Sync Point Shader Arguments

Status: Public. Work in progress. Seeking feedback.

Summary

Being able to obtain the status of a future sync point during shader execution will enable users to implement carry-select type optimizations to reduce latency and improve frame completion.

For example, a user could conditionally bind one texture or another depending on the status of a sync point, using a newer texture if it happens to be complete. *Note: Conditional texture binding would require the ARB_bindless_texture OpenGL extension.*

Assuming Future Sync Points and Shader Storage Buffer Objects exist, we should only need to add one new OpenGL function to support this feature: glSampleSyncPointCompletion.

Overview of Approach

To support reading sync points from shaders, we need to be able to do the following:

1. Set an array of syncpoint locations from client code.
2. Sample the syncpoints provided at execution time and convert them into boolean values indicating the corresponding sync point's completion.
3. Make the boolean completion values visible to a shader.

Details

We can built this feature using two Shader Storage Buffer Objects, one representing the array of syncpoints (syncpointLocations) and another representing the array of completion values (syncpointCompletions).

The user populates the values in syncpointLocations and calls the following function, which inserts a small program into the command stream that checks the status of each sync point in syncpointLocations and writes the status to the corresponding element in syncpointCompletions.

```
glSampleSyncPointCompletion(syncpointLocations, syncpointCompletions, count);
```

Potential Implementations

Two potential implementations are described below, one using a push mechanism and the other a pull mechanism. The push mechanism incurs a cost every time a relevant sync-point retires, but makes the time of sampling very quick. The pull mechanism incurs no cost at sync point retirement, but requires querying the status of each sync point at the sample time.

Push using Asynchronous Shader Input

Details TBD. See doc for Asynchronous Shader Input.

Pull by Querying Sync Points Synchronously

Details TBD.

Alternatives

1. Use Shader Storage Buffer Objects and assign a memory location to represent a particular completion point. If 0, the point has not been reached. if 1, it has.
    a. Pro: Doesn't require a new API.
    b. Con: A bit more manual. Need to manage memory location lifetimes.
    c. Con: Requires a unique memory location for every sync point we *might* care about.
    d. Unknown: Does setting a memory location allow a shader to ensure visibility of it's results to other queues?
2. Querying the state of a syncpoint synchronously from a shader.
    a. Con: Calls would be blocking and waste execution cycles.
    b. Con: Calling from a fragment shader doesn't make sense; why would anyone want to make a different decision pixel to pixel?
    c. Con: Calling from a vertex shader might make sense, but not if that vertex shader is split across multiple execution units.