Tetris 3D Unity

deepwaterooo

2019 年 7 月 6 日

目录

1 references

1.1 concurrent

- 用 Semaphore 实现对象池
  - https://donald-draper.iteye.com/blog/2360817

```java
package juc.latch;
import java.util.concurrent.Semaphore;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
/**
 * 信号量实现的对象池
 * @author donald
 * 2017 年 3 月 6 日
 * 下午 9:43:06
 * @param <T>
 */

public class ObjectCache<T> {

    // 对象工厂
    public interface ObjectFactory<T> {
        T makeObject();
    }

    // 将对象封装节点中，放到一个先进先出的队列中，即对象池
    class Node {
        T obj;
        Node next;
    }

    final int capacity; // 线程次容量
    final ObjectFactory<T> factory;
    final Lock lock = new ReentrantLock(); // 保证对象获取，释放的线程安全
    final Semaphore semaphore; // 信号量
    private Node head;
    private Node tail;

    public ObjectCache(int capacity, ObjectFactory<T> factory) {
        this.capacity = capacity;
        this.factory = factory;
        this.semaphore = new Semaphore(this.capacity);
        this.head = null;
        this.tail = null;
    }

    /**
     * 从对象池中，获取对象
     * @return
     * @throws InterruptedException
     */
    public T getObject() throws InterruptedException {
        semaphore.acquire();
        return getObjectFromPool();
    }

    /**
     * 线程安全地从对象池获取对象
     * @return
     */
    private T getObjectFromPool() {
        lock.lock();
        try {
            if (head == null) {
                return factory.makeObject();
            } else {
                Node ret = head;
                head = head.next;
                if (head == null)
                tail = null;
                ret.next = null;//  help GC
                return ret.obj;
            }
        } finally {
            lock.unlock();
        }
    }
    /**
     * 线程安全地，将对象放回对象池
     * @param t
     */
    private void putBackObjectToPool(T t) {
        lock.lock();
        try {
            Node node = new Node();
            node.obj = t;
            if (tail == null) {
                head = tail = node;
            } else {
                tail.next = node;
                tail = node;
            }
        } finally {
            lock.unlock();
        }
    }
    /**
     * 将对象放回对象池
     * @param t
     */
    public void putBackObject(T t) {
        putBackObjectToPool(t);
        semaphore.release();
    }
}
```

- Object pool pattern
  - https://en.wikipedia.org/wiki/Object_pool_pattern

```
namespace DesignPattern.Objectpool  {

    // The PooledObject class is the type that is expensive or slow to instantiate,
    // or that has limited availability, so is to be held in the object pool.
    public class PooledObject {
        DateTime _createdAt = DateTime.Now;
        public DateTime CreatedAt {
            get { return _createdAt; }
        }
        public string TempData { get; set; }
    }

    // The Pool class is the most important class in the object pool design pattern.
    // pooled objects, maintaining a list of available objects and a collection of o
    // requested from the pool and are still in use. The pool also ensures that obje
    // are returned to a suitable state, ready for the next time they are requested.
    public static class Pool {
        private static List<PooledObject> _available = new List<PooledObject>();
        private static List<PooledObject> _inUse = new List<PooledObject>();
        public static PooledObject GetObject() {
            lock(_available) {
                if (_available.Count != 0) {
                    PooledObject po = _available[0];
                    _inUse.Add(po);
                    _available.RemoveAt(0);
                    return po;
                } else {
                    PooledObject po = new PooledObject();
                    _inUse.Add(po);
                    return po;
                }
            }
        }
        public static void ReleaseObject(PooledObject po) {
            CleanUp(po);
```

```
                lock (_available) {
                    _available.Add(po);
                    _inUse.Remove(po);
                }
            }
            private static void CleanUp(PooledObject po) {
                po.TempData = null;
            }
        }
    }
```

- Sun ‘刺眼的博客: 随笔分类 - Unity3D、C#
  - https://www.cnblogs.com/android-blogs/category/879304.html
- Unity 协程（Coroutine）原理深入剖析
  - https://dsqiu.iteye.com/blog/2029701
- Unity3d IEnumerator 协程的理解
  - https://blog.csdn.net/jasonwang18/article/details/55519165
- 关于对象池的一些分析
  - https://droidyue.com/blog/2016/12/12/dive-into-object-pool/

1.2 Ð³Ì Coroutine

- http://dsqiu.iteye.com/blog/2029701
- http://dsqiu.iteye.com/blog/2049743

1.3 tetris 3d specific

- https://www.youtube.com/watch?v=UZSotPFf0ug with tutorial, Maya Unity
- above 2d tutorial http://noobtuts.com/unity/2d-tetris-game
- commands http://users.csc.calpoly.edu/~zwood/teaching/csc471/finalproj24/gzipkin/
- 3 other resources:
  - http://subject.manew.com/source/index.html
  - http://jingyan.baidu.com/article/4e5b3e195bde8991901e243a.html
  - http://www.cnblogs.com/bitzhuwei/p/unity3d-tank-sniper.html

1.4 buttons

- https://forum.unity3d.com/threads/touch-and-hold-a-button-on-new-ui.266065/
- https://stackoverflow.com/questions/38198745/how-to-detect-left-mouse-click-but-not-

1.5 3d games

- https://www.youtube.com/watch?v=_oEUJ_sirC8 with vedio downloaded
-
-
-
-
-
-
-