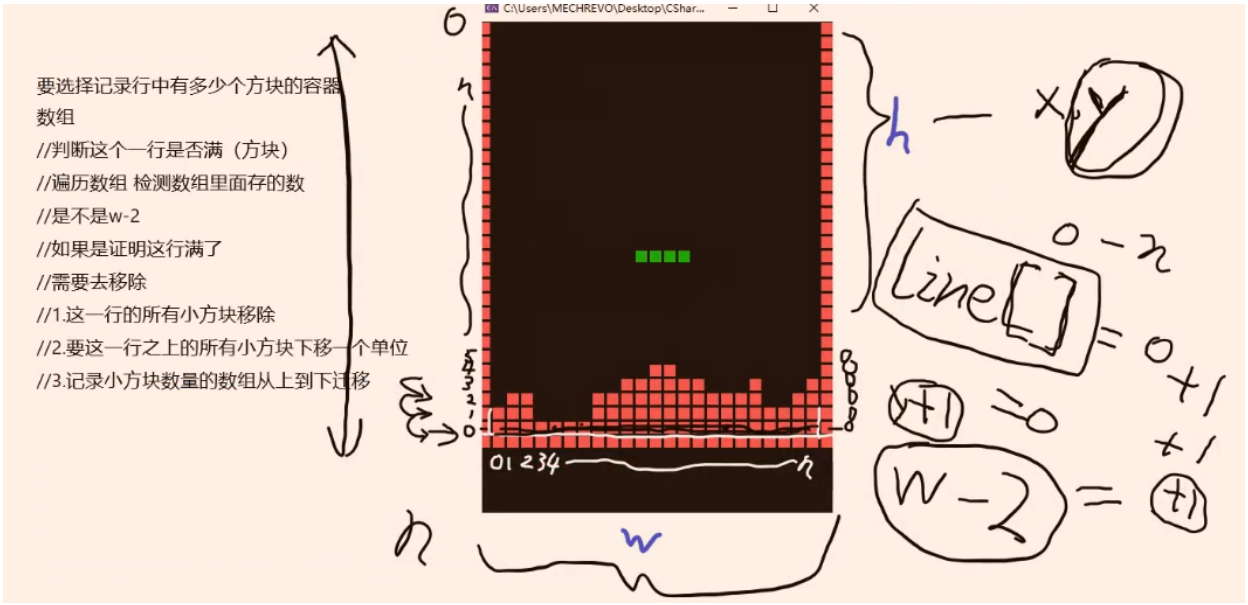
11：消除方块




```
namespace 俄罗斯方块
{
    6 个引用
    class Map:IDraw
    {
        //固定墙壁
        public List<DrawObject> walls = new List<DrawObject>();

        //动态墙壁
        public List<DrawObject> dynamicWalls = new List<DrawObject>();
        //动态墙壁的宽容量
        public int h;
        //为了外部好获取墙体的范围
        public int w;

        //记录每一行有多少个小方形的容器，索引是保存的行号
        private int[] recordInfo;
        1 个引用
        public Map()
        {

            h = Game.h - 6;//优化代码
            //这个代表对应每行的计数初始化，默认都为0
            //h 就是0~Game.h-7
            recordInfo = new int[h];//这里的h是0~Game.h-6，从上往下0开始，到h-6的容器
            for (int i = 0; i < Game.w; i+=2)//添加下面的墙壁
            {
                walls.Add(new DrawObject(E_DrawType.Wall, i, h));
                ++w;
            }
            w -= 2;//最后需要减去左右两边的格子

            for (int i = 0; i < h; i++)//添加左右的墙壁
            {
                walls.Add(new DrawObject(E_DrawType.Wall, 0, i));//添加左边
                walls.Add(new DrawObject(E_DrawType.Wall, Game.w-2,i));//添加右边
            }
        }

        3 个引用
        public void Draw()
        {
            for (int i = 0; i < walls.Count; i++)//绘制固定墙
            {
                walls[i].Draw();
            }

            for (int i = 0; i < dynamicWalls.Count; i++)//绘制动态墙壁
            {
                dynamicWalls[i].Draw();
            }
        }

        1 个引用
        public void ClearDraw()//清除动态方块
        {
            for (int i = 0; i < dynamicWalls.Count; i++)
            {
                dynamicWalls[i].ClearDraw();
```

```
        }
    }

    //后续外部会有添加动态墙壁
    2 个引用
    public void AddWalls(List<DrawObject> walls)
    {
        for (int i = 0; i <walls.Count ; i++)
        {
            //传递方块进来时，把其类型改为墙壁
            walls[i].ChangeType(E_DrawType.Wall);
            dynamicWalls.Add(walls[i]);//添加到动态墙壁列表中

            //进行添加动态墙壁的计数
            //根据索引来得到行
            // h就是 game.h-6 那我们的第一行是game.h-7 所以需要-1
            //y最大就是Game.h-7
            //且不会重复，因为重合会停下，所以每一行都有最大值(Game.w-2)最大值就表示一行满了
            recordInfo[h - walls[i].pos.y - 1] +=1;//这里就可以得到最开始是从0行开始的（也就是小方形y轴的最大值）
        }


        //检查是否需要跨层
        //1、先移除动态方块（就算真的不需要迁移，后面还会绘制出来
        ClearDraw();
        //2、检测移除
        CheckClear();
        //3、在绘制动态方块
        Draw();
    }

    /// <summary>
    /// 跨层函数
    /// </summary>
    2 个引用
    public void CheckClear()
    {
        for (int i = 0; i < recordInfo.Length; i++)
        {
            List<DrawObject> delList = new List<DrawObject>();

            //这里的w已经去掉了两头的固定墙壁
            if(recordInfo[i]==w)//成立说明一行满了
            {
                //1、这一行的小方块移除
                for (int j = 0; j < dynamicWalls.Count; j++)
                {
                    //当前行号与我们记录数组的行号一致说明这一行满了
                    if (i == h-dynamicWalls[j].pos.y-1)
                    {
                        //为了安全移除，添加到一个记录表
                        delList.Add(dynamicWalls[j]);
                    }
                    //2、要这一行的所有小方块下移一个单位
                    //如果当前的这个位置，是该行以上，那就需要下移一格
                    else if (h - 1 - dynamicWalls[j].pos.y > i)
                    {
                        //因为能进入到这个for语句中，说明已经有一行是需要清除的
                        ++dynamicWalls[j].pos.y;
                    }
                }
                //移除待删除的小方块
                for (int j = 0; j < delList.Count; j++)
                {
                    dynamicWalls.Remove(delList[j]);
                }
                //3、记录小方块数量的数组从上到下移
                for (int j = i; j < recordInfo.Length-1; j++)
                {
                    //这里的每一个元素就是行号，不是方块的坐标
                    //所以这里就是把上面的一行往下移动
                    recordInfo[j] = recordInfo[j + 1];
                }
                //还有最关键的一行，清除最顶上的一行(因为下移了一行)
                recordInfo[recordInfo.Length - 1] = 0;
                //垮掉一行之后，再次从头检测是否还有跨层，因为如果2层要掉
                //但是跨了第一层i=1了，下来的一层也要跨，但是i=0，所以
                //就永远也跨不了，所以需要从新遍历，这里就可以用到递归
                CheckClear();//当没有可以垮掉的，就不会进入这个if，就会终止递归
                break;
            }
        }

    }
```