四叉树技术总结

2024 年 9 月 25 日

1 对象池

1. 初始化与构造

对象池在创建时通过构造函数传入了对象的创建、获取、释放、销毁操作的委托函数。构造函数根据参数预先创建一定数量的对象，并将它们存储在内部集合中，确保对象池在初始化后能够立即提供可用对象。最大容量也在此阶段确定，防止池无限膨胀。

2. 对象获取

- 当需要对象时，`Get()` 方法会从内部集合中取出一个对象。如果对象池为空，会调用创建函数生成新的对象并返回。
- 对象获取后，`getFunc` 委托被调用，执行任何必要的初始化或激活操作。这样确保对象在被使用前处于正确的状态。
- `Get()` 方法还会动态记录池中最大同时激活的对象数量，方便统计和后续调整。

3. 对象释放

- 使用完对象后，通过 `Release()` 方法将对象释放回池中。
- 在释放过程中，`releaseFunc` 委托被调用，执行必要的重置或清理操作。

- 如果对象池未达到最大容量，释放的对象将被添加回池中，以供将来重复使用；如果已达到最大容量，调用 `destroyFunc` 销毁对象，防止内存过度占用，并减少对象总数的计数。

4. 对象池的清空与销毁

- `Clear()` 方法提供了清空对象池的功能，遍历内部集合，对每个对象执行销毁操作，并释放内存。
- `Dispose()` 方法实现了 `IDisposable` 接口，用于在对象池不再需要时自动释放资源。调用 `Dispose()` 时，会清空对象池并释放所有对象，确保没有内存泄漏。

5. 调整与适应

- 对象池支持动态调整最大容量，通过 `SetMaxSize()` 方法可以修改池的容量。
- `AdaptSize()` 方法基于记录的最大激活数量调整池的最大容量，确保对象池能根据实际使用情况自适应调整。

6. 对象的创建

- `Instantiate()` 方法用于批量创建对象，将其添加到池中。这一过程会根据传入的数量循环调用创建函数，并将创建的对象加入池内，同时更新对象总数。

2. 四叉树

1. 初始化与构造

- 在四叉树的初始化过程中，初始化对象池。
- 创建四叉树的根节点时，利用了对象池获取四叉树节点，并设置初始的深度和父节点。

2. 节点的插入与分割

- 插入方法根据节点当前是否为叶子节点进行不同的处理。当节点的子元素数量超过预设阈值并且深度未达到最大值时，节点会被分割成四个子区域，并将子元素重新插入到相应的子区域中。
- 在进行分割操作时，节点会从对象池中获取四个新的子节点，并初始化它们的矩形区域与深度。
- 对于非叶子节点，插入方法会根据元素所在的象限递归地将元素插入到对应的子区域中。

3. 清空与释放

- 四叉树节点提供了清空方法，能够将节点及其子节点中存储的对象列表和子节点本身释放回对象池，实现资源的重复利用，避免内存泄漏。
- 根节点在清空操作中会调用初始化方法，确保根节点始终保持有效的初始化状态。

4. 空间查询与遍历

通过递归获取所有子节点，判断目标对象是否在当前节点下，逐层遍历来查找有可能和目标节点相交范围内的点。

5. 整合与交互

- 四叉树的测试类通过键盘输入创建大量对象，并将其插入四叉树中进行管理和查询。
- 通过每帧建立四叉树，动态插入对象并进行空间查询，支持在场景中实时显示分割区域和对象，方便可视化调试。
- 提供了对目标对象的快速查询和相邻对象查找的能力，能够将目标对象与其他相交对象进行交互，例如高亮显示或处理碰撞。