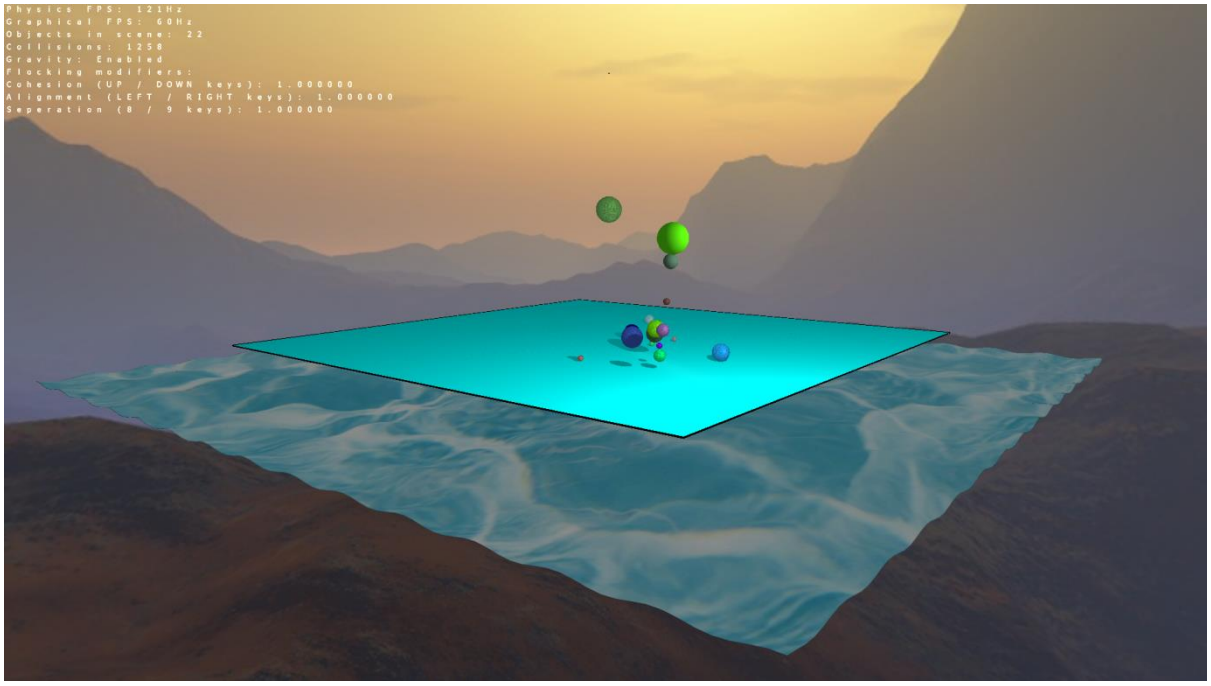
Game Technology Coursework Overview



Control Scheme

W,A,S,D and the mouse controls camera position.
O toggles “Slow-Mo” for the physics engine.
1 will launch a sphere in the direction the camera is looking.
Up, down, left and right arrow and 8 and 9 control the weightings of the flocking system.
G will disable gravity, and enable the “Gravity well” for the sphere placed in the centre.
P toggles the wireframe draw mode.

Part One – Basic Physics Engine

The engine uses Semi-Implicit Euler integration to achieve real-looking physics simulations, which are able to model sphere-sphere collisions and sphere-plane collisions. The engine is threaded separately from the rest of the game logic, and uses mutex locking to prevent access-control issues. The physics updates are limited to 120Hz, and the graphics to 60Hz. The engine also maintains a running total of collisions modelled since the start, along with a count of the objects present in the scene.

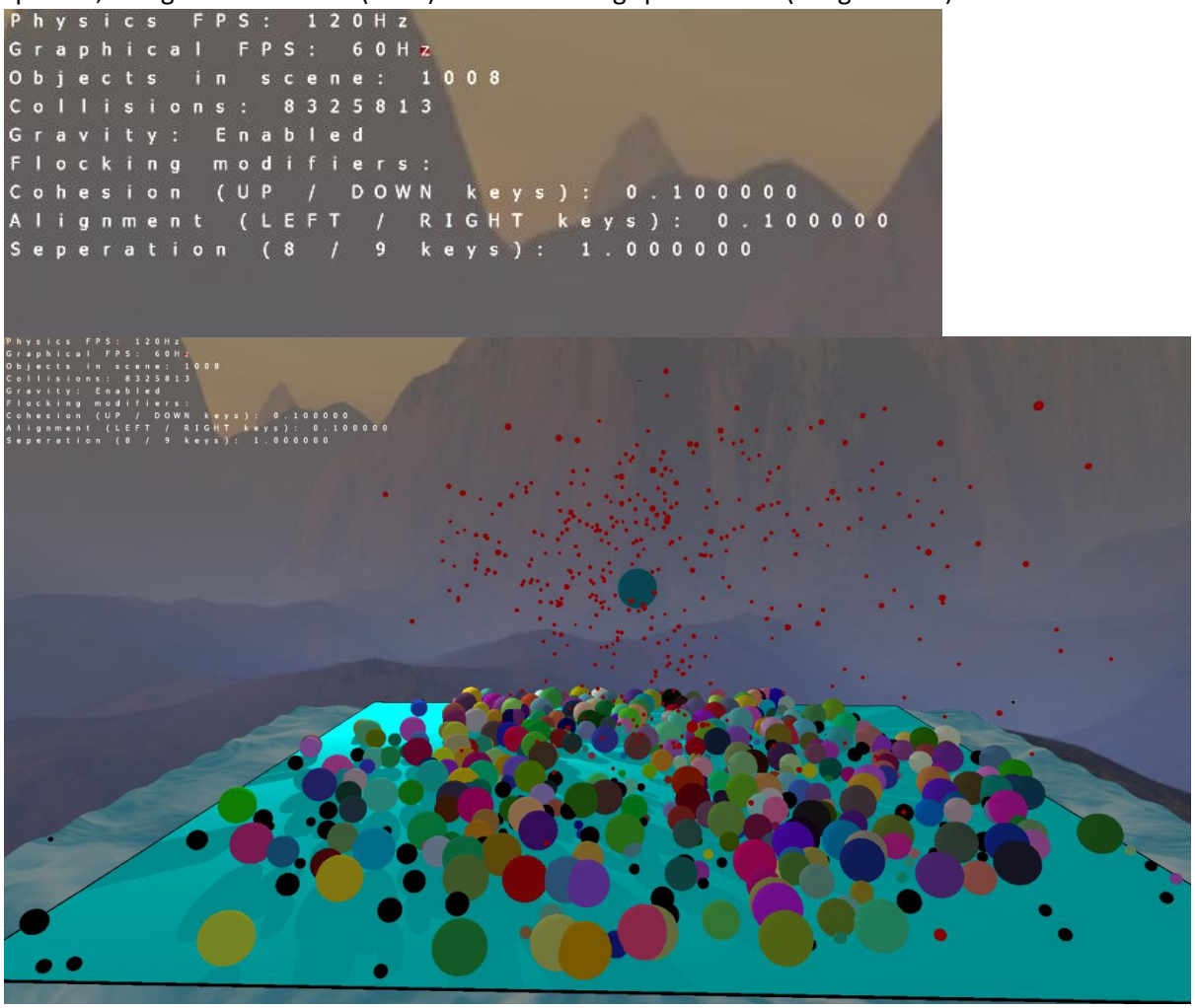
Spheres can be fired from the camera’s point of view and are created with (mostly) random colours and in a range of sizes, with which the collision volume will scale to properly represent the size of the spheres.

Spheres can and will be set to a ‘resting’ state when they cease to move, and will change colour to black to represent this visually, in this state the sphere will not be tested for collisions with the plane to reduce the computational overhead. Spheres can still collide with other spheres when at rest, and

doing so will return the sphere to its normal colour and normal collision testing (until it next goes to rest again).

Part Two – Enhanced Physics Engine

The first and perhaps most important aspect of the 'Enhanced' engine features is the Broad-Phase collision detection algorithm. Using world space partitioning, a large increase in performance can be achieved – testing has been able to show that the engine can handle 1000+ active and colliding spheres, along with a number (100+) of non-colliding spheres too. (image below)

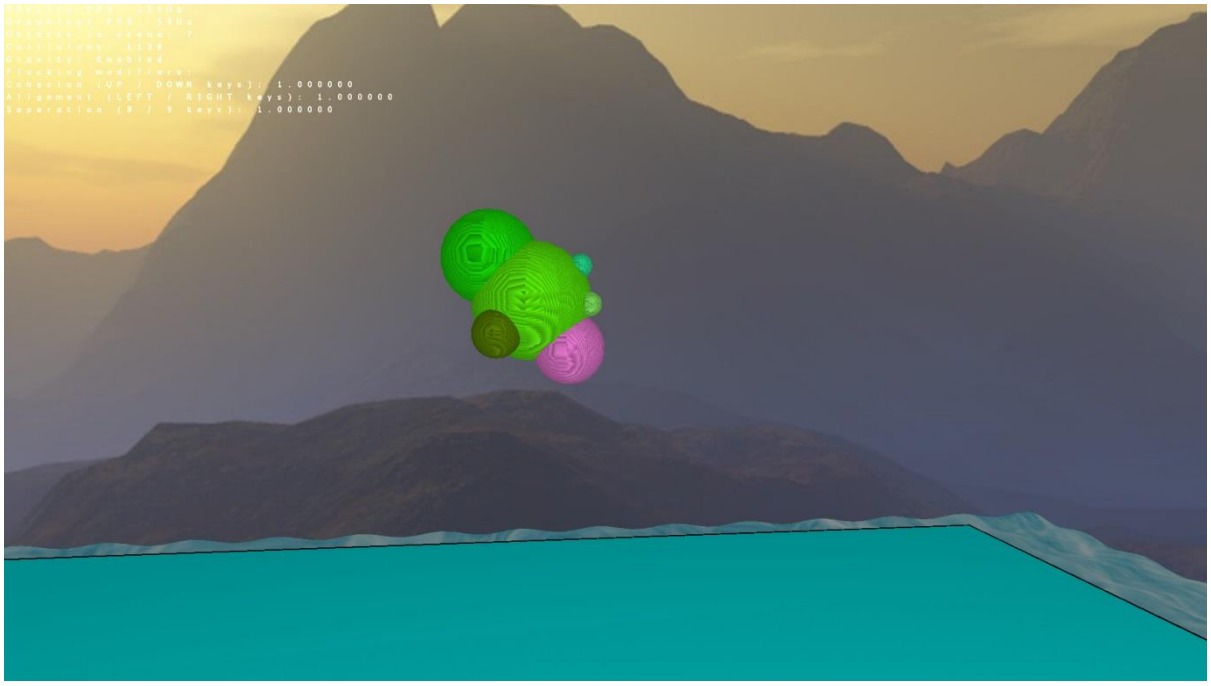


The red spheres seen in the image are flocking entities, these entities have tweak-able weightings to determine the cohesion, separation and alignment preferences of the algorithm. It still represents an unrefined system as 'true' flocked entities were hard to replicate with the algorithm employed.

The large water textured, and slightly deformed quad beneath the 'floor' is an example of CUDA/OGL interoperation, in which a sine-wave function is used to create a wave-like effect on an otherwise flat quad.

The flocked entities are triggered by first disabling gravity with the 'G' key, then firing spheres at the anchored sphere in the middle of the scene. The anchored sphere, when gravity is disabled, mimics a gravity well and will attract spheres towards its centre. When spheres collide with the anchored

sphere, instead of the usual bouncing Newtonian response, the sphere is sucked in, and ‘absorbed. This will increase the size, and gravity well size of the anchored sphere. Once the anchored sphere has absorbed enough spheres, it will explode, returning to normal size, enabling gravity and ejecting anywhere between 1 and 10 spheres in random directions. These spheres will have collisions simulated in the same way as any other sphere spawned in the scene (except the anchored sphere which cannot move). This ‘explosion’ also triggers the release of the flocked entities.



Coursework Video

http://youtu.be/ZQ4WuHxABlA