Design Game Optimisation Brief

Optimisation 1: Object Pool

One performance update that I identified that would be helpful within the original design game, is using an object pool to handle the critter creation and destruction. I chose an object pool to handle the critters as I could see they were being created and destroyed repeatedly and by using an object pool it would allow the game to pre-allocate a pool of critters and reuse them as needed. Once the object pool is implemented, whenever the game would need a new critter, instead of dynamically allocating memory, it retrieves a critter from the pool, initialises it and flags it as in use. When a critter is destroyed it would be sent back to the pool, marked as available again and then would be ready to be reused. I anticipate that in implementing the object pool into the game, it would reduce the overhead caused when critters are created and destroyed. This could also help CPU usage in the game, as the CPU would be able to focus more on the game’s logic and rendering, rather than managing memory. I believe by implementing the object pool, it would also reduce memory fragmentation over time, as there would be a block of memory allocated when the game starts, which is reused. This would result in better performance and smoother gameplay (improved runtime).

Optimisation 2: Hash Table

The second optimisation I identified that could be used in the original design game is a hash table to manage the texture loading and unloading of the critters. I chose to use a hash table to manage the textures the critters use, as I could see that textures for them were being loaded and unloaded frequently. By using a hash table, it would allow the game to load in the critter texture once and reuse that loaded texture for any new critters that are spawned, rather than unloading the critter texture and then loading it again whenever a critter is destroyed and recreated. I predict that by implementing a hash table that memory usage will be more optimised and the risk of a memory leak is reduced. This optimisation once implemented, should result in improved game performance, faster loading times and smoother gameplay (improved runtime).

I suggest that both optimisations be implemented according to the class diagram on page two of this document. My recommendation is to make a template class for Object Pool (named ObjectPool) and a class for the Hash Table (named TextureManager).

A screenshot of a computer

Description automatically generated