Optimization Project

The main problem I found that was causing the project to be unoptimized was that We are deleting and reinstantiating critters whenever one is killed or respawned. This can be greatly improved by using an ObjectPool.

ObjectPool Requirements

The object pool is a templated class implemented with a Double Linked List utilizing it to have an active and inactive pool of objects.

How the object pool works

The object pool is made of two Double Linked List, ActiveList and InactiveList. It controls whether the objects in the pool are in the Active or Inactive list with 3 simple functions: AddToPool, SetInactive, and Activate. With these functions objects will be put into the pool with AddToPool and cycled into the InactiveList with SetInactive rather than deleted when killed. And later reactivated with Activate by being sent to the ActiveList in a first in first out pattern when respawning.

AddToPool

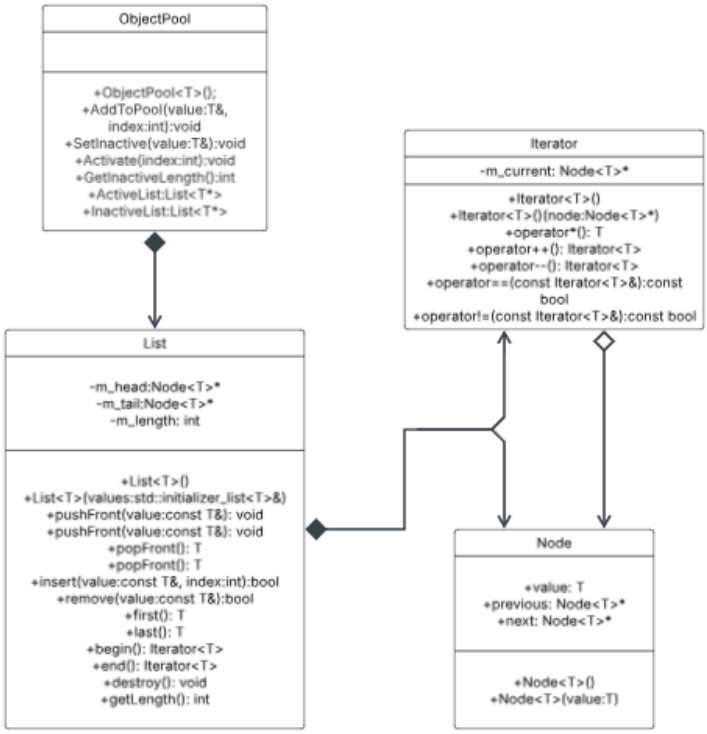
`ObjectPool.AddToPool(T& value, int index);`
When given a value it will insert the value into the ActivePool at the index given. This is what is used to fill up the pool with objects.

SetInactive

`ObjectPool.SetInactive(T& value);`
When given a value it will put the value in the back of InactiveList and then remove it from the ActiveList. This is how we will “kill” the critters in the project rather than deleting them.

Activate

`ObjectPool.Activate(int index);`
This will send the first value in the InactiveList to the ActivePool at the index specified, then it will popFront of InactiveList. This is how we will “respawn” the critters in the project rather than reinstantiating them.


ObjectPool
+ObjectPool<T>();
+AddToPool(value:T&,
index:int):void
+SetInactive(value:T&):void
+Activate(index:int):void
+GetInactiveLength():int
+ActiveList:List<T*>
+InactiveList:List<T*>
Iterator
-m_current: Node<T>*
+Iterator<T>()
+Iterator<T>(node:Node<T>*)
+operator*(): T
+operator++(): Iterator<T>
+operator--(): Iterator<T>
+operator==(const Iterator<T>&):const
bool
+operator!=(const Iterator<T>&):const bool
List
-m_head:Node<T>*
-m_tail:Node<T>*
-m_length: int
+List<T>()
+List<T>(values:std::initializer_list<T>&)
+pushFront(value:const T&): void
+pushFront(value:const T&): void
+popFront(): T
+popFront(): T
+insert(value:const T&, index:int):bool
+remove(value:const T&):bool
+first(): T
+last(): T
+begin(): Iterator<T>
+end(): Iterator<T>
+destroy(): void
+getLength(): int
Node
+value: T
+previous: Node<T>*
+next: Node<T>*
+Node<T>()
+Node<T>(value:T)