Student Name: Joshua Evins
Project: Fluids and Simulation

Capstone Weekly Project Summary

Week 1	Project Status: N/A (initial meeting)
Tasks Completed/New Functionality	• Set up work environment and began documentation
Comments	This was not a full week.

Week 2	Project Status: Yellow
Tasks Completed/New Functionality	• Researched mesh deformation (found what I needed) • Worked out variables required to implement the deformations • Created the model mesh • Updated (corrected) the window (Qt) setup in Engine • Restructured build settings and directories (our old build structure had issues with .dll management) ○ This segment took the largest amount of time, but was definitely worth it. • Wrote a pilot blog post for this project • Wrote a blog post to follow-up to the pilot
Comments	This week focused more on bringing Engine up to par than anything.

Week 3	Project Status: Green

Tasks Completed/New Functionality	• Researched Uniform Buffer Objects
	  ○ How to define one in the shader
	  ○ How to mirror that structure in the application
	  C++ data packing (N4 packing)
	  Using a layout explicitly in GLSL (using layout(std140) is very close to N4 packing)
	  ○ How to send data from the application to GL
	  GL_UNIFORM_BUFFER
	  Uniform Block Indices
	  Uniform Binding Points
	  Buffer offset constraints:
	  GL_UNIFORM_BUFFER_OFFSET_ALIGNMENT is a compile-time constant that is used to access a per-machine value. Attempting to bind a Uniform Block to a buffer range that starts at an offset that is not a multiple of the per-machine value will result in a failed binding. In other words, offset%perMachineValue MUST be zero for the bind to succeed. (I originally used the compile-time constant rather than the per-machine value. That gave me problems.)
	• Implemented Uniform Buffer Object capability to Engine
	  ○ Created a UniformBlockInfo struct to serve as an interface for reading data from / writing data to the buffer range bound by a uniform block. Visible from the application.
	  ○ Renderer now manages creation of, offsets into, etc. of GL_UNIFORM_BUFFERs
	  ○ Renderable now holds UniformBlockInfos
	  ○ Renderable now has a method to create UniformBlockInfos visible to the application author.
	  ○ Uniform Block gotchas

	If a Uniform Block definition is used as the type of an array then each individual element of that array will need to bound separately.
	Defining your "object" as a struct in the shader and defining a single Uniform Block that has an array of that struct only requires one bind.
	Rephrasal of the above: a uniform array of ripples is difficult to manage, but a uniform that contains an array of ripples is easily expandable (make a wrapper for the collection)
	• (Re)Implemented Ripples
	  ○ RippleEffect struct
	  ○ RippleEffects Uniform Block
	• Implemented Waves
	  ○ Developed a function to define a wave
	  ○ WaveEffect struct
	  ○ WaveEffects Uniform Block
	  ○ (Had an issue with my UBO implementation in Engine, fixed it)
	  ○ Implemented function in shader
	• Added a "crosshair"
	• Implemented user-generated effects
	  ○ Crosshair lets the user know where the effect will start
	  ○ Waves travel in the direction the camera is looking

Comments	This week took a large amount of digging through poorly-written documentation and vague references. The development for this week took place almost exclusively inside Engine.

Week 4	Project Status: Green

Tasks Completed/New Functionality	• Researched Smoothed Particle Hydrodynamics
	  ○ Bad news then good news
	  ○ Bad: the speed comes from a graphics-card solution that avoids the CPU and requires much more knowledge about using the graphics card than just a standard rasterize pipeline
	  ○ Good: I may be able to do it
	• Research particle rendering for fluids
	  ○ Found some math with umlauted letters that were on top of other letters, which is definitely outside of my expertise
	  ○ Found the master slideshow from GDC which described that math as textures (which helped very much)
	  ○ Also requires special render passes
	• Researched constraints
	  ○ Normal wall-like constraints are very easy to implement into SPH, and the constraints on geometry seem to use the same methods as SPH itself and rendering SPH, so this should not be a problem.

Comments	Lesson learned: Academic research papers are not a very good place to learn about a topic unless you already have formal training in that topic. Note from the future: I was wrong about the constraints for geometry using the same technique as wall constraints.

Week 5	Project Status: Yellow

Student Name: Joshua Evins
Project: Fluids and Simulation

Tasks Completed/New Functionality	• More SPH research
	  ○ Bad news: SPH needs to be done entirely the graphics card, which requires openCL or CUDA
	  ○ Good news: AMD did an openCL demo specifically for SPH, which means that for the first time in all of my capstone I have a tutorial for exactly what I am doing
	  ○ "I may be able to do it" is now "I can do it" (the only question now is time)
	• Researched openCL
	  ○ Made to function well with openGL
	  ○ Not typical C code
	• Researched and implemented Frame Buffer objects
	  ○ Implementing a custom FBO is a relatively short process, it's just full of intuitive mistakes and very, very misleading documentation (I would even go so far as to say the documentation was in error).
	• Saved data textures to disk, as promised
	  ○ There are two main methods of pulling a texture out of openGL buffers. One of them works for what I was doing and one of them does not. Once again the lack of general tutorials and documentation on what I was doing threw me off, but I did finish the deliverable.

Comments	If I had not promised data textures for this week's deliverable I would have started learning openCL more.

Week 6	Project Status: Green

Tasks Completed/New Functionality	• Researched OpenCL implementation
	  ○ The C++ bindings are almost not even worth the effort
	  ○ Learned new terminology "function object"
	  ○ Setup is very difficult. There are several different objects that need to be queried and/or initialized based on other objects that must be queried and/or initialized, etc.
	  ○ Demos are scarce
	• Implemented OpenCL
	  ○ Added a new rendering item in Engine that assigns vertex attributes from multiple buffers instead of just one
	• This demo was pieced together from an AMD demo that used the c bindings, a Siggraph Asia slide-deck about CL GL interop, blog posts from a Norwegian programmer in his blog "The Big Blob" who uses a UNIX system, the OpenCL API, a few Stack Overflow posts, and a whole bunch of trial and error.

Comments	

Week 7	Project Status: Green
Tasks Completed/New Functionality	• Watched AMD tutorial several times • Started to develop pieces in the video • Realized that developing this from the ground up in a single week would not be possible • Searched for and eventually found the source code for the tutorial • The demo was made with DirectX, not OpenGL, so I did not bother to get their demo running • The demo did have some functional code

	• The demo has some drawbacks
	  ○ Dated: OpenCL is more strict about syntax than it was when this was written, so I had to swim in esoteric errors for a couple of days before I had the kernels compiling
	  ○ Not scalable: changing the number of particles at all causes OpenCL to error
	  ○ Not efficient or easy to follow: all of the parameters are passed every single frame rather than once after kernel creation (only parameters that change needs to be passed in each frame)
	  ○ Messy, redundant, and pointless in some areas:
	  C++ global variables drive almost all of the functionality
	  The "dot" function was aliased by preprocessor macro to "DOT"
	  There was a preprocessor macro involving "if 1" (aka "always") and an "else" section (in this case meaning "never")
	  Devoid of useful comments, reasoning behind design choices, and in some cases even sensible identifier names
	• Kernels choose their own local size (and it is not always the same value)
	  ○ The local size must evenly divide into the global size so that there are no fractured work-groups, but the kernels are choosing maximum values small enough that I am forced to run at ¼ of the desired local size (in my situation larger local sizes are faster than smaller local sizes)

Comments	This week's deliverable was a relatively functional fluid simulation in my own engine. I did meet that deliverable, but it was buggy and not enough of it was my own to call demo three complete.

Week 8	Project Status: Yellow

Tasks Completed/New Functionality	• Demo Kernel File
	  ○ Preprocessor definitions
	  Almost all of them are redundant, useless, confusing or hard-coded values that make scaling difficult.
	  ○ Non-physics kernels
	  All of these were straightforward and did exactly what I expected
	  Most of them were not implemented intuitively. The unintuitive functionality seemed to be aimed toward reducing running time.
	  In multiple cases their focus on low running time made changing the number of particles or neighbors difficult.
	  ○ Physics kernels
	  Still working through these calculations
	  Easier to follow than the pure mathematical representations
	• Radix Sort
	  ○ Sorting in a massively parallel environment is a foreign concept coming from iterative algorithms
	  ○ Wrote a functional parallel algorithm, but it was slow (>0.4s)
	  ○ Currently designing a more efficient algorithm that makes use of groupings of threads instead of looping through all previous values
	  ○ This took about 14 of my 19.5 hours

Comments	This week I sifted through the pieces of the demo I was using to determine how they work. I was not surprised by any of them other than the fact that there are several places that parameters are unused (no matter what you pass in as the range value of the myRandom function the actual maximum return value will always be PARTICLE_COUNT-1, range is never used).

Week 9	Project Status: Green

Tasks Completed/New Functionality	• Looked for a pre-existing GPU sort
	  ○ There are very few that do not require a large library
	  ○ Out of those few, two were compatible with my platform
	  ○ Of those two, only one was compatible with my setup and helper library. Unfortunately, that one was not applicable to the format of the data I had to sort.
	  ○ Realized at that point that I had to improve my own version drastically
	• Radix Sort
	  ○ Optimized reading/writing to global memory
	  ○ Restructured the last stage kernel to remove a costly loop
	  ○ Separated the first kernel into three kernels to eliminate the need to single out threads from each work-group for special work
	  ○ Now runs 3 times faster than qsort
	• Also optimized several of the other kernels
	  ○ All optimizations but one were done by using more compact reading and writing of global memory
	  ○ The last optimization was just an early exit if there was nothing left to do

Comments	

Week 10	Project Status: N/A (presentation week)

Tasks Completed/New Functionality	• Prepared Mesh Deformation for presentation
	  ○ Removed duplicate meshes
	  ○ Defined a comfortable camera speed
	  ○ Changed rendering back to simple lines instead of using the geometry shader driven “point to camera-oriented quad” technique I developed
	  ○ Increased the maximum number of simultaneous ripples to 50 from 20
	  ○ Removed initial effects
	  ○ Repositioned camera
	• Prepared NBody for presentation
	  ○ Colored the particles by velocity
	  ○ Increased point size to make the particles more visible on the projector
	  ○ Defined a comfortable camera speed
	• Prepared Smoothed Particle Hydrodynamics for presentation
	  ○ Added a source and drain to create flow instead of the fluid just sitting there (this was a very good decision)
	  ○ Colored the source particles green to give them more contrast so they are more noticeable
	  ○ Increased point size to make the particles more visible on the projector
	• Asked some of the staff for questions to drive the presentation’s contents and

	better prepare for questions during the defense
	• Drafted a presentation that took 13 minutes. Reduced time by:
	  ○ Including fewer implementation details on the first two demos
	  ○ Removing some content from the presentation that is also in the project summary (may require more questions, but those are better than hitting the 9-minute cutoff)
	  ○ Running through the presentation twice with an audience to get feedback on what pieces could be removed
	  ○ Repeatedly rehearsing sections to find what needed to stay in the presentation

Comments	I need more than 9 minutes for this presentation.