**Data Structure and Algorithm Analysis B Project 2022 Spring: RGBY Cell Life**

**李显涵 12013003 赵瑞州 12011622 沈浩瑜 11911022**

ⅰ.Introduction

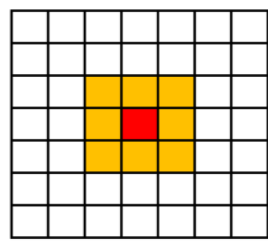
在这个项目中，我们在一个矩形的二维平面空间上模拟一些简化的虚拟单细胞生命。所有的细胞都是完美的圆，且每一个细胞都具有自己的ID，这些身份信息包括细胞的半径、位置、颜色和感知范围等。本系统通过细胞特定的运动探测感知范围并变化颜色实现了对于球体运动状态的理解与预测。本模型基于元胞自动机的原理，可以模拟细胞的生长与变化、气体中分子的运动、化学反应的动态过程、原子扩散、最密堆积问题、某些元素的相变等许多问题。

ⅱ.Principles

1. 碰撞以及检测区域方法

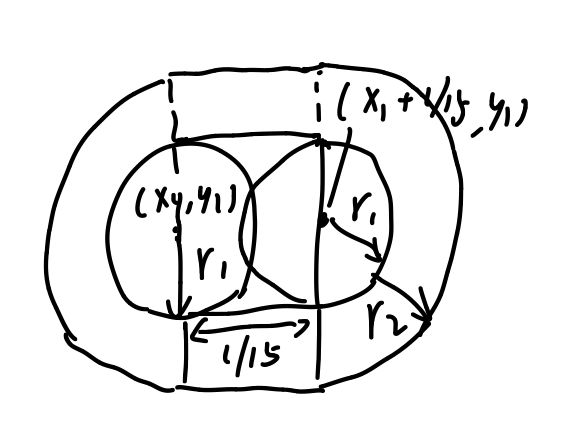
项目基于元胞自动机的原理，先简单介绍元胞自动机的概念。元胞自动机是一种时间、空间、状态都离散，空间相互作用和时间因果关系为局部的网络动力学模型，具有模拟复杂系统时空演化过程的能力。

元胞是元胞自动机最基本的要素，在本项目中元胞即为每一个“细胞”。元胞状态就是某一元胞在某一时刻所属的状态，在我们的项目中每一个细胞都具有自己特定的身份。元胞的状态会随着空间和时间不断改变。元胞邻居是某一元胞状态更新时所要搜索的空间域就是该元胞的邻居。即为下图所示，本项目中细胞的搜索范围为以其圆心为中心的一个正方形范围。

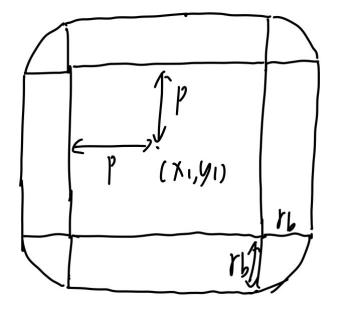
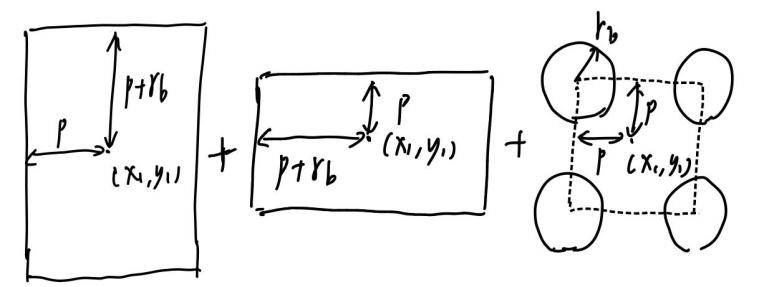


元胞状态会由其邻居而发生状态的改变，最边界的那个元胞没有邻居此时即为具有边界条件，在理想情况下，元胞空间是无限的但是实际情况无法达到这种情况，所以就要给这些元胞空间加上一个边框，创造出虚拟的邻居，本项目中窗口的边界即为“墙”，细胞不得越界和重叠。元胞的状态和邻居的状态这个规则下，决定下一时刻元胞的状态。在监测范围内细胞的状态会根据要求发生变化。这样就从量变到质变的过程，形成了一种演化。

在本项目中，细胞根据数学上的几何规则进行检测与变化。首先，细胞不得重叠，根据不等式，细胞不会发生撞击后交叉。这里我们先划出A细胞此回合移动的轨迹范围，再判断其他细胞是否可能与该轨迹重合。具体实现是在A细胞移动轨迹上的范围上再加上一圈另一个细胞的半径，划出一个更大的范围（如下图，以向右移动为例），另一个细胞的圆心落入该范围就意味着正常会与正常移动下的A细胞穿模。我们将会与A细胞穿模的所有细胞装入一个动态数组内，接着遍历动态数组，分别算出A细胞本回合需要移动多远才能正好碰到数组中每个细胞，并取出移动距离最小值。接着判断A细胞本回合内正常移动是否会撞墙（这一步就算上述动态数组为空也可以执行），如果会则算出撞墙须移动距离，并与上一步移动距离最小值比较并取小的值（若上一步没有则直接跟1/15比较），得到一个最终距离。最后细胞移动最终距离（若都不碰则正常移动1/15）。此外，为避免A细胞前半部分由于别的细胞运动的微小误差产生的微小干涉而产生直接穿模或向反方向移动的结果，我们把碰到每一个干涉范围内细胞的移动距离进行处理，如果为负数则取0.0；为避免A细胞后半部分由于别的细胞运动的微小误差产生的微小干涉而被卡住不能正常运动，我们将A细胞运动轨迹范围原来圆心位置之后的半个圆的部分去掉，这样可以解决。



其次，在矩形检测周围细胞中，只要另一个细胞上任意一点进入到该细胞的正方形检测范围中，就视为被该细胞检测到。因此，另一个细胞的圆心只要进入到如左下图所示的范围内即被检测到。该范围在代码实现上可以分为几个范围的并集，如右下图所示。

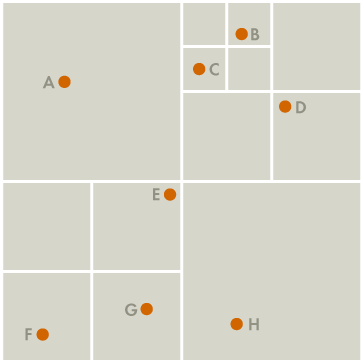
1. 采用Barnes-Hut Tree

采用 Barnes-Hut Tree 递归地储存输入的球体数据

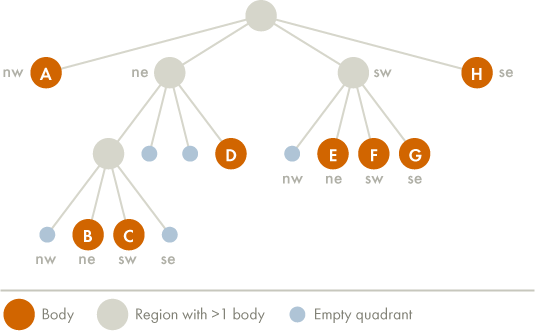
Barnes-Hut Tree

The Barnes-Hut algorithm is a clever scheme for grouping together bodies that are sufficiently nearby. It recursively divides the set of bodies into groups by storing them in a quad-tree. A quad-tree is similar to a binary tree, except that each node has 4 children (some of which may be empty). Each node represents a region of the two dimensional space.

The topmost node represents the whole space, and its four children represent the four quadrants of the space. As shown in the diagram, the space is recursively subdivided into quadrants until each subdivision contains 0 or 1 bodies (some regions do not have bodies in all of their quadrants. Hence, some internal nodes have less than 4 non-empty children).



Each external node represents a single body. Each internal node represents the group of bodies beneath it, and stores the center-of-mass and the total mass of all its children bodies. Here is an example with 8 bodies:



To calculate the net force on a particular body, traverse the nodes of the tree, starting from the root. If the center-of-mass of an internal node is sufficiently far from the body, approximate the bodies contained in that part of the tree as a single body, whose position is the group’s center of mass and whose mass is the group’s total mass. The algorithm is fast because we don’t need to individually examine any of the bodies in the group.

If the internal node is not sufficiently far from the body, recursively traverse each of its subtrees. To determine if a node is sufficiently far away, compute the quotient s / d, where s is the width of the region represented by the internal node, and d is the distance between the body and the node’s center-of-mass. Then, compare this ratio against a threshold value θ. If s / d < θ, then the internal node is sufficiently far away. By adjusting the θ parameter, we can change the speed and accuracy of the simulation. We will always use θ = 0.5, a value commonly used in practice. Note that if θ = 0, then no internal node is treated as a single body, and the algorithm degenerates to brute force.

ⅲ.Code Structure

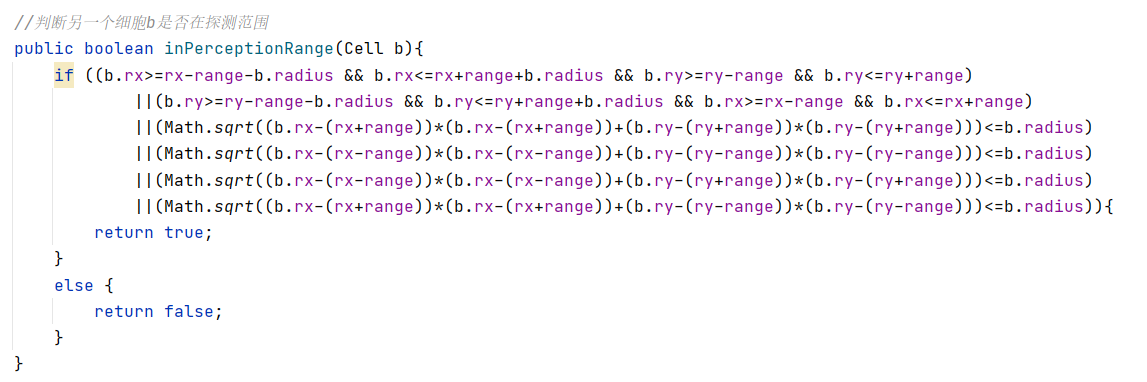
1. Cell

首先定义细胞参数（身份信息），这些参数包括细胞的半径，位置，细胞运动的速度，检测范围，颜色以及细胞检测周围细胞个数的能力。然后初始化这些细胞参数，将细胞参数以数组的形式读入，并初始化边界参数。

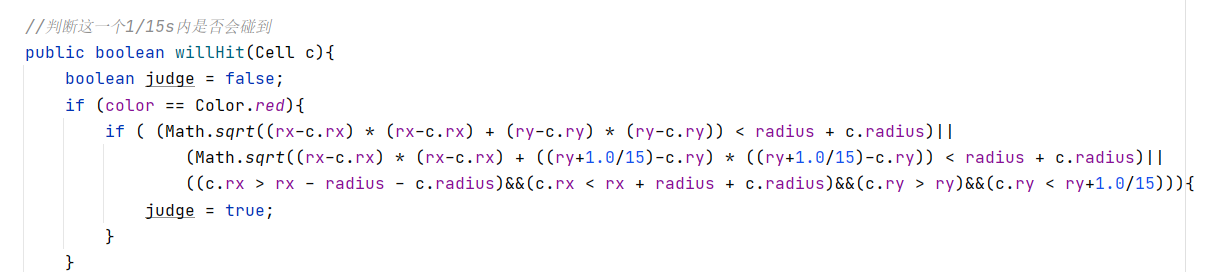


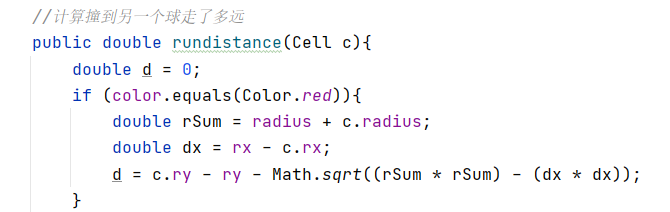
用细胞颜色来获取其速度，根据定义确定其速度移动的方向，以1为单位确定其初始速度。之后根据物理公式s=v\*t，以细胞的速度与单位时间的乘积rx += vx \* dt，ry += vy \* dt更新细胞移动一格的位置，同时以速度方向确定细胞的左右与上下移动的方向。

利用之前前文提及的数学几何关系判断另一个细胞b是否在探测范围，在NCMap通过判断细胞数量和颜色的原则画出当前细胞并更新颜色，将当前细胞颜色转换为小写字母标号，为读出准备。之后判断细胞是否在Map中划分对应的四个象限内，根据四叉树的优化算法（补充）。

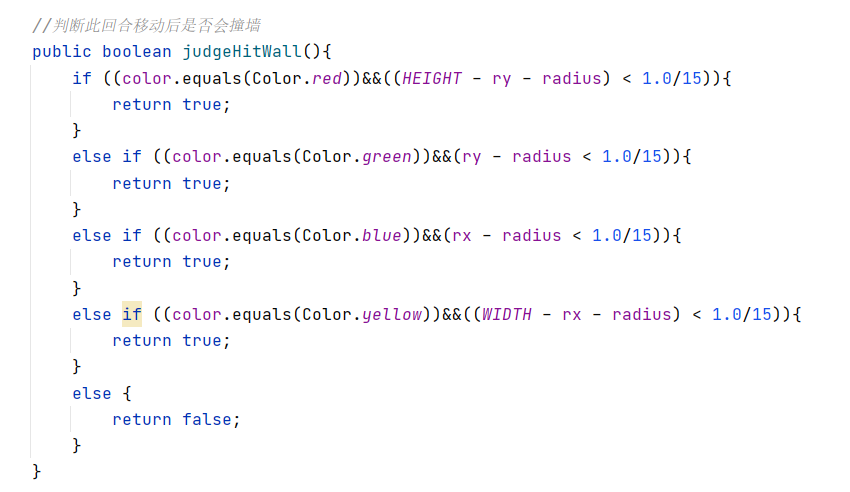


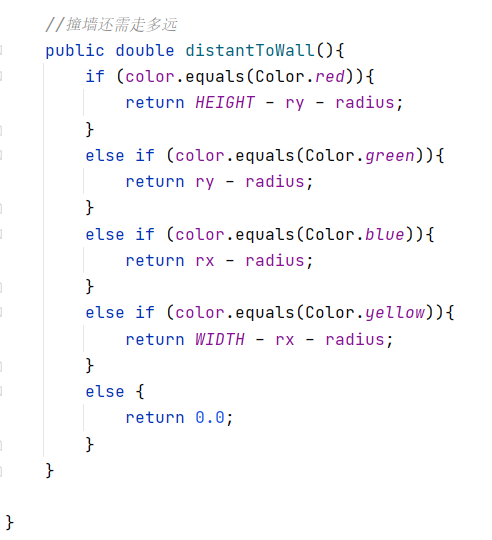
接下来是小球之间和小球与墙碰撞的判断。四种颜色的小球互相撞击有4种判断方法，我们在这里只举例红球的判断方法，其他三种同理。对于红球来说引入半径距离rx和其他小球半径c.rx，通过公式计算当前时刻和下一时刻的小球不得相互重叠。以及四种情况（不太会描述，补充）。同时计算小球撞到另一个小球走了多远（补充）。





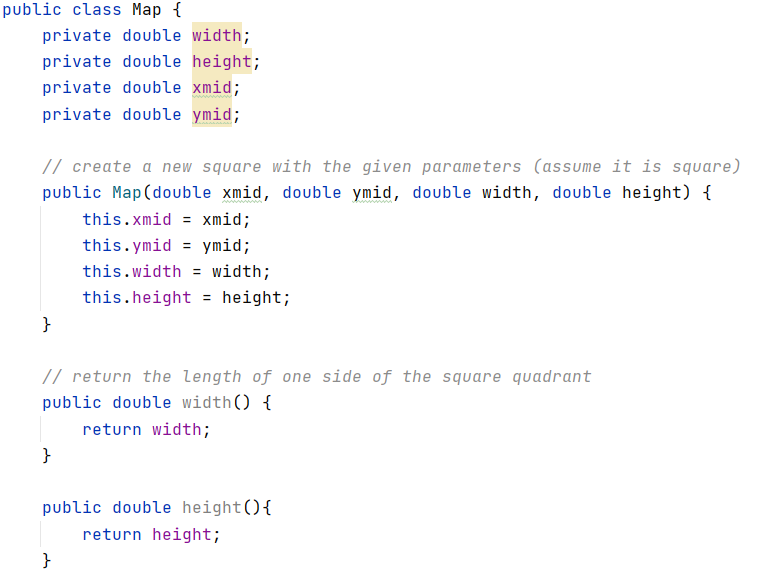
判断回合移动后细胞是否会撞墙同理，若细胞与墙面的距离大于单位移动步距，则细胞继续向前运动，若细胞与墙面的距离小于单位移动步距，则计算与墙面距离的真值并移动。



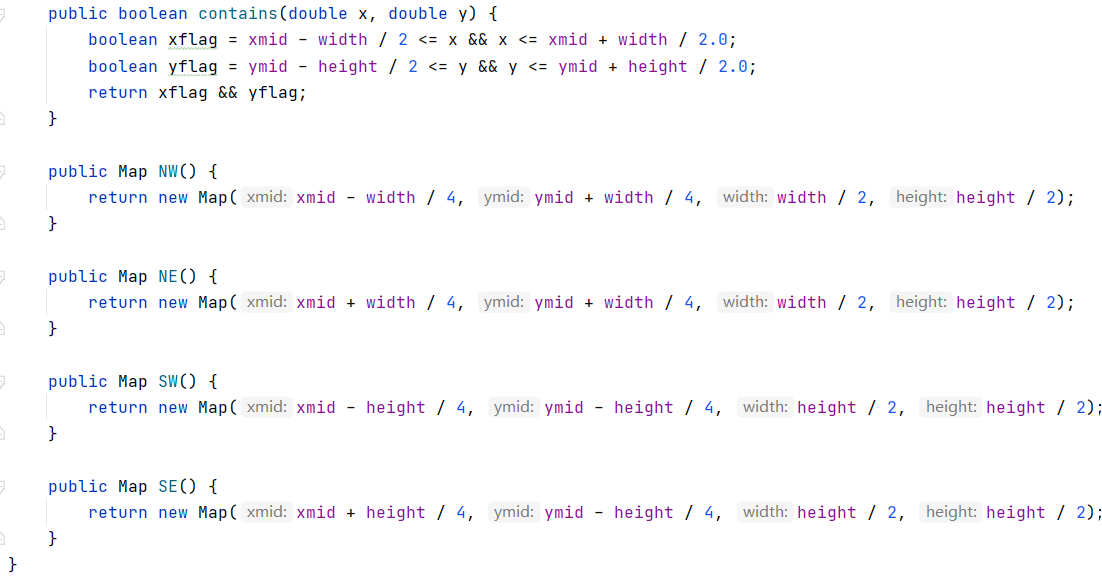


1. Map

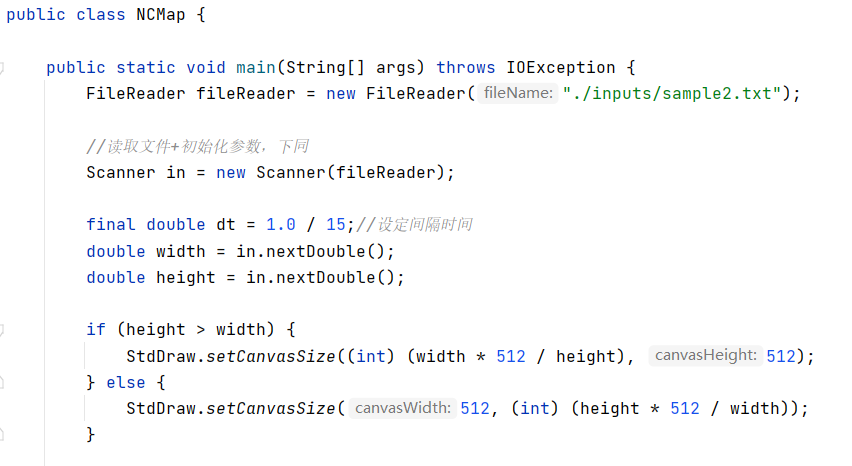
运行面板的绘制，利用坐标参数创造一个矩形运行面板



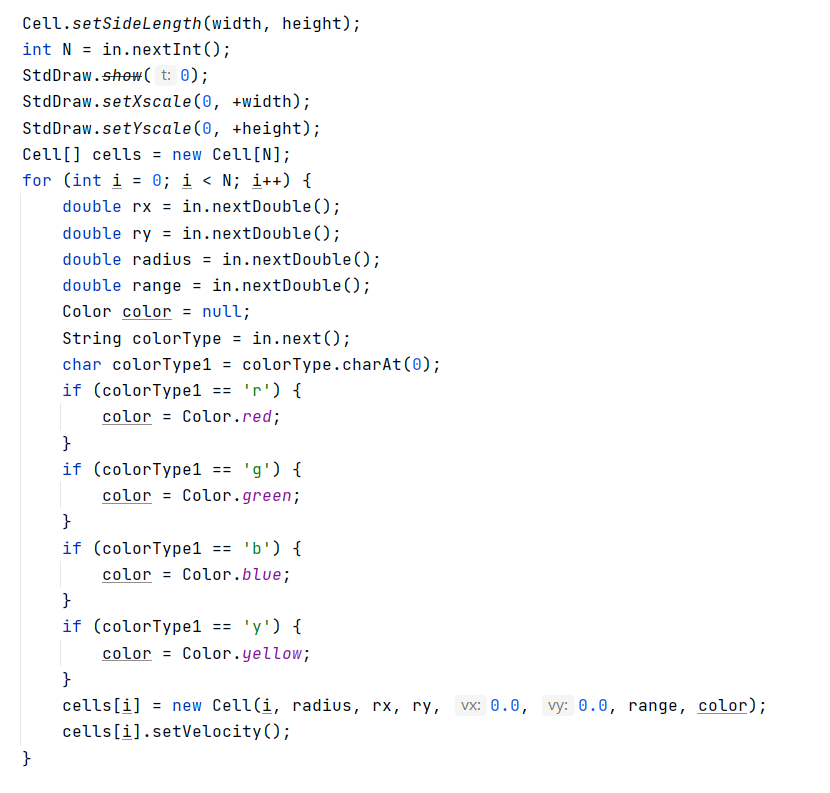
创造四个方位象限，并做划分和定义，为CellBHTree四叉树设定基准



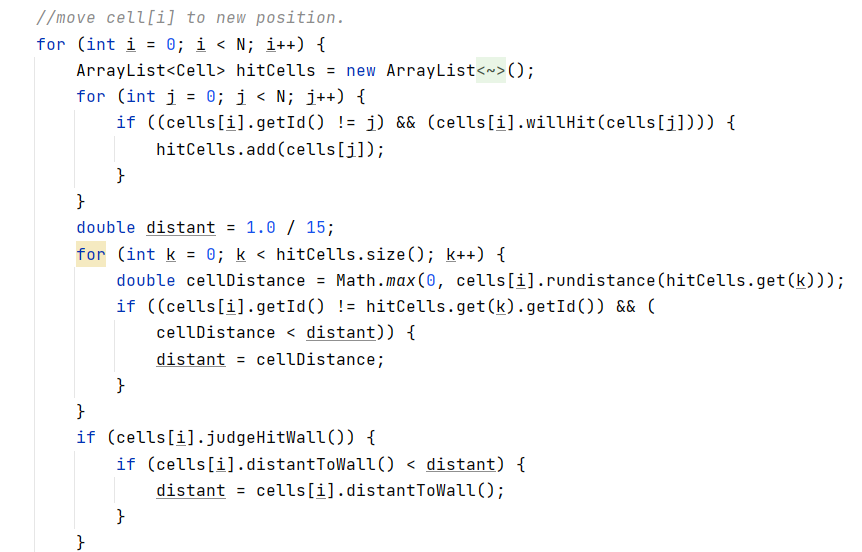
1. NCMap

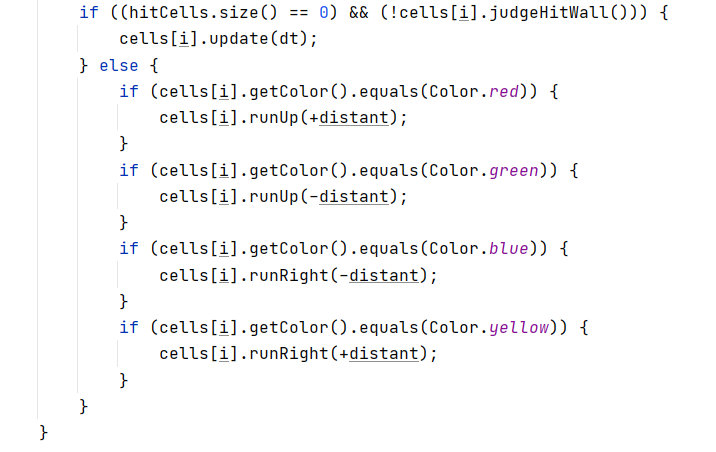


NCMap即为读取文件并在窗口绘制和运行的代码文件。首先读取文件细胞参数，设定时间间隔并绘制窗口。将样例中的细胞参数读取为数组，同时转化为窗口颜色显示。

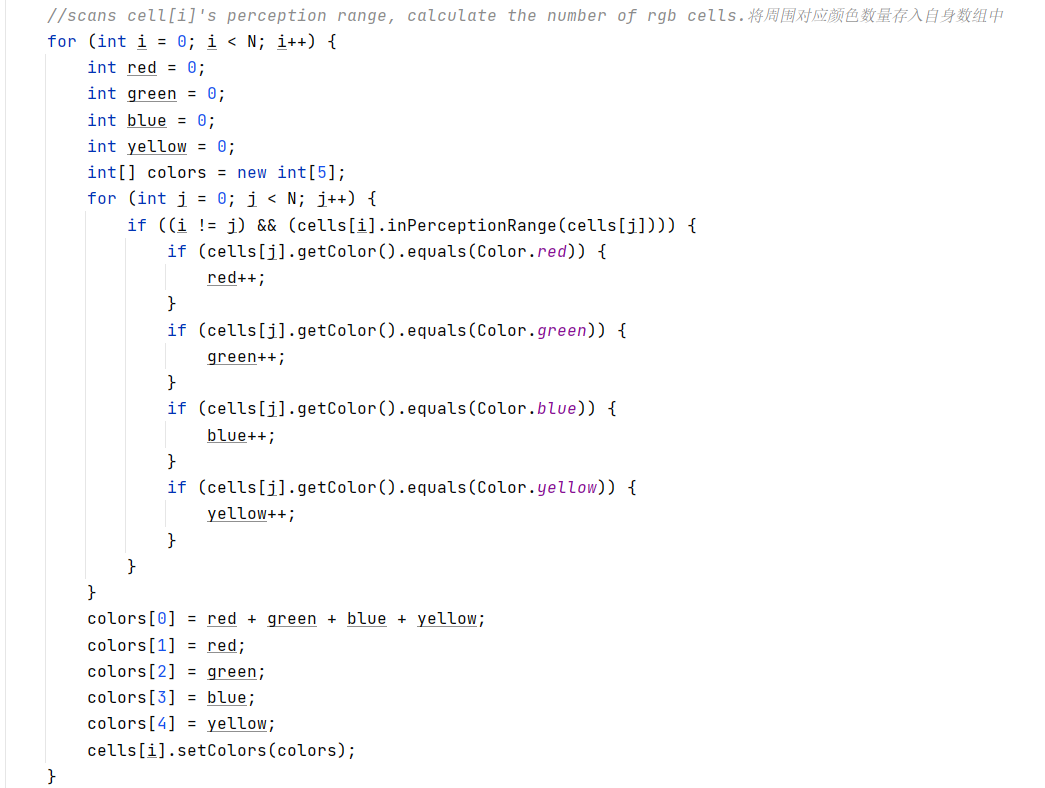


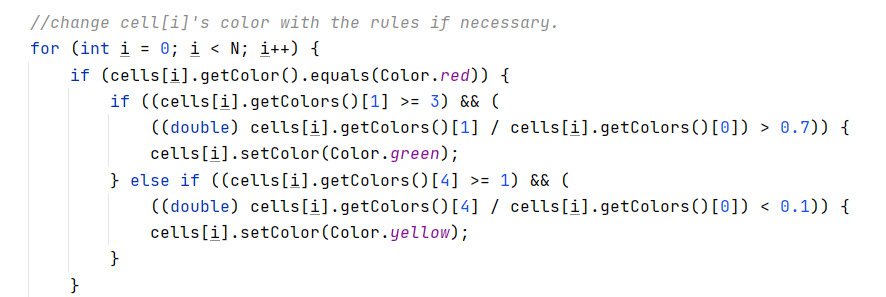
（感觉没啥好写的，而且这部分穿模bug的没修复好）





细胞对周围的感知范围进行扫描，读取周围四种不同颜色的细胞数量并存入自身的数组中。根据细胞变色规则，对于红色的细胞来说，首先如果其感知范围内(不包括自身)至少有3个红细胞，且其感知范围内的红细胞百分比大于70%(不包括自身)，则为绿色。否则，如果在其感知范围内至少有1个黄色细胞，且其感知范围内黄色细胞的百分比小于10%，则变为黄色。这两种情况均不满足则不变色继续运动。另外三种变色情况根据规则同理。





4.KdTree

优化部分采用了K-dimension Tree 的结构进行了变色搜索的优化。KdTree作为二叉树的变种，适用于范围搜索，碰撞检测等应用中。点对平面分割按照纵向和横向交替进行。规则为如果要插入的点在父节点的左边或者下边，则插入其左子树，反之插入右子树。

KdTree部分主要由三个程序组成，Cell.java, Rect.java以及KdTree.java。

**KdTree.java**

在该程序中主要实现了KdTree的搭建，节点的构造与插入Cell，查询是否包含某细胞，返回矩形空间内所有Cell以及查找最近点。

文本

描述已自动生成

该部分为节点构造函数，Node的参数为Cell。

文本

描述已自动生成

该部分为节点的插入函数。如果当前节点为null，则将cell插入该节点。否则按照分割方向，若分割方向为纵向，比较要插入Cell的x坐标值与node.cell的x坐标值，若插入Cell x坐标值较小，则向该节点的左子树递归插入，反之向右子树。

文本

描述已自动生成

查找函数类似插入函数，若当前节点的Cell与查找值相等，返回true，反之继续递归。

文本

描述已自动生成

查找range并返回包含所有Cell的list，逻辑与前两个函数类似。判定是否包括将在Rect.java中分析。

**Rect.java**

在该程序中实现了KdTree查询的矩形范围以及Cell查找范围的查询矩形。

该部分为Rect的构造函数，矩形为以Cell的坐标为中心，2\*Cell.range为边长的正方形。

文本

描述已自动生成

这部分为判断Cell是否在Rect范围内。首先include2函数判断Cell的中心坐标是否在Rect内，在则返回true。不在访问distanceSquareToPoint函数，查询点到矩形的距离。若Cell圆心在X方向外，则设定dx为圆心到矩形边界的x距离，反之dx = 0。Y方向同理，最后返回dx与dy的平方和。在include2中与Cell的半径平方和比较，若小于等于则返回true，在查询范围。

文本

描述已自动生成

Cell.java与NCMap.java中剩下部分与变色有关。

ⅳ.Results and Analysis

ⅴ.Conclusion and Experience

ⅵ.References

N-Body Simulation:

https://introcs.cs.princeton.edu/java/assignments/nbody.html

N-Body Simulation, with Barnes-Hut tree:

https://introcs.cs.princeton.edu/java/assignments/barnes-hut.html