QuadTree Technical Summary

September 25, 2024

1 Object Pool

1. Initialization and Construction

The object pool is created with delegate functions passed into the constructor for creating, acquiring, releasing, and destroying objects. The constructor pre-creates a certain number of objects based on parameters and stores them in an internal collection, ensuring the object pool can provide available objects immediately after initialization. The maximum capacity is also determined at this stage to prevent the pool from expanding indefinitely.

2. Object Acquisition

- When an object is needed, the `Get()` method retrieves an object from the internal collection. If the object pool is empty, it calls the creation function to generate a new object and return it.
- After acquiring an object, the `getFunc` delegate is called to perform any necessary initialization or activation, ensuring the object is in the correct state before use.
- The `Get()` method also dynamically records the maximum number of simultaneously active objects in the pool, facilitating statistics and subsequent adjustments.

3. Object Release

- After using an object, the `Release()` method releases the object back into the pool.
- During the release process, the `releaseFunc` delegate is called to perform necessary reset or cleanup operations.
- If the object pool has not reached its maximum capacity, the released object is added back to the pool for future reuse; if it has reached maximum capacity, the `destroyFunc` is called to destroy the object, preventing excessive memory usage and reducing the total count of objects.

4. Clearing and Destroying the Object Pool

- The `Clear()` method provides the functionality to clear the object pool, traversing the internal collection, performing destruction operations on each object, and freeing memory.
- The `Dispose()` method implements the `IDisposable` interface, which is used to automatically release resources when the object pool is no longer needed. When `Dispose()` is called, it clears the object pool and releases all objects, ensuring no memory leaks.

5. Adjustment and Adaptation

- The object pool supports dynamic adjustment of its maximum capacity, which can be modified using the `SetMaxSize()` method.
- The `AdaptSize()` method adjusts the maximum capacity of the pool based on the recorded maximum number of active objects, ensuring the object pool can adapt to actual usage conditions.

6. Object Creation

- The `Instantiate()` method is used for batch creation of objects, adding them to the pool. This process loops through the specified quantity, calls the creation function, adds the created objects to the pool, and updates the total object count.

2. QuadTree

1. Initialization and Construction

- During the initialization of the quad-tree, the object pool is also initialized.
- When creating the root node of the quad-tree, the object pool is used to acquire the quad-tree node, and the initial depth and parent node are set.

2. Node Insertion and Subdivision

- The insertion method handles the process differently based on whether the node is currently a leaf node. When the number of child elements of a node exceeds a preset threshold and the depth has not reached the maximum value, the node will be subdivided into four subregions, and the child elements will be reinserted into the corresponding subregions.
- During the subdivision process, the node acquires four new child nodes from the object pool and initializes their rectangular regions and depth.

- For non-leaf nodes, the insertion method recursively inserts elements into the corresponding subregion based on the quadrant where the element is located.

3. Clearing and Releasing

- The quad-tree node provides a clear method, which releases the object list and child nodes stored in the node and its child nodes back to the object pool, achieving resource reuse and avoiding memory leaks.
- During the clearing operation, the root node will call the initialization method to ensure that the root node remains in a valid initialized state.

4. Spatial Query and Traversal

By recursively acquiring all child nodes, the algorithm determines whether the target object is within the current node, traversing layer by layer to find the points that might intersect with the target node's range.

5. Integration and Interaction

- The quad-tree test class creates a large number of objects through keyboard input and inserts them into the quad-tree for management and querying.
- By establishing the quad-tree on each frame, objects are dynamically inserted and spatial queries are conducted, supporting real-time visualization of the partitioned regions and objects in the scene, making debugging more convenient.
- The system provides the ability to quickly query the target object and find neighboring objects, allowing interactions such as highlighting or handling collisions with other intersecting objects.