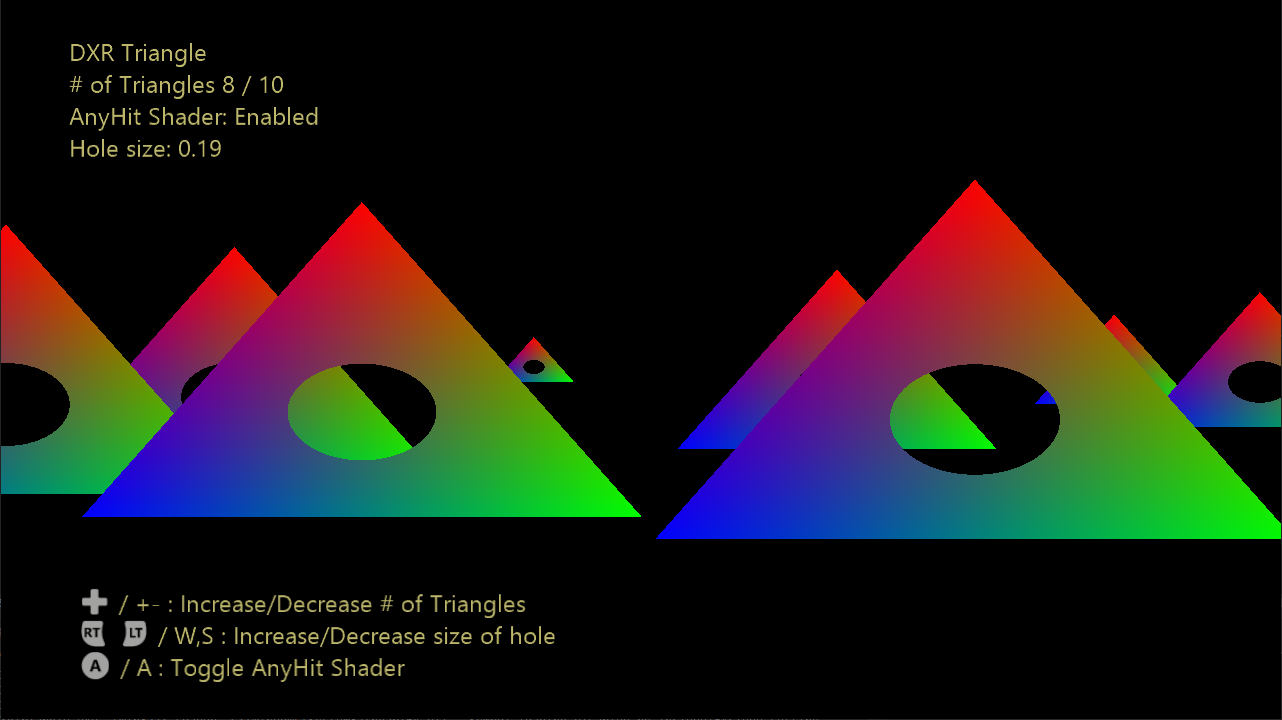


DXR Triangle Sample

*This sample is compatible with the Microsoft Game Development Kit (June 2020) and Windows 10 (Version 1903) May 2019 Update*

# Description

This sample demonstrates the basic use of the DXR API. It creates a Raytracing Pipeline State Object (RTPSO), Shader Tables and Top-Level / Bottom-Level Acceleration Structures. It also demonstrates the use of Ray Generation, Closest-Hit, Any-Hit and Miss Shaders as well as Instancing of Bottom-Level Acceleration Structures.



# Building the sample

If using Project Scarlett, set the active solution platform to Gaming.Xbox.Scarlett.x64.

If using PC with appropriate hardware and Windows 10 release, set the active solution platform to Gaming.Deskop.x64.

This sample does not support Xbox One.

*For more information, see* Running samples*, in the GDK documentation.*

# Using the sample

Taking the classic ‘SimpleTriangle’ and bringing it into the raytracing era, a configurable number of triangles slide across the screen. By pressing A on the keyboard/gamepad the use of Any-Hit shaders can be toggled on and off – punching a hole through each triangle to reveal the triangles behind it.

# Controls

|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Keyboard |
| Toggle Any-Hit Shader | A | A |
| Increase # of Triangles | Directional Pad Up | + |
| Decrease # of Triangles | Directional Bad Down | - |
| Increase size of the hole | Right Trigger | W |
| Decrease size of the hole | Left Trigger | S |
| Exit | View Button | Escape |

# Implementation notes

The sample begins by creating a Raytracing Pipeline State Object (RTPSO) using the ‘CreateStateObject’ API. The RTPSO contains one Hit-Group which consists of a Closest-Hit Shader and an Any-Hit Shader. Other configuration properties such as recursion level and ray-payload size are also specified.

The Bottom-Level Acceleration Structure (BLAS) consists of a single triangle and is instanced between 1 and 10 times into the Top-Level Acceleration Structure (TLAS) depending on the currently selected number of triangles. Each instance is given its own position and size in the scene. By default, both the TLAS and BLAS are built every frame in order that they appear in PIX captures should you wish to take one. However, in a shipping title you would typically only want to rebuild the TLAS every frame and reuse BLAS’s for long periods of time (potentially forever).

The Ray Generation shader sets up a simple Orthographic Projection looking along the Z-axis and fires one ray per thread launched in the DispatchRays calls.

The Miss Shader is invoked for every ray that misses every triangle and writes black to the UAV (this is in place of pre-clearing the UAV to black before the DispatchRays call).

The Closest-Hit shader calculates the barycentric coordinates and uses those as the final colour for the pixel.

The Any-Hit shader calculates a simple distance to the centre of the triangle (0.333, 0.333, 0.333) in barycentric coordinate space and requests the hit be ignored if the distance is less than some tolerance. The ray is then allowed to pass through the triangle and potentially hit other triangles further along the z-axis.

# Update history

11/1/2019 – Sample creation.

# Privacy Statement

When compiling and running a sample, the file name of the sample executable will be sent to Microsoft to help track sample usage. To opt-out of this data collection, you can remove the block of code in Main.cpp labeled “Sample Usage Telemetry”.

For more information about Microsoft’s privacy policies in general, see the [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement/).