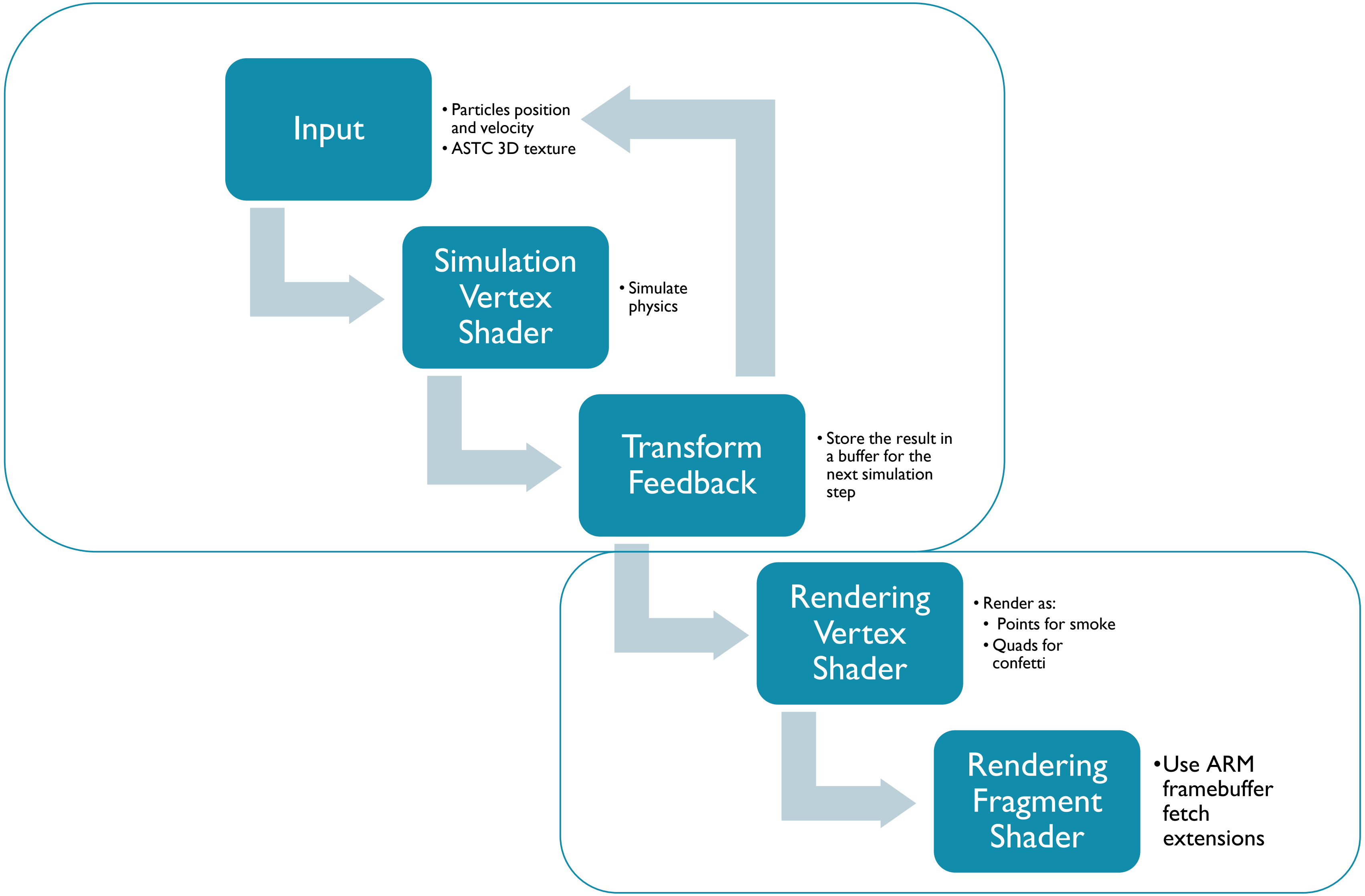
ASTC: The Extra Dimension

Daniele Di Donato
Senior Software Engineer, ARM

Particles system overview



- Particles position and velocity
- ASTC 3D texture

- Simulate physics

- Store the result in a buffer for the next simulation step

- Render as:
  - Points for smoke
  - Quads for confetti

- Use ARM framebuffer fetch extensions

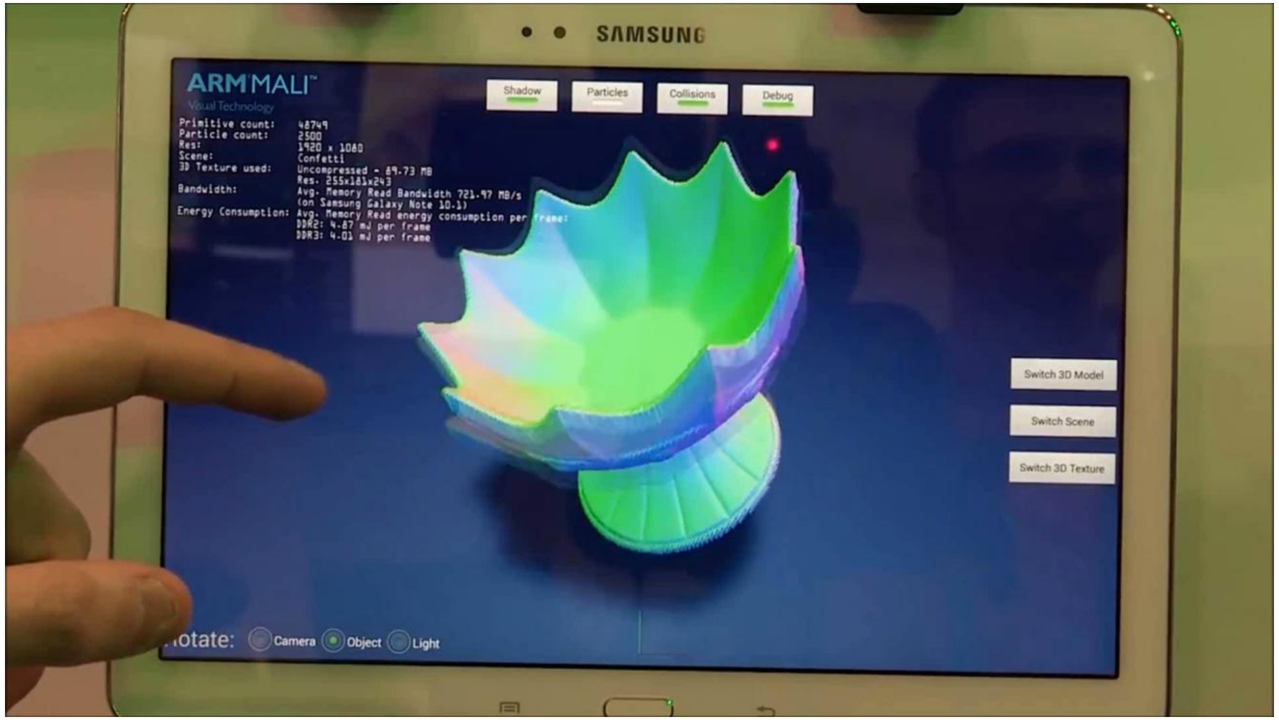
Physics Simulation

3D textures for collision simulation

- Since the simulation is run in the vertex shader we need to provide information about the environment to collide with

- Voxelizing the environment allow us to save it as 3D texture

- In the vertex shader simulate the physics and sample the 3D texture to check for collisions



```
Primitive count:       48749
Particle count:        2500
Res:                   1920 x 1080
Scene:                 Confetti
3D Texture used:       Uncompressed - 89.73 MB
                       Res. 255x181x243
Bandwidth:             Avg. Memory Read Bandwidth 721.97 MB/s
                       (on Samsung Galaxy Note 10.1)
Energy Consumption:    Avg. Memory Read energy consumption per frame:
                       DDR2: 4.87 mJ per frame
                       DDR3: 4.01 mJ per frame
```

Physics Simulation

3D textures for collision simulation

- Texture data is big
- 32bpp * 256 * 256 * 256 = 67MB per texture
- We need to compress the data to cope with the hardware memory limitations
- Would be good to have hardware decompression that saves memory and bandwidth

ASTC
(Adaptive Scalable Texture Compression)

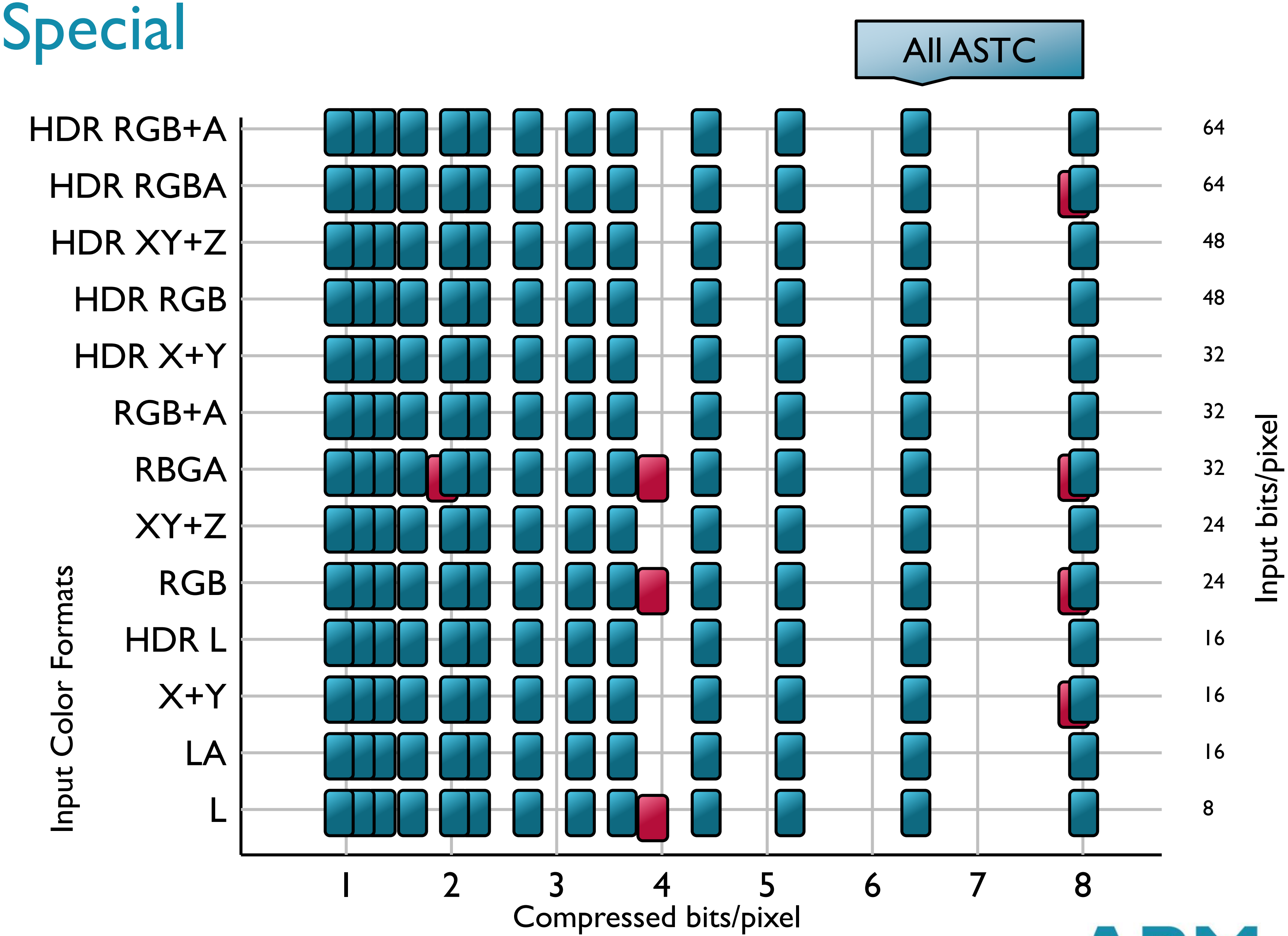
What Makes ASTC Special

- Wide range of bit rates
- Wide range of formats

- Handles sRGB
- Handles HDR

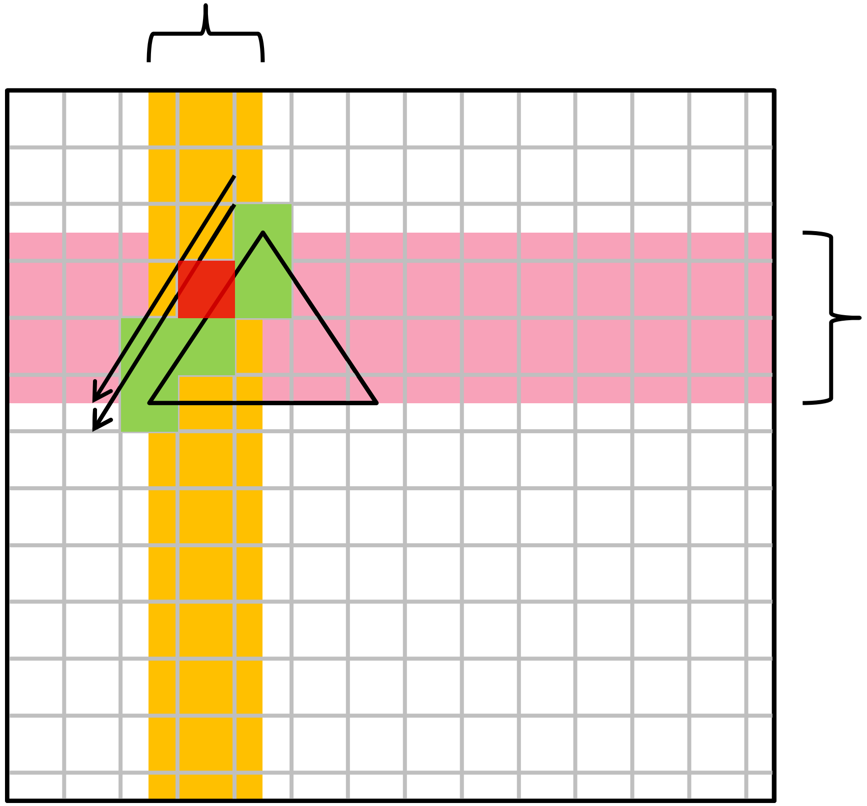
- 3D textures

- Non proprietary



What Makes ASTC Special

- Hardware needs random access
- Texture compression is block based
- Look up a block from the texel coords
- Decompress into local cache
- Sample cached block



What Defines Quality in ASTC?

- Quality decided by 3 factors
  - Precision of data points (bit rate)
  - Number of attempts per tile (limits)
  - Types of error to reject/ignore (priority)

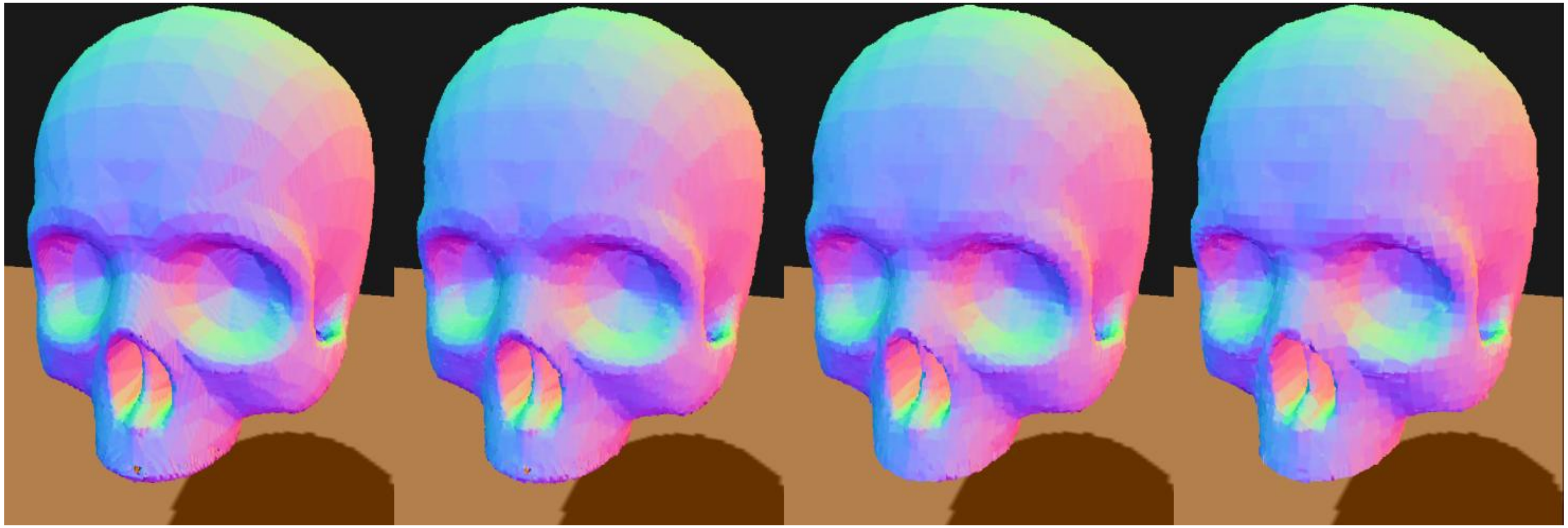
3D Texture Support

- Various block sizes to choose from
- Each compressed block will still occupy 128 bits
- HDR support allows us to store 16 bit half-floats per channel

Block Dimension	Bit Rate (bits per texel)
3x3x3	4.74
4x3x3	3.56
4x4x3	2.67
4x4x4	2.00
5x4x4	1.60
5x5x4	1.28
5x5x5	1.02
6x5x5	0.85
6x6x5	0.71
6x6x6	0.59

ASTC 3D Texture Example

Statistics



- ~90% memory reduction
- ~62% memory bandwidth reduction

	Skull
Texture resolution	180x255x255
Texture Size MB	
Uncompressed	82.62
ASTC 3x3x3	6.12
ASTC 4x4x4	2.63
ASTC 5x5x5	1.32
Memory read bandwidth in MB/s	
Uncompressed	752.18
ASTC 3x3x3	285.78
ASTC 4x4x4	179.43
ASTC 5x5x5	167.90
Energy consumption per frame DDR2 mJ per frame	
Uncompressed	5.08
ASTC 3x3x3	1.93
ASTC 4x4x4	1.21
ASTC 5x5x5	1.13
Energy consumption per frame DDR3 mJ per frame	
Uncompressed	4.17
ASTC 3x3x3	1.59
ASTC 4x4x4	1.00
ASTC 5x5x5	0.93

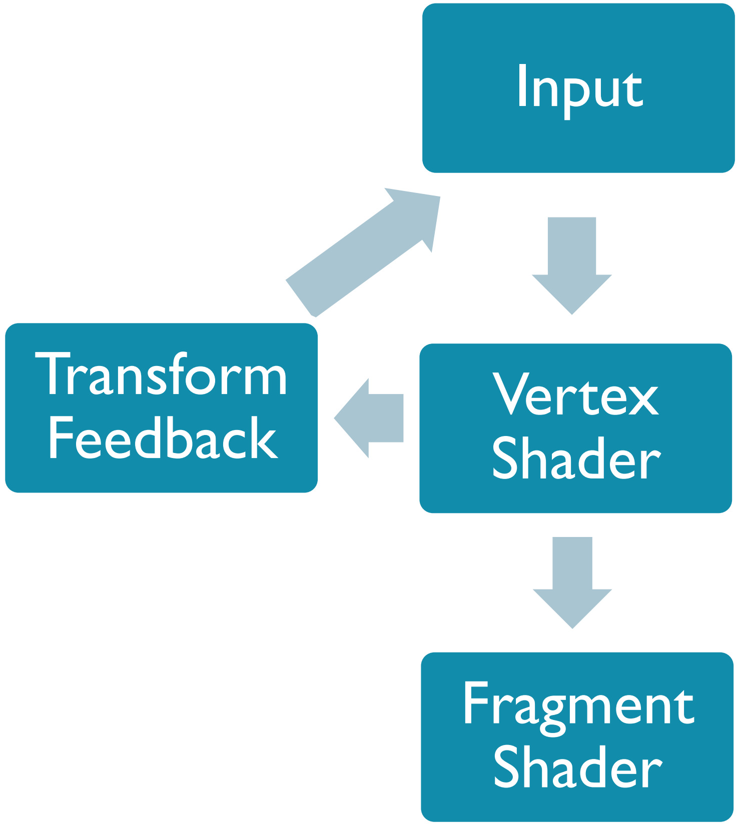
Physics Simulation

Exploiting OpenGL® ES 3.0 for numerical explicit methods

- Explicit methods used in numerical analysis compute the state of a system at a later time using the current state

$$Y(t + \Delta t) = F(Y(t), \Delta t)$$

- The "future" attributes (position and velocity) of a particle can be computed from the current state
- We use Transform Feedback feature from OpenGL ES 3.0 to compute this entirely on GPU

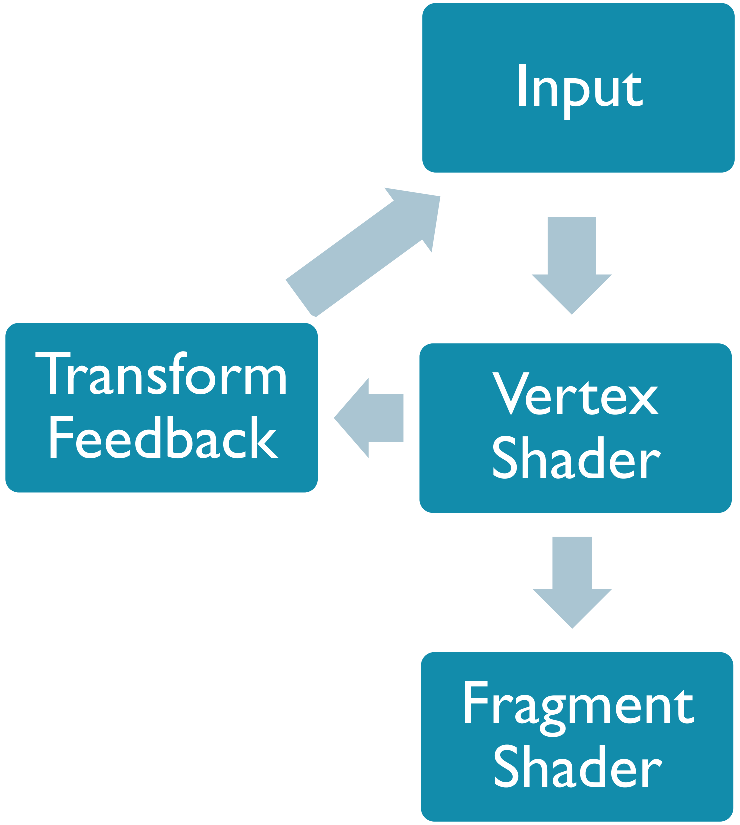


Physics Simulation

Exploiting OpenGL® ES 3.0 for numerical explicit methods

- In the demo:

1) Define 2 buffers as input/output
2) Initialize it with the initial particle's data
3) Define which output attribute that goes from vertex to fragment shader needs to be also saved with Transform Feedback
4) For each frame:
   i. Issue a draw command as GL_POINTS to update the particle's position
   ii. Render the particles with the desired effect.
   iii. Swap input/output buffers



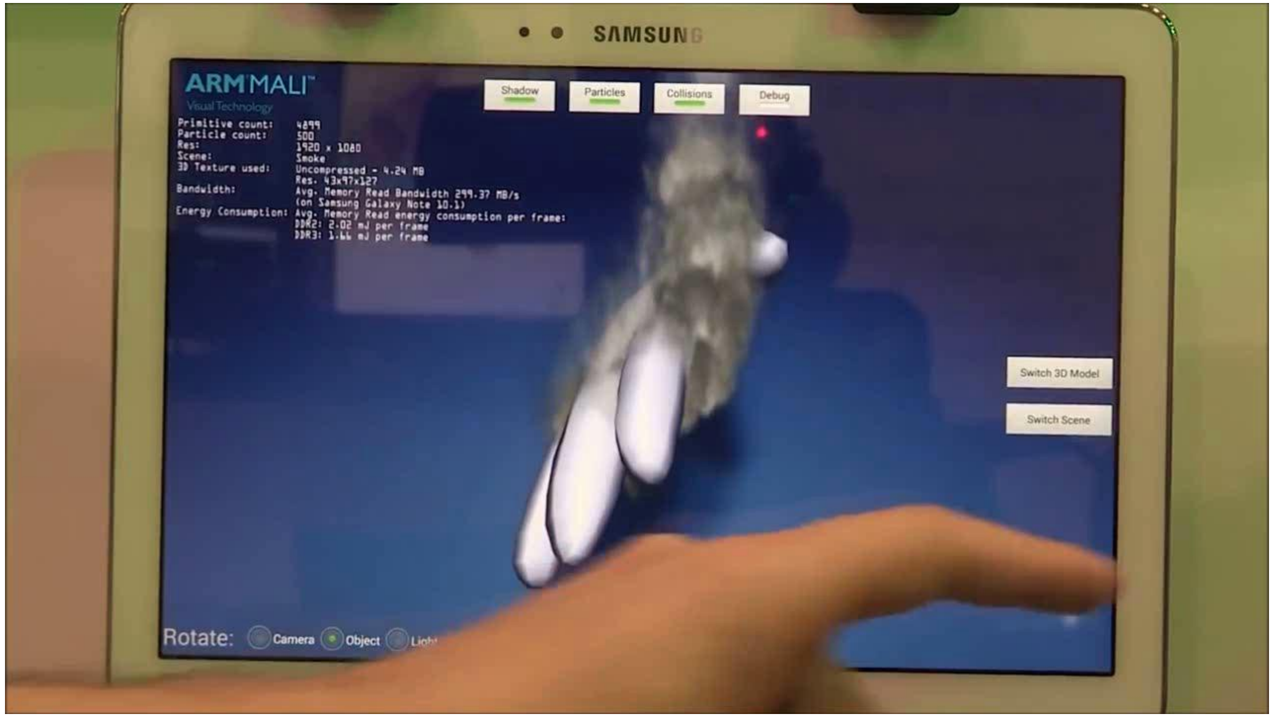
Rendering the Particles I

Soft particles for the smoke effect

- The smoke has been rendered using a noise texture to compute a normal for the lighting and an opacity factor

- Typically, quads that intersect the geometry will cause sharp edges due to Z-Test

- In the demo we implemented soft-particles disabling Z-Test and using the GL_ARM_shader_framebuffer_fetch_depth_stencil to read the value previously stored in the depth buffer

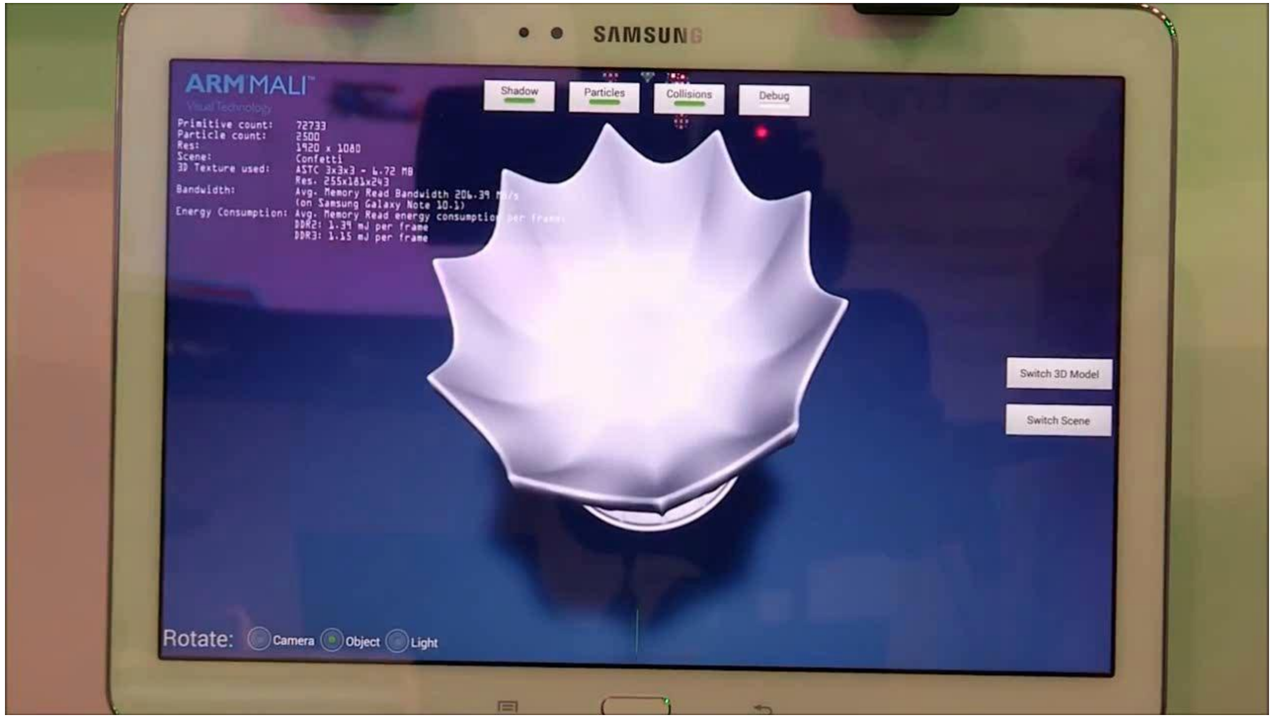


```
Primitive count:      4899
Particle count:       500
Res:                  1920 x 1080
Scene:                Smoke
3D Texture used:      Uncompressed - 4.24 MB
                      Res. 43x97x127
Bandwidth:            Avg. Memory Read Bandwidth 299.37 MB/s
                      (on Samsung Galaxy Note 10.1)
Energy Consumption:   Avg. Memory Read energy consumption per frame:
                      DDR2: 2.02 mJ per frame
                      DDR3: 1.66 mJ per frame
```

Rendering the Particles II

Collision orientation for the confetti effect

- In the confetti scene, we wanted to give a more realistic behaviour orientating the confetti upon collision
- We store the normal of the collision surface in the attributes of the particles
- We compute the TBN matrix and apply it to the unit quad in the plane Z=0
- Use OpenGL® ES 3.0 instancing to improve performance



```
Primitive count:    72733
Particle count:     2500
Res:                1920 x 1080
Scene:              Confetti
3D Texture used:    ASTC 3x3x3 - 6.72 MB
                    Res. 255x181x243
Bandwidth:          Avg. Memory Read Bandwidth 206.39 MB/s
                    (on Samsung Galaxy Note 10.1)
Energy Consumption: Avg. Memory Read energy consumption per frame:
                    DDR2: 1.39 mJ per frame
                    DDR3: 1.15 mJ per frame
```

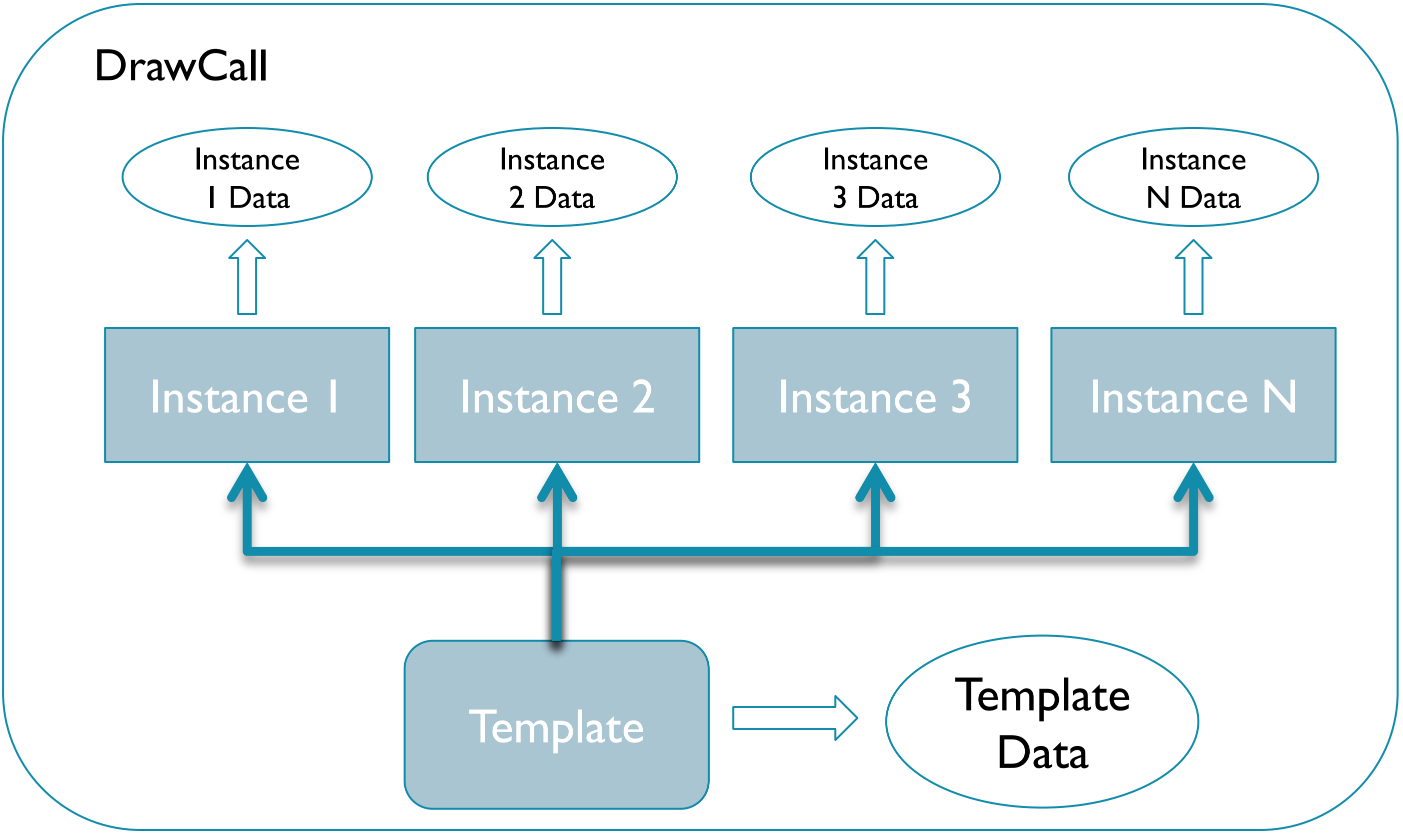
Rendering the Particles III

OpenGL® ES 3.0 instancing for the confetti effect

- Use case: render the same geometry multiple times with different parameters but with a single drawcall

- The confetti is a perfect match for the feature since they are all quads and the different shapes are implemented procedurally

- Instancing allows the user to render multiple instances of a template geometry, each instance will have common and specific parameters (ModelView matrix, materials…)



Why Not Try OpenGL ES 3.0 and ASTC Right Now?

- Command line compressor
  - ASTC Evaluation Codec

- GUI compressor
  - ARM® Mali™ Texture Compression Tool

- Lacking compatible hardware?
  - ARM Mali OpenGL® ES 3.0 Emulator

Mali Developer Center:

MaliDeveloper.arm.com

Thank you

Any questions?

MaliDeveloper.arm.com

community.arm.com