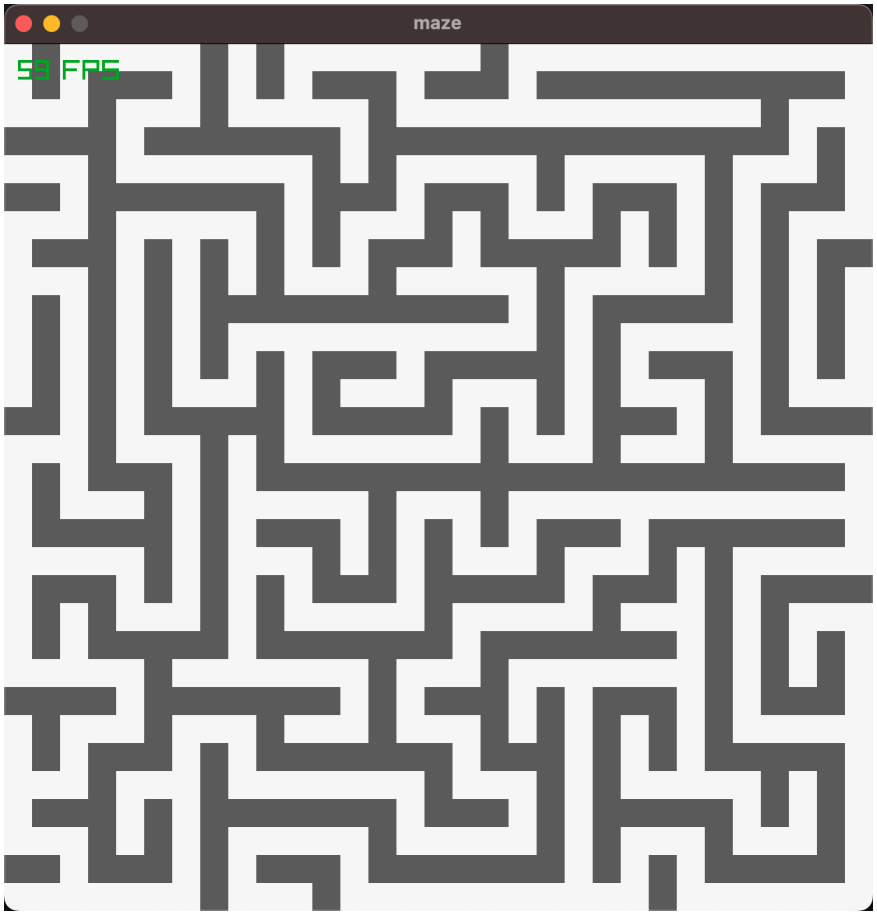
Challenge 3 迷宫可视化

本次challenge需要同学们使用raylib将第二题中的迷宫进行可视化，具体要求为：

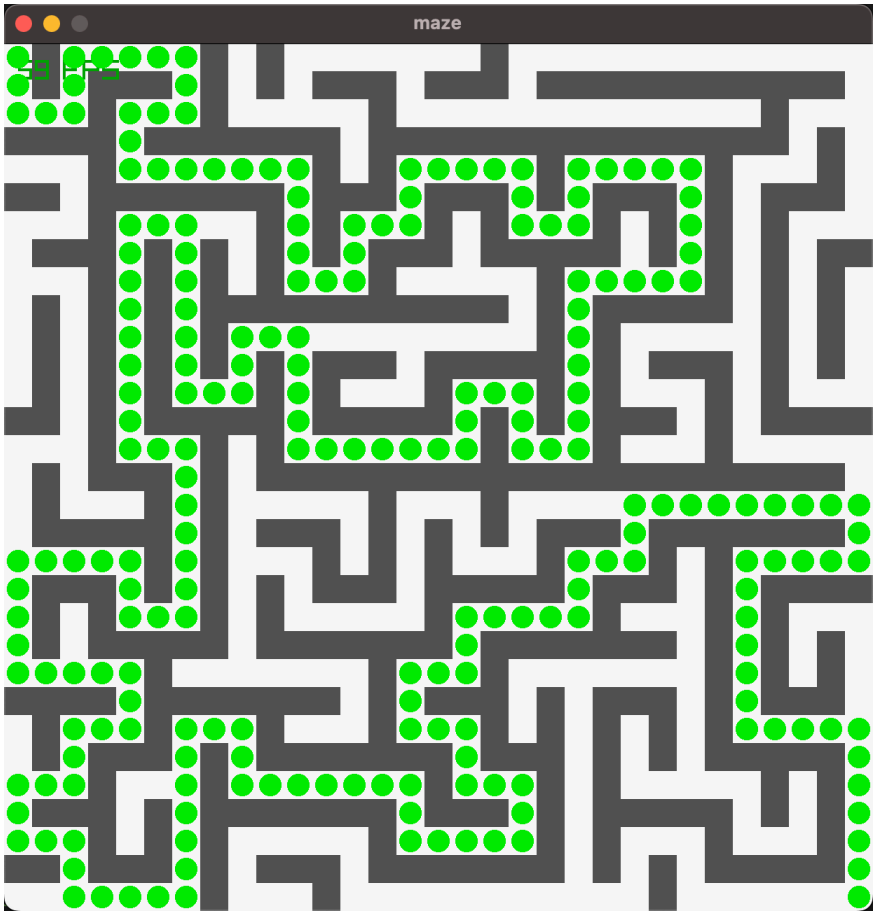
1. **打印出如下图所示的迷宫（2分）**



输入中迷宫的格式与第二题的一致，上图中灰色部分表示墙`W`，白色部分表示通道`.`，墙壁和通道的展示方式不做要求。

2. **画出一条从迷宫起点到终点的路径（2分）**

规定迷宫的起点为左上角，终点为右下角，同学们需要展示求解完成后的一条从起点到终点的路径，路径的展示方式（下图为绿色点序列）不做要求。



3. **动态地展示出迷宫求解的过程（6分）**

同学们需要像canvas上展示的示例一样画出迷宫求解的动态过程，展示出求解算法是如何一步一步从起点到达终点，以及中间过程是如何尝试其他的路径。求解算法和展示的速度（不能太快导致无法看清）不做要求。

提示

使用raylib展示一般会先设置帧率，然后在主循环里面渲染内容，程序框架如下：

```
SetTargetFPS(60);               // Set our game to run at 60 frames-per-second
// Main game loop
while (!WindowShouldClose())    // Detect window close button or ESC key
{
  // Draw
  BeginDrawing();
  // Draw something
  EndDrawing();
}
```

在 raylib 里，`SetTargetFPS(int fps)` 用于设置游戏（程序）的目标帧率（Frames Per Second，FPS）。它会限制 `raylib` 主循环的更新速度，使游戏运行在指定的 FPS 之下，而不会无限制地占用 CPU 资源。`SetTargetFPS(fps)` 内部会计算每帧的最小时间间隔 `frameTime = 1.0 / fps`，然后在 `EndDrawing()` 之后调用 `WaitTime()`，让 CPU 休眠一段时间，以保证每帧的时间至少等于 `frameTime`。如果游戏逻辑和渲染速度快于 `frameTime`，`raylib` 会让 CPU 休眠，避免不必要的资源消耗。如果逻辑处理时间超出了 `frameTime`，则不会等待，游戏会变慢（FPS 降低）。

所以你可以在每一帧求解一次路径，比如说如果你使用广度优先搜索，在每一帧中你可以访问当前队列头部的点，下一帧再访问下一个点，在每一帧画出起点到当前访问到的路径，从而展示出动态求解的效果。

输入输出格式

输入的第一行为`n m`，分别表示迷宫的行数，列数，第二行开始输入`n`行`m`列的迷宫，本题提供的迷宫样例都存在一条从起点到终点的路径。

作业评测标准

助教会选择`data`目录下面的若干个迷宫样例运行你的程序，根据三个要求完成的情况进行给分。

提交格式

```
<your student number>.zip
|- main.cpp
|- utilities.h
```