* 课程设计报告，包含选题描述、设计方案、代码模块的功能划分与描述、实现效果等。

选题描述：

贪吃蛇是一款小游戏。玩家需要控制一条小蛇，在每个时间步中，做出左转、向前或右转中的一个操作，控制小蛇前进，在一个长度为M，宽度为N的游戏环境中，不断吞吃食物，并且不撞到自己的身体或者墙壁。目标是吃掉最多的食物，尽量填满整个空间，同时在这个前提下尽量在最短的步数、最高的效率下达成填满屏幕的目标。

设计方案：

1. 环境搭建：利用pygame库，实现游戏的逻辑、规则和可视化，使得玩家能够控制小蛇移动。
2. 编写策略：小蛇应该具有如下的功能：
3. 寻路能力：能够在面对许多障碍物的情况下，找到前往某一点的最短路径。通过A\*算法解决。
4. 判断是否安全的能力：能够对自己行为的安全性进行考虑，即想象移动到下一格后是否会陷入死局。这通过 “能否找到前往自己的尾巴”来实现。同样也包括不要直接撞上身体或者墙壁的能力。
5. 恶劣环境下改变恶劣环境的能力：在无法前往食物或者无法前往尾巴的时候，往往蛇身的分布已经相当复杂，导致蛇头很容易撞到身体或者墙壁导致死亡。此时小蛇应当具有尽量整理自己的身体从而使得身体更加整齐，进而改变恶劣环境的能力。整理身体的功能通过“尽量远离尾巴”的倾向解决。
6. 摆脱循环状态的能力：由于小蛇的寻路和对未来的想象并不是全面的，所以在某些情况下会陷入循环的路径。小蛇应当具有发现自己的循环状态并摆脱循环状态的能力。通过设置小蛇的属性来判断循环状态，而后通过前往尾巴来整理身体。
7. 根据当前状态做出抉择的能力：根据当前身体的长度以及前往食物路径的长度等状态，小蛇应当做出前往食物，前往尾巴，尽量绕行来整理身体、摆脱循环等等行为。
8. 程序的可视化与运行速度等功能主要通过game类的方法实现，程序的主要功能（即小蛇的寻路、思考和判断）通过snake类的方法实现。

代码模块的功能划分和描述：

1. 设置中的各种参数：

出于方便考虑，程序在一个文件中完成，并采用全局变量在程序中共享设置的参数

参数包括地图的大小M，N；可视化部分的格子和线条大小；可视化部分的各种颜色；游戏进行的速度、频率等。

2）主函数

if \_\_name\_\_ == '\_\_main\_\_':  
 game = Game()  
 game.run()

主函数创建了一个game对象，并调用了game的run方法。程序的大部分工作在run方法中完成

3）Game类：

游戏运行主体，包括screen成员，snake成员，food成员，game\_mode成员，draw\_screen方法等

draw\_screen方法：

程序可视化的主体部分，将会调用draw\_body,draw\_lines,draw\_eyes等方法对蛇身，蛇眼，蛇的目标路径，食物进行可视化。

run方法：

程序运行的主体部分，将会调用check\_event方法对于按键操作进行检测，调用snake.think,snake.move\_and\_eat,food.check方法完成蛇的思考，移动和食物的再次生成。程序的大部分工作在think函数中完成。之后会调用draw\_screen方法进行可视化。在之后检测小蛇是否胜利或者陷入死循环，最后控制游戏的帧率/速度。

4）snake类

模拟贪吃蛇。包括着贪吃蛇具有的许多基本属性的行为。

think方法：

最重要的函数。主要调用search\_the\_road, get\_long\_road函数，对于snake对象的direction, choose, image\_road, image\_body, action等成员进行更改。内部包含着小蛇进行思考判断的全部逻辑。

get\_long\_road方法：

获取尽可能长的一条路。将会对蛇头部周围的三个格子进行判断，然后返回安全且离尾巴最远的格子。

move\_and\_eat方法：

进行小蛇移动的相关处理。对蛇身和地图上的空格子进行更新。更新整理身体的倾向，清空记忆中的历史选择。

5) search\_the\_road函数

A\*算法。若存在路径，则返回记录着最短路径信息的两个字典；若不存在路径，则返回None。使用这个函数来判断起点和终点之间是否存在路径，以此进行安全性的判断，或者获得一个最短的路径。

6） get\_image\_road函数

对于传入的两个字典中的信息进行处理，返回记录着最短路径坐标的一个列表。

7）possible\_airs函数和airs\_near\_the\_cell函数：

分别返回地图上空着的格子和在某个格子旁边的空格子。

实现效果：

一、游戏界面与流程：

进入游戏后，玩家可以面对具有三个按钮的开始界面。第一个按钮是玩家模式，玩家可以通过w a s d键来控制蛇身的转向。在一个时间步内仅能转向一次，只能选择左转，右转或直走，无法向后转身。若蛇头碰到蛇身或者墙壁，或者玩家填满了屏幕，则游戏结束，窗口关闭；第二个按钮是AI模式，小蛇将不再由玩家控制，而是自己思考并做出决策，游戏结束后，会自动开始下一轮游戏。在两个模式中，玩家都可以使用p键对游戏进行暂停，使用上下键调节游戏的速度。通过更改设置，可以选择是否显示小蛇身体上的花纹以及是否通过蛇头的颜色展现小蛇做出的决策（默认不展示）。

二、当使用AI模式时，小蛇的思考遵循如下思路：

在没有进入死循环的情况下：

1. 小蛇会首先搜索前往食物的道路。
2. 如果这样的道路不存在，那么会调用get\_long\_road方法，做出尽量远离尾巴的行为，从而走出一条比较长的路，让身体较为聚集，知道发现前往食物的路
3. 如果这样的路存在，同时吃完这个食物小蛇就铺满了屏幕，游戏胜利，那么直接吃掉食物；否者就要判断这个最短道路是否安全，如果不安全，同样开始远离尾巴
4. 如果道路安全，且没有整理身体的需求，那么直接吃掉食物；如果有整理身体的需求，此时就会开始远离尾巴，从而对身体进行整理

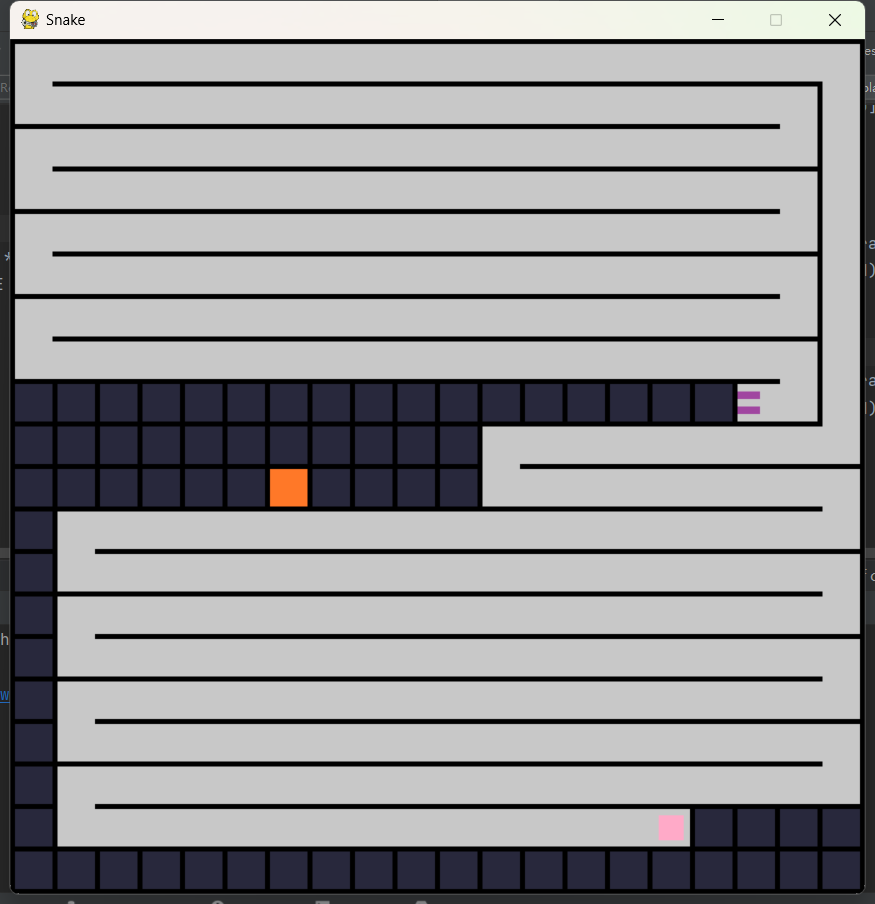
如果发现已经进入死循环，小蛇会进入随机状态，随机选择周围一个安全的格子前进，在随机前进若干步之后，退出随机状态，重新判断当前的情况

AI模式的实现效果与面临的问题：

在10 \* 10的地图中，可以以平均1500左右的步数填满屏幕，最慢也不会超过2000步，而且时长也在1.5秒左右。

但是当地图为奇数 \* 奇数时，填满屏幕的时间就会大大延长。例如当地图大小为9 \* 9时，填满屏幕平均需要3000步左右。且有几率遇到剩余两格无法填满，进入死循环的情况。

奇偶性差异的产生，是因为对于偶数 \* 偶数或者奇数 \* 偶数的地图，小蛇在远离尾巴，整理身体的过程中，会自动地形成一个规整的循环路线，将整个地图都遍历一遍。这不仅避免了死循环的出现，而且提高了效率；而当地图为奇数 \* 奇数时，这样的路线无法自然地形成，导致后期小蛇的身体杂乱无章，不仅降低效率，而且可能进入死循环。



（如图，20\*20的地图下，后期的蛇身相当规整）

对于这个情况目前还没有想到特别好的解决方法，可能可以通过在后期人工规划一个类似循环的遍历路径来解决。

同时，当格子数比较多的时候，花费的时间也会更长。例如，对于14 \* 14的地图，一般前进6000步，需要13秒左右的时间才能填满；而当地图大小为15 \* 16时，则需要前进10000步，花费将近30秒；而当地图大小为20 \* 20时，步数达到了26000步左右，时长也达到了160秒。

三、开发历程和思路：

首先写了一个字符画版本的程序，实现了贪吃蛇游戏的基本规则，进行了一些初步的探究。然后简单学习了pygame模块的使用，对程序进行了初步的可视化。

在版本v11中，小蛇只是简单的判断头旁边的格子哪一个离食物更近，然后选择最近的格子，没有使用任何算法，也不懂得躲避自己的身体或者障碍物。

在v12版本中，增加了躲避墙壁和蛇身的功能，但是由于躲避的方法只是简单的转向，且完全只是盲目的考虑头旁边的三个格子，所有依然会撞到自己或者墙壁。

在v13版本中，加入了A\*算法。此时小蛇已经可以很好地躲避墙壁和自己的身体，不会撞上去。然而由于此时的蛇只是简单粗暴地不停朝食物前进，所以当蛇头和食物间不存在路径，或者自己把蛇头包围的时候，小蛇依然会死去。

在v14版本中，让小蛇时刻检测自己是否具有前往尾部的退路，以此作为是否安全的标准，并且对于某条前往食物的道路进行想象，判断吃完食物后是否安全，从而判断是要沿着这条道路行走，还是沿着前往尾巴的最短路径行走。同时将小蛇‘前往食物’和‘前往尾巴’两种状态通过头部的绿色和红色展现出来，并将小蛇想象中的道路呈现出来。但是由于

1.“此刻是否有前往尾巴的道路”这一判断标准并不完全恰当（因为小蛇并不会立刻死亡，蛇头周围仍然存在空间，可能在行走几步之后前往尾巴的路就会出现）。

2.在不安全的时候沿着最短路径前往尾巴不是正确的做法（做法应该是尽量走最长的路，看看事情能否发生转机）。

3.A\*算法仅能返回几条最短路径中的一条，而这条路径可能不是最好的。

所以这个版本的小蛇常常在十分安全的情况下陷入“前往食物”、“追尾巴”、“前往食物”、“追尾巴”的死循环。

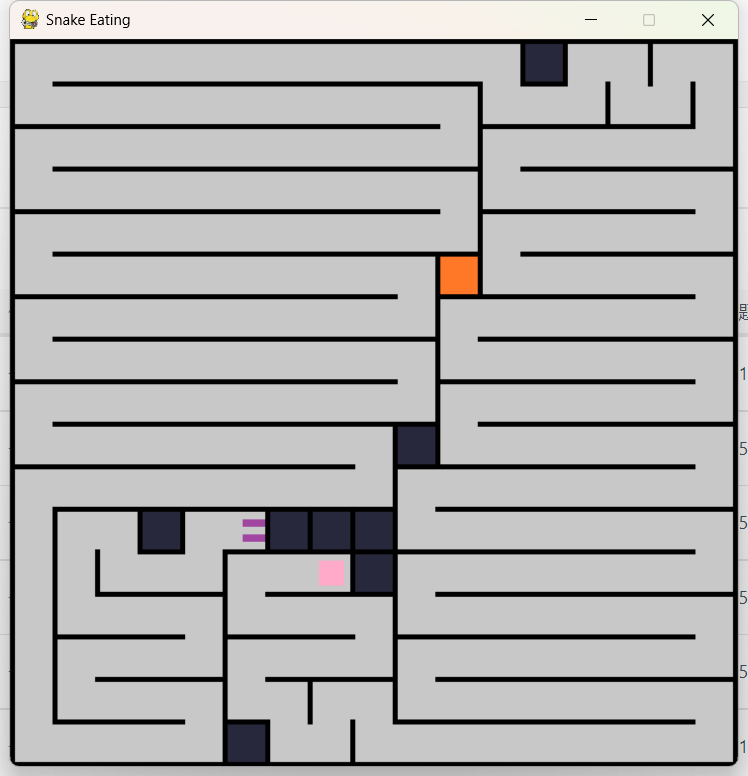
在v15版本，小蛇会在需要前往尾巴的时候尽量走出比较长的一条路，呈S型绕行，将能填满的地方填满，具体实现为：在蛇头附近（蛇的前面一截）较为笔直的情况下，倾向于保持和蛇身一样的方向；否则就倾向于保持和上一个时间步一样的方向。同时，在小蛇陷入死循环（具体表现为小蛇相当就都没有吃到食物的时候），随机选择一个安全的方向前进，直到吃到食物。这个版本的小蛇虽然效率较低，但是已经具备了几乎填满整个屏幕的能力，只是很大概率会遇到有两个空洞无法填满的情况。

V21版本：这个版本的小蛇会在需要走出较长路径的时候，选择人工在程序中构建出一个S型路径并按照这个S型路径前进。但是很快就发现这个想法在实现上具有很大的困难，随即被废弃了。

V22版本：S型路径的方案被放弃后，在网上查看他人的解法，并发现了“每一步都选择尽量远离尾巴的格子”这一“贪婪”的算法。实践之后发现效果很不错，很大程度上实现了“最远路径”这个功能，可以对身体进行很好的整理。于是改写了小蛇思考的函数，使其在长度很长时，不前往食物，始终沿着最长路径前进；蛇身填满屏幕一半后，仅在食物很近的时候前往食物，其余时候沿最长路径前进；蛇身较短时，仅在食物很远的时候沿最长路径前进。并期望通过调整每个阶段的划分指标来提高效率。

V23版本：发现在游戏还是会经常碰到循环（包括由于设计缺陷导致的假循环，即明明有一条合理的道路却不会前往，而是陷入循环状态；以及真循环，即陷入了无法解决的、除了循环别无他路的死局）。所以在循环时让小蛇沿着前往某个随机点的最长路径前进，期望通过随机来解决循环的问题。但是效果不佳。

V24版本：运行流程方面，在每次胜利之后自动重新开始游戏，并打印出上一局花费的步数；可视化方面，为小蛇加上了眼睛，将小蛇的每一节身体连接起来，从而展示出小蛇的运动轨迹；思考逻辑方面，一方面加入了choose属性，使得小蛇能够记住自己前面所做出的选择，并且在很长一段时间都没有吃到食物（即陷入循环）的情况下，倾向于做出和之前不同的选择，从而摆脱循环；另一方面让小蛇在长度较短的时候不整理身体（即在可能的时候总是会向食物前进），在较长的时候每吃掉一个食物都会整理自己的身体，身体越长，会整理得越久。这些思考逻辑方面的改变使得小蛇能够尽量避免在陷入假性循环的情况下尽量摆脱，同时尽量整理好自己的身体避免陷入真正的循环死局。



（19 \* 19的地图中的一种死局）

V25版本：尝试通过将奇数\*奇数的地图转换为包含偶数的地图，从而解决容易陷入死局的情况。具体实现为：以宽度和高度均为19的地图为例，若食物位于最上方一排，则在寻路时不考虑最下面的一排格子；若食物位于最下方一排，则不考虑最上方的格子，从而将地图近似地转换为了宽为19，高为18的地图，使得身体能够尽量的规整。然而即使这样，最终还是常常陷入死局，同时这个做法也会带来一些问题，因而这个思路被舍弃了。

最终版本：增加了游戏的开始页面与教程，增加了玩家控制游玩的模式；在AI模式中，每一局结束后，都会自动进行下一局，并且输出这一局消耗的步数、时间，最大步数和最小步数，总体的平均步数，直到完成75局为止。

尚存在的问题：

1. 效率还没有达到最高。一方面需要进一步调整小蛇的决策逻辑，在宏观行为上优化小蛇寻找食物的效率；另一方面要对程序本身进行一定的优化，对代码结构进行调整，删除一些不必要的运算。
2. 依然会陷入死局。这个问题可以考虑将奇数地图转换为偶数地图的思路（例如空出一块3\*3的区域，使得其他剩余区域成为偶数），也可以考虑让小蛇尽量避免产生“空洞”，即蛇身尽量不要将一块区域包围起来，还可以考虑让小蛇在后期沿着某个设计好的固定的路线前进。
3. UI界面太过丑陋。交互不够丰富。