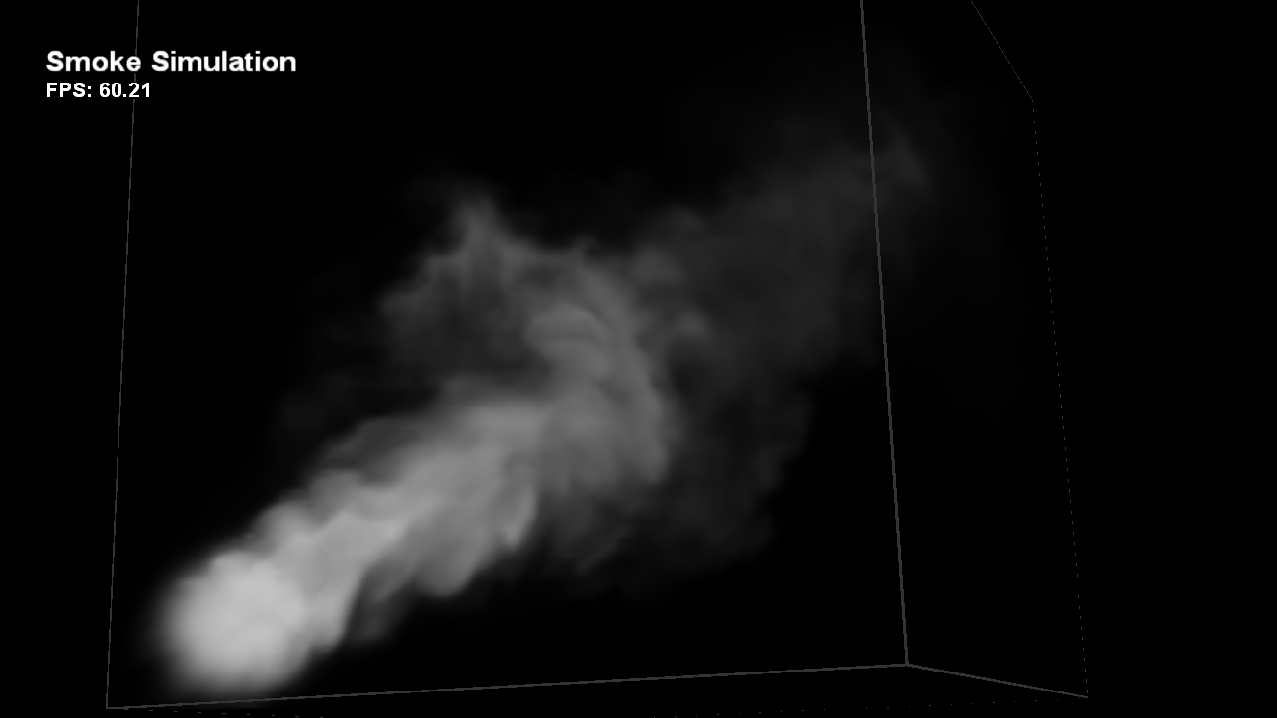


Smoke Simulation

*This sample is compatible with the Microsoft Game Development Kit (June 2020)*

# Description

This sample demonstrates how to use Compute Shader 6.0 and 3D textures to implement basic 3D Navier-Stokes flow simulation. The sample also demonstrates how to render volumetric data using a simple ray-marching algorithm.

**

# Building the sample

If using an Xbox One devkit, set the active solution platform to Gaming.Xbox.XboxOne.x64.

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

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

# Using the sample

This sample uses the following controls.

|  |  |
| --- | --- |
| Action | Gamepad |
| Exit the sample. | Select |
| Rotate Camera | Right Stick |
| Rotate Emitter | Left Stick |
| Move Emitter | Left/Right Trigger |
| Reset Emitter Direction | X Button |
| Toggle Simulation Pause | A Button |
| Reset Simulation | B Button |

# Implementation notes

**Simulation Technique**

The detailed information about the simulation technique used in this sample can be found in GPU Gems Chapter 38 Fast Fluid Dynamics Simulation on the GPU. While that article explains and applies the simulation technique to a 2D fluid, this sample extends it to 3D data. In this sample the simulation runs on a 128x128x128 grid, the state of which is stored in 3D textures.

Using Compute Shader 5.0 we can directly manipulate all slices of a 3D texture at once. This results in higher efficiency than an alternative strategy of employing a Geometry Shader, which can be used to specify which 3D texture slice to update.

**Rendering Technique**

The result from the simulation is a 3D velocity field, which represents the status of the fluid at a certain simulation step. Since we cannot directly see velocity, we need to have trace particles we can see that are carried around by the velocity field. So, in addition to all the 3D textures required by the simulation itself we also have a 3D texture which stores these particles. This 3D texture is what we visualize in the rendering phase.

When rendering, we render the front faces of a cube. In the pixel shader, we cast a ray from the eye position to the current point on the cube and sample the 3D texture mentioned in the last paragraph along the ray at a fixed interval, while accumulating color and opacity.

The sample also demonstrated a simple yet effective acceleration technique for the ray-casting algorithm. During simulation, a 1/8 sized 3D texture is generated using parallel reduction. When rendering, instead of directly sampling the original 3D texture at a fixed interval and possibly waste lots of time sampling empty spaces, we first sample this 1/8 sized 3D texture and skip chunks in space if the value sampled is smaller than a threshold. This technique is especially effective when the data stored in the 3D texture is sparse, like our smoke here.

# Update history

April 2019 – Port to new template from legacy Xbox Sample Framework.

# 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/).