*Diaz, Andrés* *andi16@student.bth.se*
*Lind, Fredrik* *f.flind@hotmail.com*

# 3d3 - Report

## Introduction

We have created a Dx12 program which uses a technique called *bindless* or *unbounded arrays* with the combination of *dynamic indexing*.

Bindless is a fairly new technique which was first tested in GLSL as an extension but was later fully implemented in Dx12 and Vulkan. This technique lifts the constraints of a texture array, which means that its array's range is *unlimited*.

With dynamic indexing, shaders can index into an array without knowing the value of the index at compile time (Microsoft, 2020).

## Implementation Methodology

Every object can be created from the main file where each has its own *PSO*, *vertex buffer*, *constant buffer* and *textures*. They share the same *root signature*.

We decided to test the bindless/dynamic indexing techniques by splitting an animated gif into separate frames, and then we access all the different frames through the descriptor table in the pixel shader. The index of the frame we choose to use is sent from the CPU with a constant buffer.

For benchmarking we used two timestamps. One that keeps track of how long it takes to process a *frame* and one that measures the *draw calls* for each object in the rendered scene.

## Results

### Benchmarking

GPU: RTX 2070 Super
CPU: Intel Core i7 4790K
RAM: DDR3 16 GB, 4000 MHz

At startup we get some high numbers that skew the results if benchmarking is done with a low amount of samples, so these benchmarking results are done with 10 000 samples.

*Diaz, Andrés* *andi16@student.bth.se*
*Lind, Fredrik* *f.flind@hotmail.com*

Average of draw calls from 10 000 samples, in ms, per object. The scene contains six objects

| Object index | Time (ms) | Object description |
|---|---|---|
| 0 | 0.0666626 | big box with dynamic texture |
| 1 | 0.0138024 | small box with dynamic texture |
| 2 | 0.0125902 | small box with dynamic texture |
| 3 | 0.00693093 | big Piedmon(character) with dynamic texture |
| 4 | 0.0346635 | small Piedmon standing on a box |
| 5 | 0.0338773 | small Piedmon standing on a box |

Average benchmark for whole frame, 10 000 samples

**Frame:** 3.80205 ms

# Analysis

Interestingly the fourth object (index nr 3) is drawn much faster than all the other objects. We are not sure why this is the case, because the fourth object has more polygons than what the boxes have and is using the same textures as they do.

Another thing we noticed is that our first run with 105 different HD images took more than 8GB of RAM to run. We felt that this was way too much RAM usage for such a small program, and therefore we lowered the quality of the images which significantly changed the RAM usage.

# Conclusions

We noted that unbounded arrays can easily take up a lot of RAM. The images' size and the amount of images can have a big impact on the amount of RAM usage the program needs.

We have been working together through the whole project which means that our distribution is the same. The grade we are aiming for is E and we think that we have reached this goal by having created a working Dx12 program with a technique which is unique to the Dx12 and Vulkan API and which can therefore not be used in Dx11 (and earlier APIs).

*Diaz, Andrés* *andi16@student.bth.se*
*Lind, Fredrik* *f.flind@hotmail.com*

# Bibliography

[1] Microsoft, 2020. *Dynamic Indexing Using HLSL 5.1 - Win32 Apps*. [online] Available at: https://docs.microsoft.com/en-us/windows/win32/direct3d12/dynamic-indexing-using-hlsl-5-1 [Accessed 23 March 2020].