Critter Optimisations

Problem: every critter is checking collision with every other critter

The part of the critter’s code that is primarily responsible for dragging down the potential frame rate of the program is that every critter object is constantly checking for collision against every other critter object, as well as constantly checking for collision against the destroyer object. This can pass by unnoticed when working with a low number of critters but as soon as the number begins to be significant then the number of collisions being checked begins to pile up.

Solution: Spatial hashing, reduce number of critters it is checking collisions with. Partitioning

The number of calculations done per frame can be reduced by using an optimisation technique known as spatial hashing.

Notably before optimisation the collisions of every object are calculated using O(n^n) notation after which they are calculated using O(n) notation when rebuilding the hash table every frame.

Problem: checks every critter against screen bounds

Solution: Only check the critters in the border bucket

Problem: square root distance check.

Solution:

Problem: Loading texture a lot

Solution: Load texture once

<https://www.cs.ucf.edu/~jmesit/publications/scsc%202005.pdf>