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 game objects 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

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

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

Calendar

Description automatically generated

Problem: checks every critter against screen bounds

On the topic of using spatial hashing to reduce the number of collisions being calculated, another part of the project that is dragging down the frame count is every critter checking if their position is past the borders of the screen and if so, flip the velocity based on which border they are past. This check is done every frame on every critter and can lead to serious performance issues.

Solution: Only check the critters in the border bucket

As with the collision optimisation, this area of code can be improved by using spatial hashing to get every critter’s position and only run code depending on what area they are in. In this case the check to see if a critter is past the border of the screen would only be done on critters that are in the bordering area of the application.

Problem: square root distance check.

Solution:

Problem: Loading texture a lot

Solution: Load texture once