Advanced Programming* 课程设计报告 LaTeX

* Teacher: Shujian Huang. TA: Yiyu Zhang

1st 张逸凯 171840708

Department of Computer Science and Technology

Nanjing University

zykhelloha@gmail.com

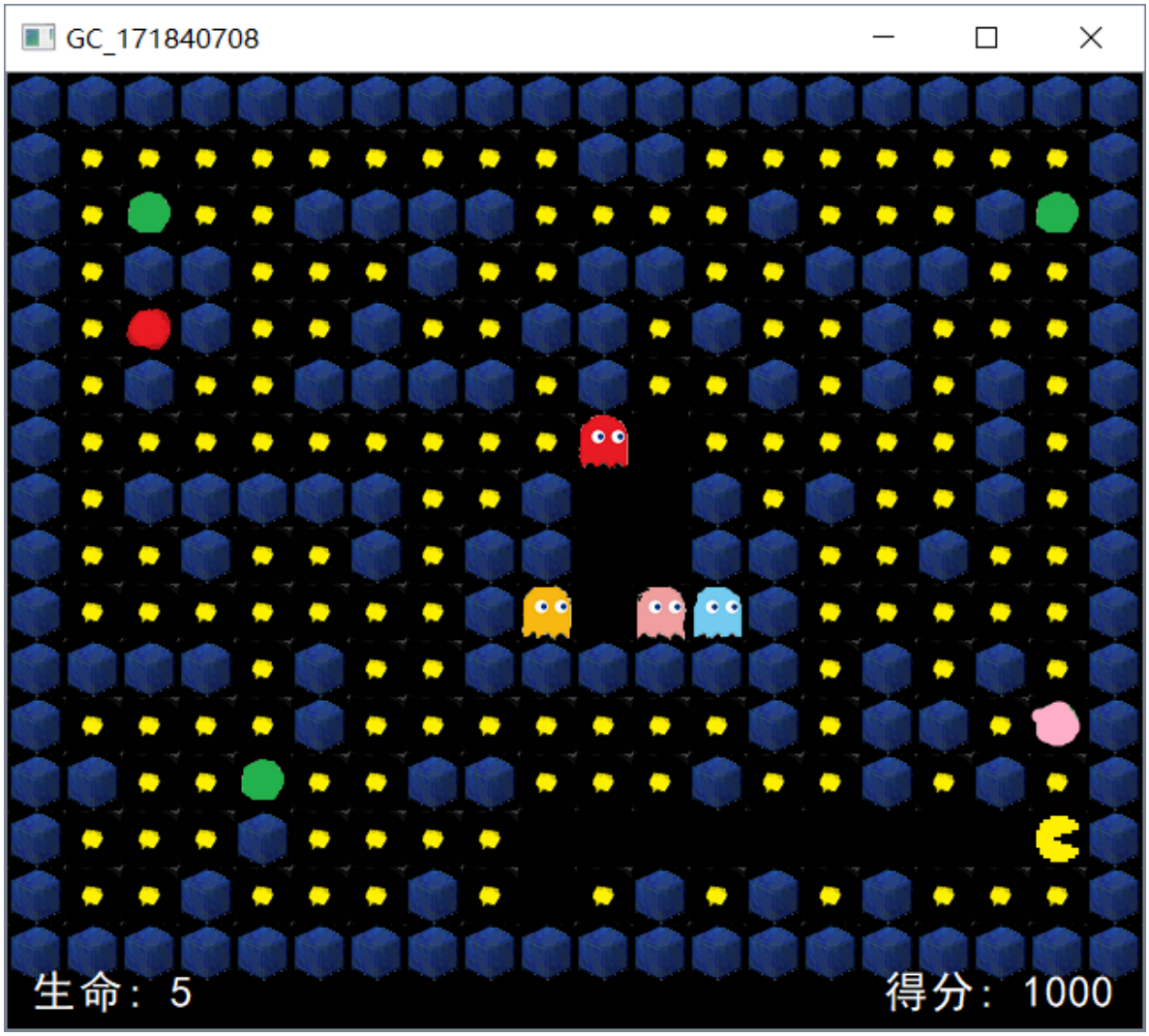
基于广义方向宽度优先搜索的吃豆人小游戏

171840708 **张逸凯**

项目主要内容与目标

实现小型GUI应用程序开发: 吃豆人小游戏, 并且是OOP风格以及C++ GUI的.

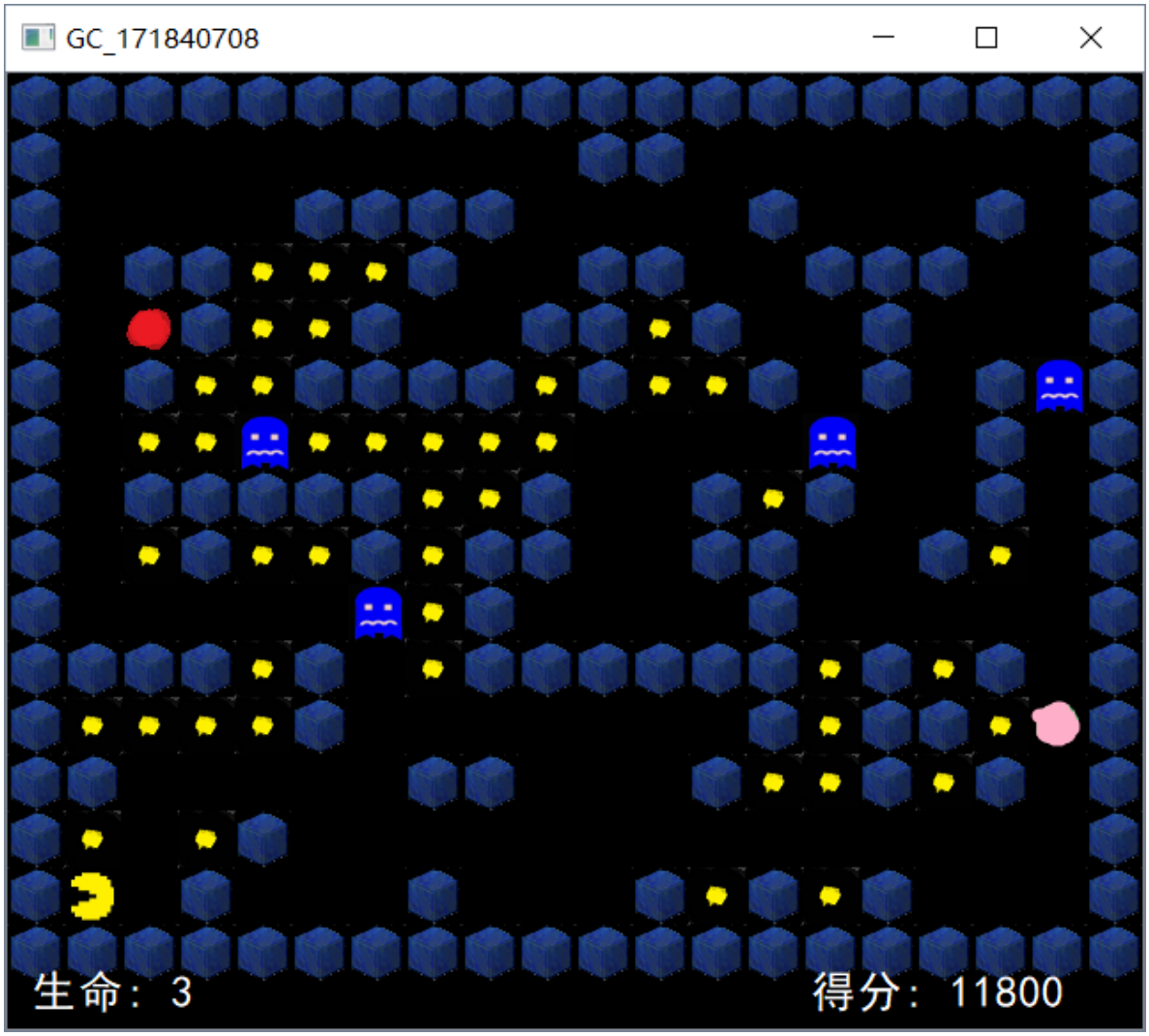
项目模仿Pacman游戏(*Pac-Man*[a] is a maze arcade game developed and released by Namco in 1980.)实现了基于Qt5框架的面向对象程序. 画面动感, 内容充实, 操作顺滑, 游戏体验极佳.



操作指南

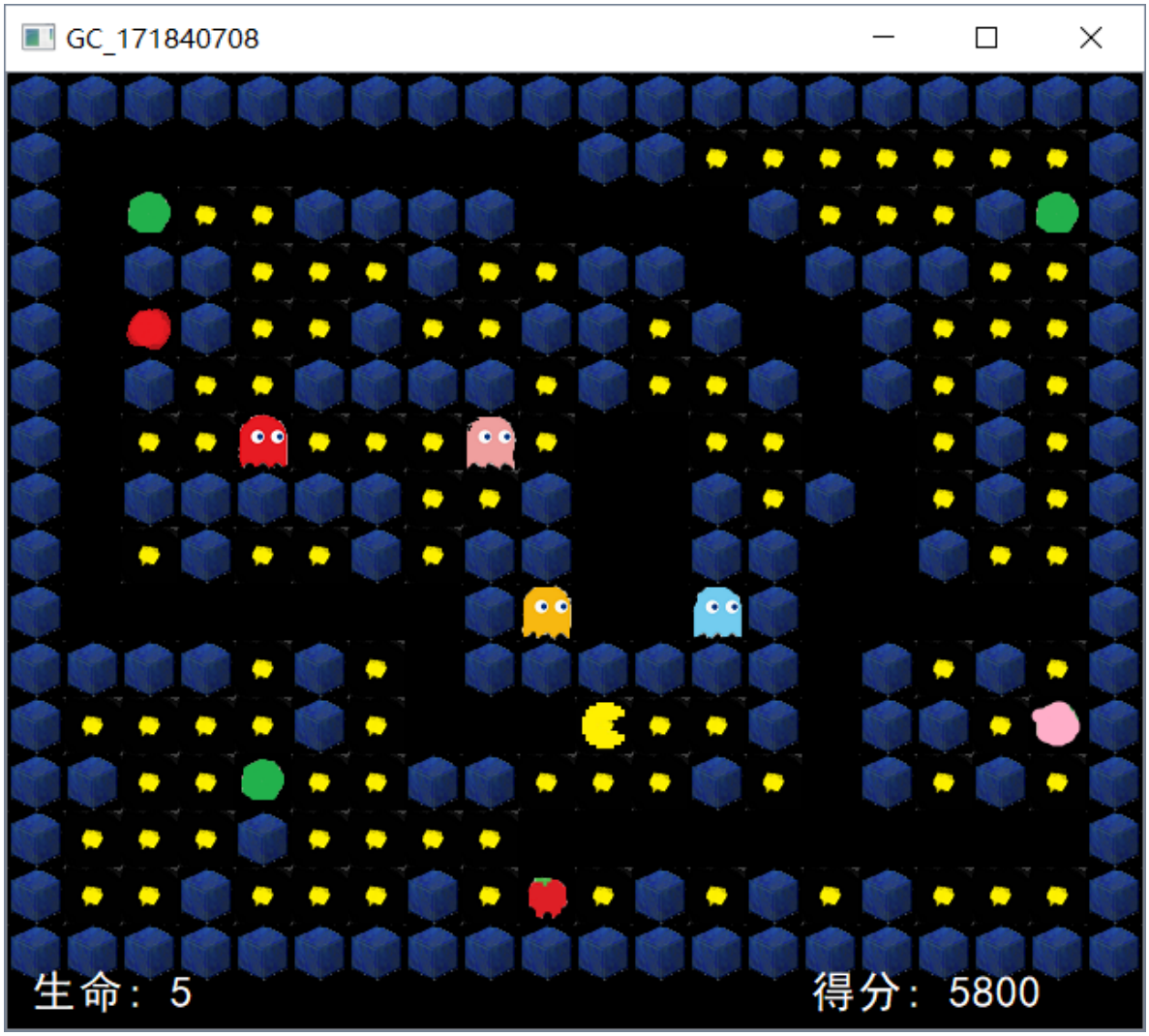
开始游戏后方向键控制吃豆人(黄色)的方向与行动, 目标是吃掉所有黄色豆豆, 碰到绿色豆豆之后**怪兽会中毒**, 需要回到相应的复活点才可以对吃豆人实施有效攻击, 吃豆人碰到怪兽就掉血, **不同颜色的怪兽会以不同的方式来追吃豆人**, 游戏过程中有随机大礼物(苹果或者爱心)出现, 捡到可以获得奖励.

还等什么呢, 赶快玩起来吧(滑稽)



上图是怪兽中毒:

上图是怪兽想回到红色点复活;



上图是出现惊喜苹果(还有惊喜爱心).

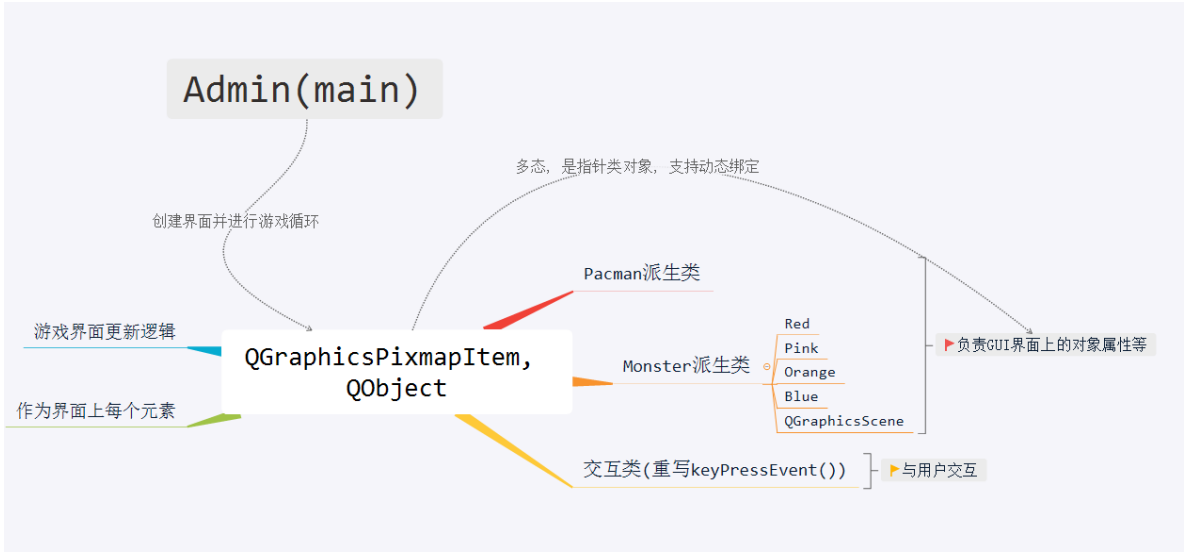
上图是游戏结束.

上图是你赢了(得分达到一定程度).

还有每次怪兽出动会发出诡异的喵呜 以及 动感的背景音乐 Funky Drums!

项目整体架构

类间关系



面向对象项目结构越写越精简, 首先需要符合Qt逻辑因为模块之间耦合度低, 模块内聚高, 每个对象(吃豆人或者怪兽)有特定的方法, 同一类的比如Monster类里的被多态调用, 这些对象所产生的影响将被写到全局的map中, 不同对象根据**写到全局map的影响以做出相应的动作**(比如掉血, 中毒, 解毒等). 对于不同的Monster, 具体的实现时不同的, 但是调用处可以相同, 所以他们的基类都是一致的, 这样使我在实现的时候可以多态调用, 游戏界面是QGraphicsScene实现的. 交互类涵盖了各种获取用户动作以及反馈给Pacman或Monster对象的方法, 还有参数调整和背景音乐控制.

整体来说结构精简, 但是实现起来有很多细节需要注意, 很多方法需要结合结构合理实现.

游戏主循环:

```
void MainLoop::curLoop() {
    timerRedr = new QTimer();
    timerPinkr = new QTimer();
    timerOranger = new QTimer();
    timerPacman = new QTimer();
    timerBluer = new QTimer();
    QObject::connect(timerPacman, SIGNAL(timeout()), pacman, SLOT(move()));
    QObject::connect(timerBlue, SIGNAL(timeout()), bluer,
SLOT(moveMonster()));
    QObject::connect(timerOrange, SIGNAL(timeout()), oranger,
SLOT(moveMonster()));
    QObject::connect(timerRed, SIGNAL(timeout()), redr,
SLOT(moveMonster()));
    QObject::connect(timerPink, SIGNAL(timeout()), pinkr,
SLOT(moveMonster()));
    timerPacman->start(310);
    timerBluer->start(420);
    timerOranger->start(400);
    timerRedr->start(370);
    timerPinkr->start(350);
}
```

加分项

基于广义方向宽度优先搜索

需求分析:

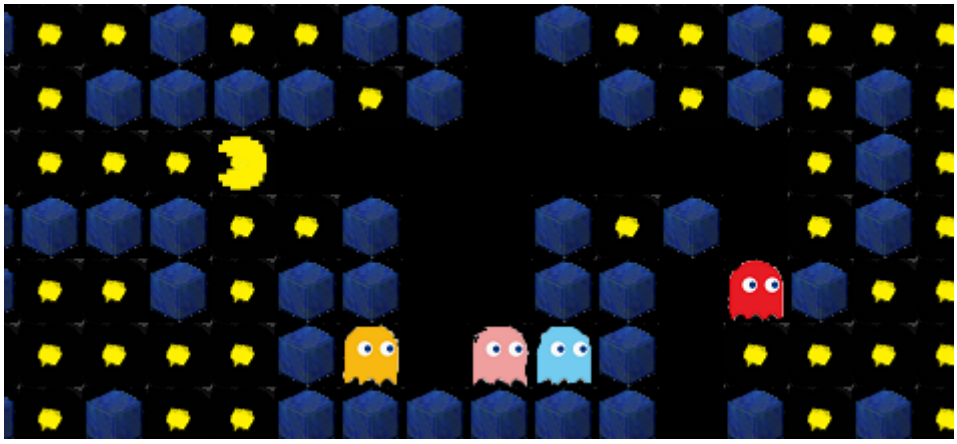
怪兽追吃豆人, 但是怪兽和吃豆人的位置时刻在变.

问题关键是确定**怪兽下一步的方向**.

解决思路:

对于怪兽的每个位置, 建立四个方向的索引, 这四个方向关联着当前BFS宽搜达到吃豆人位置的最短路径, 在下一个时间步, 如果吃豆人的位置偏移了某个方向的路径, 则更新在这个方向上的BFS路径. 同理如果在怪兽在下一个时间步步进一步时, 会带来三个方向的同步更新.

内部维护四个队列, **分别代表四个方向上的最短路径**. 每一次不更新的条件为当且仅当这条路径上的追及对象移动后的位置仍然在这条路径上. 反之则很容易找出反例.



如图所示此时如果吃豆人往右走, 则怪兽向上的追及路径是不需要更新的, 但是向下的追及路径需要更新. 对于每次怪兽移动一定是可达的, 因为地图可走地区连通并且*每次步进都是在减小计算通路上的距离*.

算法伪代码:

```
for (四个方向的循环) {
    如果对象的步进方向不在路径上, 更新路径: {
        准备队列(准备行进方向)
        do {
            取队头元素, 将队头元素宽度遍历的结点放入队列...
            如果取出元素已经达到对象, 保存路径.
        } while(队列不空时);
    }
    else {
        进行路径微调. 从偏离点开始延伸路径.
    }
}
```

利用**Qt5**逻辑精简设计

Detailed Description ¶

The QTimer class provides a high-level programming interface for timers. To use it, create a QTimer, connect its timeout() signal to the appropriate slots, and call start(). From then on, it will emit the timeout() signal at constant intervals.

利用`QTimer`以固定的间隔发出`timeout()`信号, 这样对象的移动操作可以很好的被"并行"设计.

如下游戏主循环:

```
void MainLoop::curLoop() {
    timerRedr = new QTimer();
    timerPinkr = new QTimer();
    timerOranger = new QTimer();
    timerPacman = new QTimer();
    timerBluer = new QTimer();
    QObject::connect(timerPacman, SIGNAL(timeout()), pacman, SLOT(move()));
    QObject::connect(timerBlue, SIGNAL(timeout()), bluer,
SLOT(moveMonster()));
```

```
    QObject::connect(timerOrange, SIGNAL(timeout()), oranger,
SLOT(moveMonster()));
    QObject::connect(timerRed, SIGNAL(timeout()), redr,
SLOT(moveMonster()));
    QObject::connect(timerPink, SIGNAL(timeout()), pinkr,
SLOT(moveMonster()));
    timerPacman->start(310);
    timerBluer->start(420);
    timerOranger->start(400);
    timerRedr->start(370);
    timerPinkr->start(350);
}
```

每个对象的`move`函数(对于怪兽类则**多态调用**), 是slot function, 在时钟周期会被调用, 并且接受用户的交互信息*来不断更新游戏循环.*

每个move函数里大致过程如下:

```
slot function objMove(like move(), moveMonster()...) {
    获得用户交互;
    由用户交互以及地图产生动作响应;
    更新地图;
    输出界面;
}
```

Pacman Q-learning 探索

(注: 本来想实现一个AI Pacman, 但是Python和C++交叉编译没有弄好, 也不太是高级程序设计的重点, 所以这里只说一下原理)

参考文献: http://cs229.stanford.edu/proj2017/final-reports/5241109.pdf

首先定义state、action、reward。Agent(吃豆人)会根据当前状态来采取动作，并记录被反馈的奖赏，以便下次再到相同状态时能采取更优的动作。

下面是状态和动作的定义:

- State space $\mathcal{S}$: All possible configurations in the game, including the positions of the points, positions of the ghosts, and positions of the player, etc.
- Action space $\mathcal{A}$: A set of all *allowed* actions: {left, right, up, down, or stay}.

Q为action-utility function，用于评价在特定状态下采取某个动作的优劣。它是吃豆人的记忆, 可以把Q当成是一个特定状态下特定动作得到的奖赏的映射.

接下来策略的选择 \pi 就是获得奖赏最大的那一个:

Q的值是很难优化的, 放入神经网络, 下面是loss function:

总结

最后一次课设, 有点点舍不得, 确实高程课设好有意思呀!!! 从第一次markdown转换(GUI), 到第二次动态界面的控制台pvz, 每一次都发现了新东西, 学到了新东西, 很开心!!!

谢谢助教哥的批改~ ☺