下面是一个简单的步骤，帮助你开始编写这个迷宫游戏：

1. 生成迷宫：

首先，你需要编写一个算法来生成迷宫。确保生成的迷宫有解，即从起点到终点有一条通路。

生成迷宫的算法：

1. 递归分割算法（Recursive Division）： 这是一种基于递归的分治算法，通过递归地将迷宫分割成较小的子区域，并在分割的过程中创建墙壁来生成迷宫。

2. 随机 Prim 算法（Randomized Prim's Algorithm）： 这是一种基于最小生成树的随机算法，从一个起始点开始，随机选择相邻的未访问过的格子，并将其加入迷宫中，同时将墙壁打通，直到所有格子都被访问过。

3. Kruskal's Algorithm：这是一种最小生成树算法，它将所有的墙壁按照权重（或者随机顺序）排序，然后逐个加入迷宫中，确保不形成环，直到所有的单元格都联通为止。

4. 随机深度优先搜索（Randomized Depth-First Search）：\*这是一种基于深度优先搜索的随机算法，从一个起始点开始，随机选择一个相邻的未访问过的格子，并将其加入迷宫中，同时将墙壁打通，然后递归地对新加入的格子进行深度优先搜索，直到所有格子都被访问过。

2. 绘制迷宫界面：

使用 Qt 的图形框架来绘制迷宫界面。你可以使用 QGraphicsView 和 QGraphicsScene 来显示迷宫，将迷宫的格子表示为图形项，起点和终点可以用不同的颜色来标识。

3. 处理玩家操作：

实现键盘事件或鼠标事件的处理函数，让玩家能够操作角色在迷宫中移动。你可以在迷宫中使用一个图形项来表示玩家的角色，根据玩家的操作来更新角色的位置。

4. 选择算法：

添加一个界面或者菜单，让玩家可以选择使用不同的算法来帮助解决迷宫。

解决迷宫问题的算法：

1.深度优先搜索（Depth-First Search，DFS）： 这是一种图的遍历和搜索算法，通过递归地沿着一条路径搜索直到无法继续，然后回溯到上一个节点，继续搜索其他路径，直到找到终点为止。

2.广度优先搜索（Breadth-First Search，BFS）： 这是一种图的遍历和搜索算法，通过逐层扩展搜索直到找到终点为止，它通常能够找到最短路径。

3.Dijkstra 算法： 这是一种解决单源最短路径问题的贪心算法，可以在非负权重的图中找到从单个源点到其他所有顶点的最短路径。

4.Floyd 算法：

5.A\* 算法： 这是一种启发式搜索算法，结合了广度优先搜索和启发式估计函数，可以在图中找到从起点到终点的最短路径，通常比 Dijkstra 算法更快。

5. 显示路径：

当玩家选择了算法之后，使用该算法来求解迷宫，并将求解的路径显示在迷宫界面上。你可以使用不同的颜色或者线条来表示路径。

6. 测试和优化：

完成基本功能之后，进行测试并不断优化游戏的体验。你可以添加一些额外的功能或者效果，增加游戏的趣味性和挑战性。