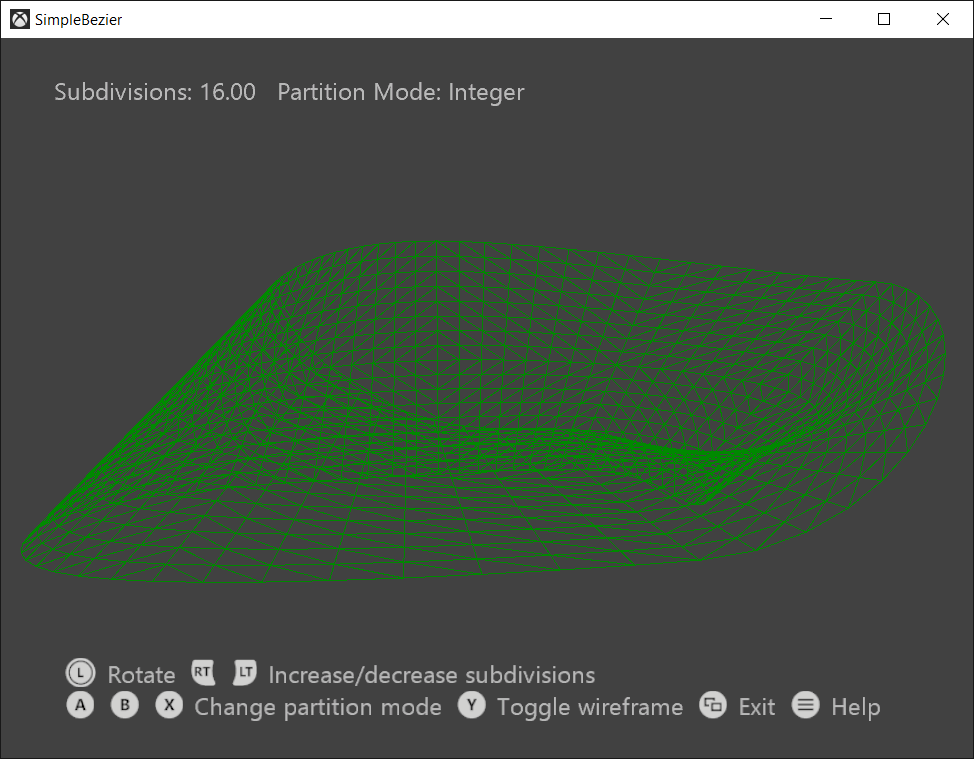
Simple Bezier Sample

*This sample is compatible with the Windows 10 Anniversary Update SDK (14393)*

# Description

This sample demonstrates how to create hull and domain shaders to draw a tessellated Bezier surface representing a Mobius strip.



# Using the sample

This sample uses the following controls.

|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Keyboard |
| Shaded/wireframe rendering | Y button | W key |
| Select tessellation method:   * Integer * Fractional even * Fractional odd | X button  A button  B button | 1 key  2 key  3 key |
| Decrease/increase number of patch divisions <4, 16> | Hold left/right trigger | Hold up/down arrow keys |
| Rotate camera left/right | Move left thumbstick left/right | Left/right arrow keys |
| Show controller help | Menu button | F1 key |
| Exit | View button | Escape key |

# Implementation notes

# Input geometry consists of four patches with 16 control points each, all stored in a vertex buffer. A simple vertex shader passes the control points straight to the hull shader. The hull shader drives the fixed function tessellator stage through a tessellation factor from a constant buffer, both of which then pass the control points and the UVW to the domain shader. The domain shader is run once per vertex, and calculates the final vertex’s position and attributes. The vertex's position is calculated by using a Bernstein polynomial; the normal is calculated as the cross product of the U and V derivatives. The pixel shader performs N dot L lighting to draw a shaded Mobius strip.

## Hardware Feature Level Requirement

The hardware tessellation feature of DirectX 11 requires [Direct3D Hardware Feature Level 11.0](https://blogs.msdn.microsoft.com/chuckw/2012/06/20/direct3d-feature-levels/) or better hardware. This sample will not run on 9.x or 10.x Direct3D Feature Level hardware.

# Update history

Initial release April 2018.