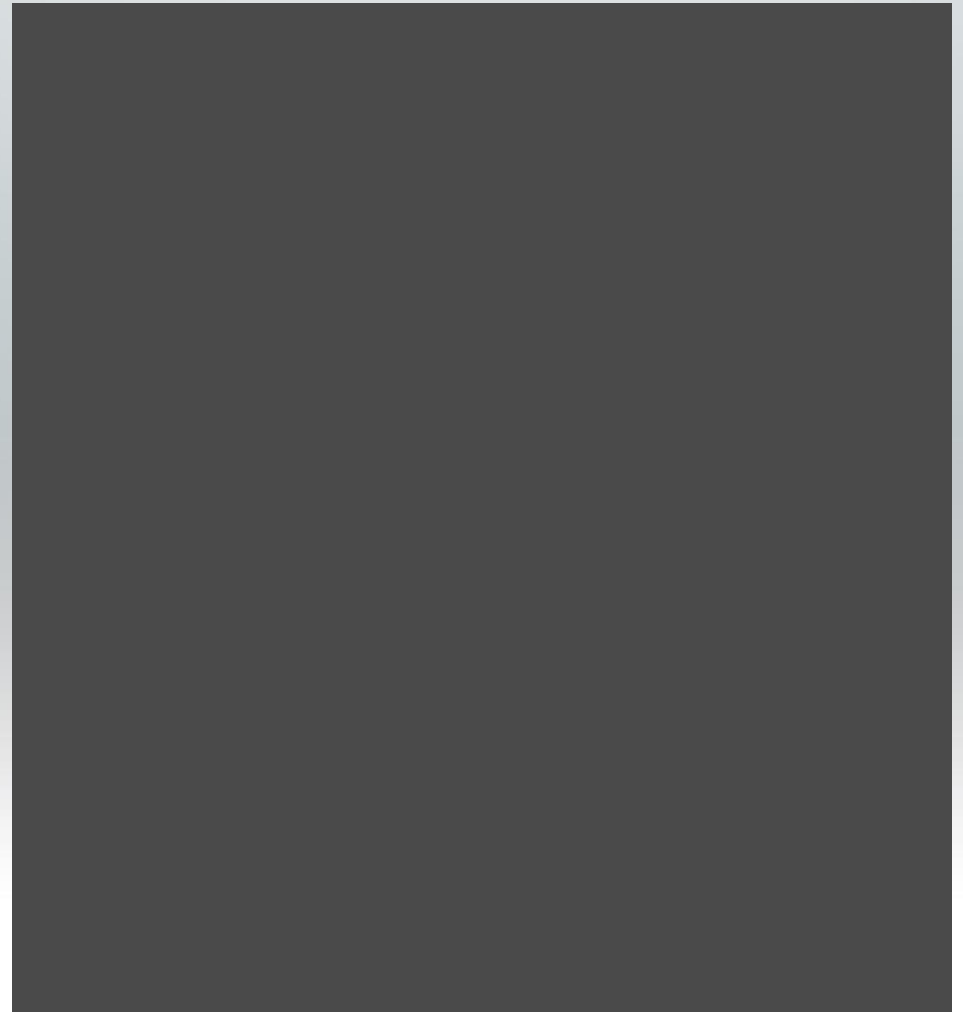
Particle Lighting

- Displacement Mapping:
- Texture offset to X and Y coordinates
- Distortion strength increases over time

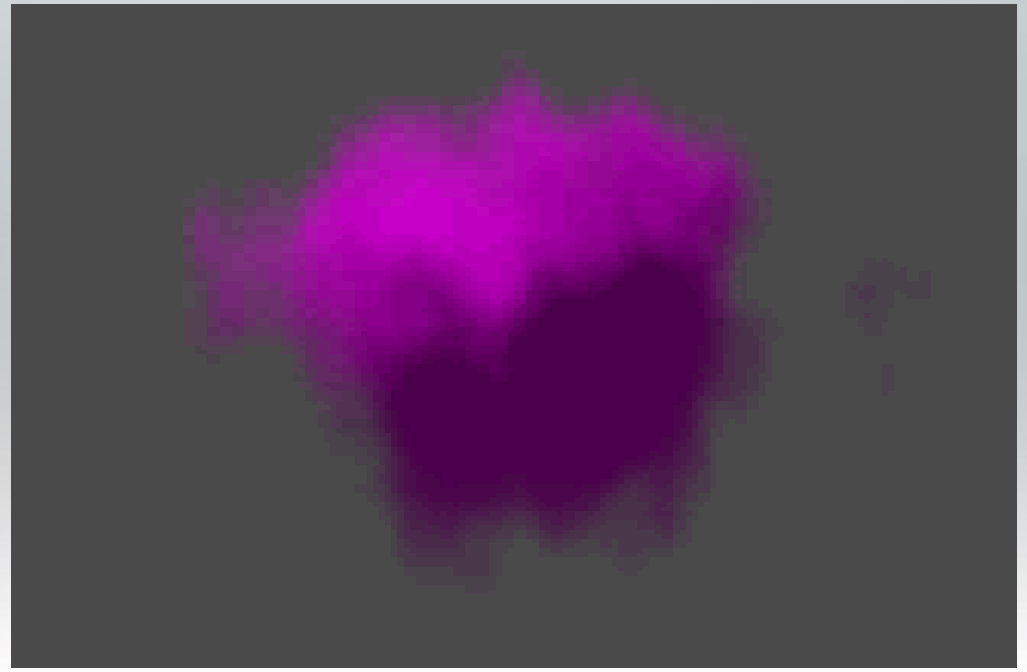




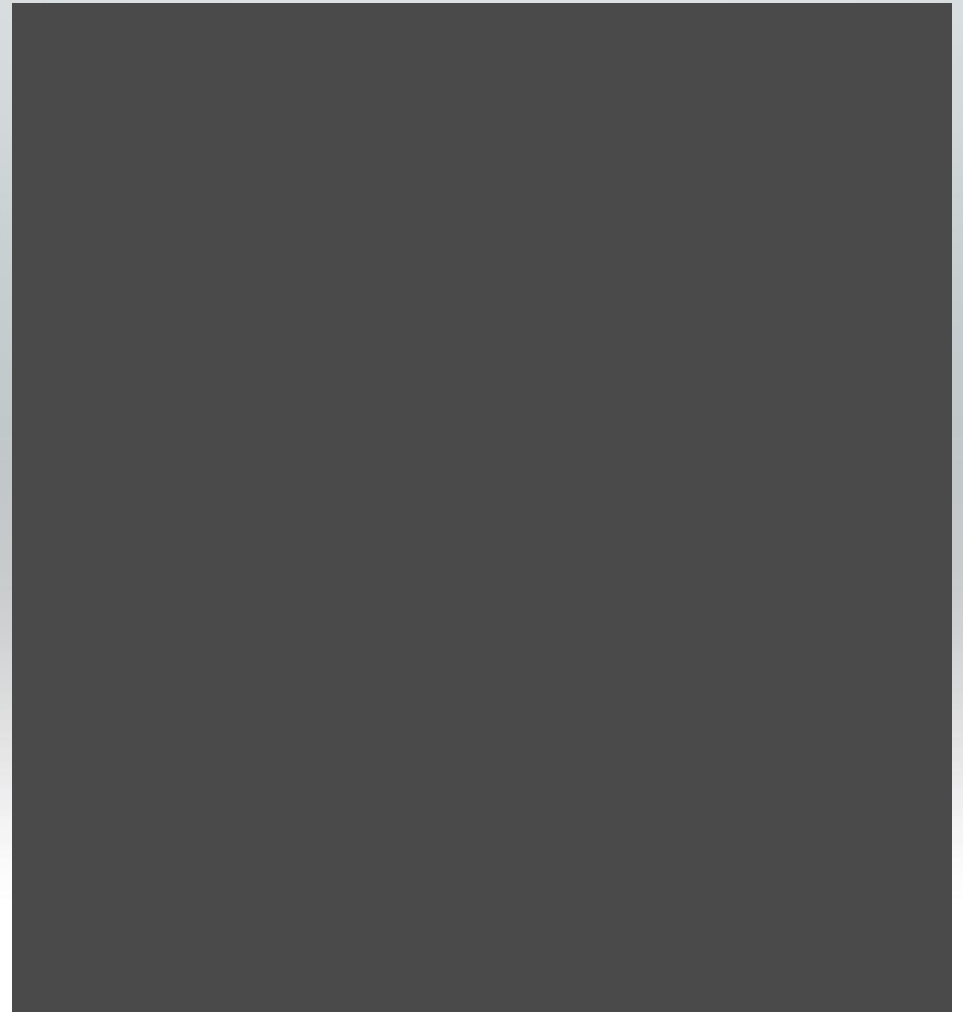
Particle Lighting



Particle Lighting



Particle Lighting



In the last 45 minutes...

- What is batching and why bother?
- How to construct and use batches?
- Batch splitting
- Batch cleverness

- What's the deal with bandwidth?
- Skin and bone
- Procedural motion
- Texture manipulation

Thank you
Any questions?

malideveloper.arm.com