

// maze construction

insertAndSetFirstWall(&head, 1, OVERALL\_WINDOW\_WIDTH/2-300, OVERALL\_WINDOW\_HEIGHT/2-200, 5, OVERALL\_WINDOW\_HEIGHT/2+165);

insertAndSetFirstWall(&head, 2, OVERALL\_WINDOW\_WIDTH/2-300, OVERALL\_WINDOW\_HEIGHT/2-200, 600, 5);

insertAndSetFirstWall(&head, 3, OVERALL\_WINDOW\_WIDTH/2+300, OVERALL\_WINDOW\_HEIGHT/2-200, 5, OVERALL\_WINDOW\_HEIGHT/2-100);

insertAndSetFirstWall(&head, 4, OVERALL\_WINDOW\_WIDTH/2-300, OVERALL\_WINDOW\_HEIGHT/2+200, 605, 5);

insertAndSetFirstWall(&head, 5, OVERALL\_WINDOW\_WIDTH/3-140, OVERALL\_WINDOW\_HEIGHT/2-150, 5, OVERALL\_WINDOW\_HEIGHT/2-50);

insertAndSetFirstWall(&head, 6, OVERALL\_WINDOW\_WIDTH/3-140, OVERALL\_WINDOW\_HEIGHT/2, 50, 5);

insertAndSetFirstWall(&head, 7, OVERALL\_WINDOW\_WIDTH/3-90, OVERALL\_WINDOW\_HEIGHT/2, 5, OVERALL\_WINDOW\_HEIGHT/2-150);

insertAndSetFirstWall(&head, 8, OVERALL\_WINDOW\_WIDTH/3-135, OVERALL\_WINDOW\_HEIGHT/2+90, 50, 5);

insertAndSetFirstWall(&head, 9, OVERALL\_WINDOW\_WIDTH/3-140, OVERALL\_WINDOW\_HEIGHT/2+90, 5, OVERALL\_WINDOW\_HEIGHT/2-125);

insertAndSetFirstWall(&head, 10, OVERALL\_WINDOW\_WIDTH/3-140, OVERALL\_WINDOW\_HEIGHT/2-100, 190, 5);

insertAndSetFirstWall(&head, 11, OVERALL\_WINDOW\_WIDTH/3-80, OVERALL\_WINDOW\_HEIGHT/2-200, 5, OVERALL\_WINDOW\_HEIGHT/2-185);

insertAndSetFirstWall(&head, 12, OVERALL\_WINDOW\_WIDTH/3-25, OVERALL\_WINDOW\_HEIGHT/2-155, 5, OVERALL\_WINDOW\_HEIGHT/2-185);

insertAndSetFirstWall(&head, 13, OVERALL\_WINDOW\_WIDTH/3+40, OVERALL\_WINDOW\_HEIGHT/2-150, 320, 5);

insertAndSetFirstWall(&head, 14, OVERALL\_WINDOW\_WIDTH/3+355, OVERALL\_WINDOW\_HEIGHT/2-150, 5, OVERALL\_WINDOW\_HEIGHT/2-185);

insertAndSetFirstWall(&head, 15, OVERALL\_WINDOW\_WIDTH/3+360, OVERALL\_WINDOW\_HEIGHT/2-60, 52, 5);

insertAndSetFirstWall(&head, 16, OVERALL\_WINDOW\_WIDTH/3+355, OVERALL\_WINDOW\_HEIGHT/2-60, 5, OVERALL\_WINDOW\_HEIGHT/2-140);

insertAndSetFirstWall(&head, 17, OVERALL\_WINDOW\_WIDTH/3+100, OVERALL\_WINDOW\_HEIGHT/2-150, 5, OVERALL\_WINDOW\_HEIGHT/2-140);

insertAndSetFirstWall(&head, 18, OVERALL\_WINDOW\_WIDTH/3-80, OVERALL\_WINDOW\_HEIGHT/2-50, 185, 5);

insertAndSetFirstWall(&head, 19, OVERALL\_WINDOW\_WIDTH/3-25, OVERALL\_WINDOW\_HEIGHT/2-50, 5, OVERALL\_WINDOW\_HEIGHT/2-50);

insertAndSetFirstWall(&head, 20, OVERALL\_WINDOW\_WIDTH/3-80, OVERALL\_WINDOW\_HEIGHT/2+140, 60, 5);

insertAndSetFirstWall(&head, 21, OVERALL\_WINDOW\_WIDTH/3-20, OVERALL\_WINDOW\_HEIGHT/2+90, 50, 5);

insertAndSetFirstWall(&head, 22, OVERALL\_WINDOW\_WIDTH/3+30, OVERALL\_WINDOW\_HEIGHT/2+40, 5, OVERALL\_WINDOW\_HEIGHT/2-185);

insertAndSetFirstWall(&head, 23, OVERALL\_WINDOW\_WIDTH/3+30, OVERALL\_WINDOW\_HEIGHT/2+40, 330, 5);

insertAndSetFirstWall(&head, 24, OVERALL\_WINDOW\_WIDTH/3+355, OVERALL\_WINDOW\_HEIGHT/2-60, 5, OVERALL\_WINDOW\_HEIGHT/2-140);

insertAndSetFirstWall(&head, 25, OVERALL\_WINDOW\_WIDTH/3, OVERALL\_WINDOW\_HEIGHT/2-50, 310, 5);

insertAndSetFirstWall(&head, 26, OVERALL\_WINDOW\_WIDTH/3+310, OVERALL\_WINDOW\_HEIGHT/2-100, 5, OVERALL\_WINDOW\_HEIGHT/2-140);

insertAndSetFirstWall(&head, 27, OVERALL\_WINDOW\_WIDTH/3+150, OVERALL\_WINDOW\_HEIGHT/2-110, 5, OVERALL\_WINDOW\_HEIGHT/2-215);

insertAndSetFirstWall(&head, 28, OVERALL\_WINDOW\_WIDTH/3+150, OVERALL\_WINDOW\_HEIGHT/2-110, 120, 5);

insertAndSetFirstWall(&head, 29, OVERALL\_WINDOW\_WIDTH/3+270, OVERALL\_WINDOW\_HEIGHT/2-110, 5, OVERALL\_WINDOW\_HEIGHT/2-215);

insertAndSetFirstWall(&head, 30, OVERALL\_WINDOW\_WIDTH/3+150, OVERALL\_WINDOW\_HEIGHT/2-90, 125, 5);

insertAndSetFirstWall(&head, 31, OVERALL\_WINDOW\_WIDTH/3+40, OVERALL\_WINDOW\_HEIGHT/2, 275, 5);

insertAndSetFirstWall(&head, 32, OVERALL\_WINDOW\_WIDTH/2+300, OVERALL\_WINDOW\_HEIGHT/2-200, 5, OVERALL\_WINDOW\_HEIGHT/2-100);

int i = 0;

while ( i < 30 ) {

insertAndSetFirstWall(&head, 33,

OVERALL\_WINDOW\_WIDTH/2-127+ (i \* 2),

OVERALL\_WINDOW\_HEIGHT/2+90 + (i \* 2),

5, 5);

i++;

}

i = 0;

while ( i < 30 ) {

insertAndSetFirstWall(&head, 33,

OVERALL\_WINDOW\_WIDTH/2-20 + (i \* 2),

OVERALL\_WINDOW\_HEIGHT/2+138 + (i \* 2),

5, 5);

i++;

}

i = 0;

while ( i < 30 ) {

insertAndSetFirstWall(&head, 33,

OVERALL\_WINDOW\_WIDTH/2-50 + (i \* 2),

OVERALL\_WINDOW\_HEIGHT/2+40 + (i \* 2),

5, 5);

i++;

}

int j = 33;

for (int i=-30; i<31; i+=1){

insertAndSetFirstWall(&head, j, OVERALL\_WINDOW\_WIDTH/2+100+i, OVERALL\_WINDOW\_HEIGHT/2+sqrt(900-i\*i)+170, 5, 5);

insertAndSetFirstWall(&head, j, OVERALL\_WINDOW\_WIDTH/2+100+i, OVERALL\_WINDOW\_HEIGHT/2-sqrt(900-i\*i)+170, 5, 5);

j+=1;

}

for (int i=-30; i<31; i+=1){

insertAndSetFirstWall(&head, j, OVERALL\_WINDOW\_WIDTH/2+35+i, OVERALL\_WINDOW\_HEIGHT/2+sqrt(900-i\*i)+80, 5, 5);

insertAndSetFirstWall(&head, j, OVERALL\_WINDOW\_WIDTH/2+35+i, OVERALL\_WINDOW\_HEIGHT/2-sqrt(900-i\*i)+80, 5, 5);

j+=1;

}

for (int i=-50; i<51; i+=1){

insertAndSetFirstWall(&head, j, OVERALL\_WINDOW\_WIDTH/2+200+i, OVERALL\_WINDOW\_HEIGHT/2+sqrt(2500-i\*i)+50, 5, 5);

insertAndSetFirstWall(&head, j, OVERALL\_WINDOW\_WIDTH/2+215+i, OVERALL\_WINDOW\_HEIGHT/2-sqrt(2500-i\*i)+190, 5, 5);

j+=1;

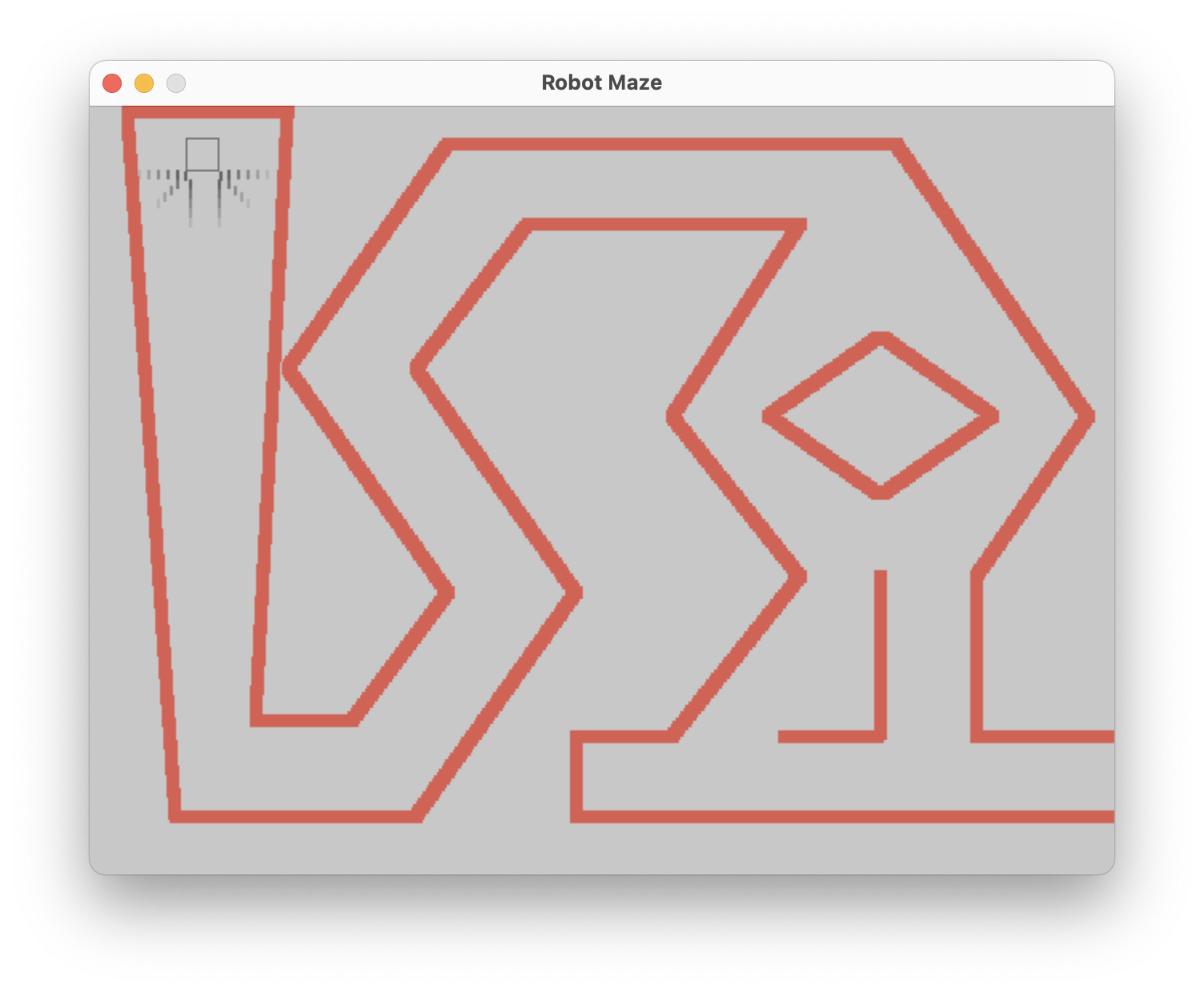
}

insertAndSetFirstWall(&head, j, OVERALL\_WINDOW\_WIDTH/2+300, OVERALL\_WINDOW\_HEIGHT/2, 5, OVERALL\_WINDOW\_HEIGHT/2-40);

**// robot starting position**

robot->true\_x = OVERALL\_WINDOW\_WIDTH/2-280;

robot->true\_y = OVERALL\_WINDOW\_HEIGHT-80;



Add in wall.h:

int insertAndSetDiagonalWall(struct Wall\_collection \*\* head, int key, int x1, int y1, int x2, int y2);

Add in wall.c:

// Draws a line based upon a given starting coordinate (x1, y1), and a ending coordinate (x2, y2)

// NOTE: positions are not exactly accurate (give 2-3 pixel) because of the size of sqaures drawn

int insertAndSetDiagonalWall(struct Wall\_collection \*\* head, int key, int x1, int y1, int x2, int y2) {

// x, y = the amount each sqaure will towards destination move each loop

// loop = amount of iteration of the for

double x, y, loop;

// absolute value of the differences of the x values and y values

double absX = sqrt((x1-x2)\*(x1-x2));

double absY = sqrt((y1-y2)\*(y1-y2));

// the biggest difference has the corresponding x/y value set to 2 (largest difference),

// the remaining is set to a ratio that allows x/y to reach destination at same time

if (absX >= absY){

x = 2;

loop = absX/2;

y = absY/loop;

}

else {

y = 2;

loop = absY/2;

x = absX/loop;

}

// Uncomment if you want to see these values manually

// printf("%d %d\n", x1, y1);

// printf("%d %d\n", x2, y2);

// printf("%f %f\n", absX, absY);

// printf("%f %f\n", x, y);

// Makes sure the line is drawn in the correct direction based on each points

// relative position to eachother

int xNeg = 1;

int yNeg = 1;

if (x1 >= x2) {

xNeg = -1;

}

if (y1 >= y2) {

yNeg = -1;

}

// Draws each square until it reaches the given ending coordinate

for (int i=0; i < loop; i++) {

insertAndSetFirstWall(head, key + i, x1 + i\*x\*xNeg, y1 + i\*y\*yNeg, 8, 8);

}

// returns number of squares drawn so it is easy to find next key needed

return (int)floor(loop + key);;

}

In robot.c, change the setup\_robot function to these instead:

// Custom maze position

robot->x = 60;

robot->y = 20;

robot->true\_x = 60;

robot->true\_y = 20;

robot->width = ROBOT\_WIDTH;

robot->height = ROBOT\_HEIGHT;

robot->direction = 0;

robot->angle = 180;

robot->currentSpeed = 0;

robot->crashed = 0;

robot->auto\_mode = 0;

robot->state = Start\_state;

In main.c, change the checkRobotReachedEnd function in the if statement to this:

if (checkRobotReachedEnd(&robot, 620, 390, 20, 50)){

Finally, in main.c add our custom maze :D! :

// Wall behind robot

double key\_num = insertAndSetDiagonalWall(&head, 1, 20, 0, 120, 0);

// First corridor, getting smaller

key\_num = insertAndSetDiagonalWall(&head, 1, 20, 0, 50, 440);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 120, 0, 100, 380);

// small path on bottom

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 50, 440, 200, 440);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 100, 380, 160, 380);

// first zig zag

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 200, 440, 300, 300);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 160, 380, 220, 300);

// second zig zag

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 300, 300, 200, 160);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 220, 300, 120, 160);

// third zig zag

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 200, 160, 270, 70);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 120, 160, 220, 20);

// long corridor on top

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 270, 70, 440, 70);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 220, 20, 500, 20);

// top diamond, top

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 440, 70, 360, 190);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 500, 20, 620, 190);

// top diamond, bottom

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 360, 190, 440, 290);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 620, 190, 550, 290);

// bottom triangle

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 440, 290, 360, 390);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 550, 290, 550, 390);

// top of bottom corridor

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 360, 390, 300, 390);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 550, 390, 650, 390);

// side of bottom corridor

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 300, 390, 300, 440);

// bottom of bottom corridor

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 300, 440, 650, 440);

// middle diamond, top

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 420, 190, 490, 140);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 560, 190, 490, 140);

// middle diamond, bottom

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 420, 190, 490, 240);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 560, 190, 490, 240);

// right angle

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 430, 390, 490, 390);

key\_num = insertAndSetDiagonalWall(&head, key\_num+1, 490, 290, 490, 390);



**For main.c**

void setup\_robot(struct Robot \*robot){

robot->x = 50;

robot->y = OVERALL\_WINDOW\_HEIGHT-50;

robot->true\_x = 50;

robot->true\_y = OVERALL\_WINDOW\_HEIGHT-50;

robot->width = ROBOT\_WIDTH;

robot->height = ROBOT\_HEIGHT;

robot->direction = 0;

robot->angle = 0;

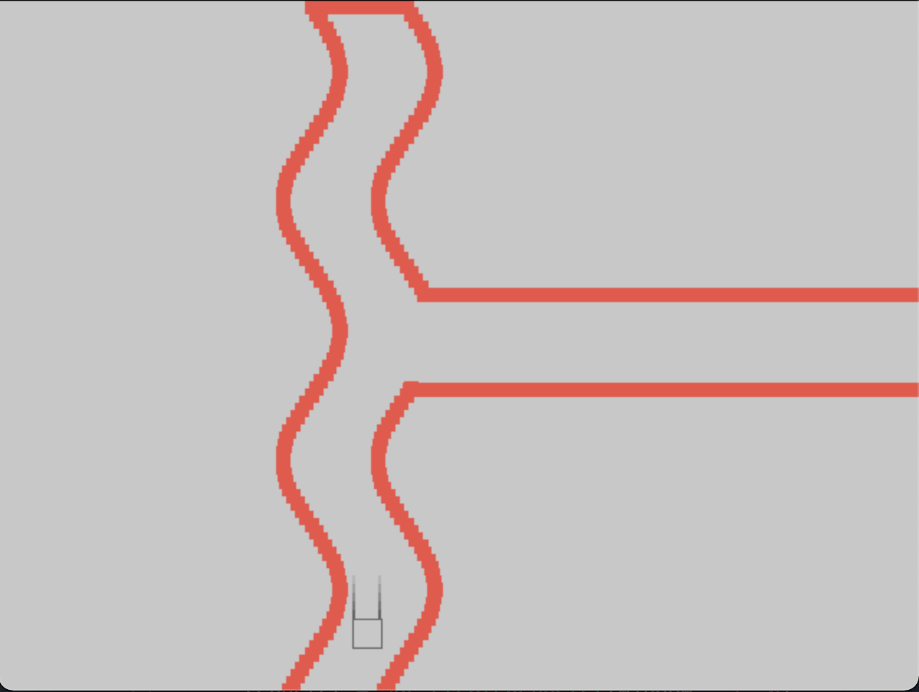
robot->currentSpeed = 0;

robot->crashed = 0;

robot->auto\_mode = 0;

printf("Press arrow keys to move manually, or enter to move automatically\n\n");

}



main.c changes

#include "math.h"

int i = 0;

double j;

while ( i < OVERALL\_WINDOW\_HEIGHT/4) {

j = i;

insertAndSetFirstWall(&head, i+1,

// the most important bit is below.

// increase the 20 for a tighter bend

// descrease for a more meandering flow

OVERALL\_WINDOW\_WIDTH/ 3 + 20\*sin(10\*j \* M\_PI/180),

// increase the 5 for a spacier curve

(i \* 5),

10, 10);

i++;

}

i = 0;

while ( i < OVERALL\_WINDOW\_HEIGHT/4) {

if (i > OVERALL\_WINDOW\_HEIGHT/12 && i < OVERALL\_WINDOW\_HEIGHT/9) {

i++;

continue;

}

j = i;

insertAndSetFirstWall(&head, i+1,

// 66 is the gap between the walls.

OVERALL\_WINDOW\_WIDTH/ 3 + 20\*sin(10\*j \* M\_PI/180) + 66,

(i \* 5),

10, 10);

i++;

}

insertAndSetFirstWall(&head, 60,

OVERALL\_WINDOW\_WIDTH/3 + 20\*sin(10\*OVERALL\_WINDOW\_HEIGHT/12 \* M\_PI/180) + 66,

5\*OVERALL\_WINDOW\_HEIGHT/12,

350, 10);

insertAndSetFirstWall(&head, 61,

OVERALL\_WINDOW\_WIDTH/3 + 20\*sin(10\*OVERALL\_WINDOW\_HEIGHT/9 \* M\_PI/180) + 66,

5\*OVERALL\_WINDOW\_HEIGHT/9,

360, 10);

insertAndSetFirstWall(&head, 62,

OVERALL\_WINDOW\_WIDTH/3 + 20\*sin(0),

0,

66, 10);

if (checkRobotReachedEnd(&robot, OVERALL\_WINDOW\_WIDTH, 35\*OVERALL\_WINDOW\_HEIGHT/72, 10, 100)){

end\_time = clock();

msec = (end\_time-start\_time) \* 1000 / CLOCKS\_PER\_SEC;

robotSuccess(&robot, msec);

}

## robot.c changes

void setup\_robot(struct Robot \*robot){

robot->x = OVERALL\_WINDOW\_WIDTH/3 + 33;

robot->y = OVERALL\_WINDOW\_HEIGHT-50;

robot->true\_x = OVERALL\_WINDOW\_WIDTH/3 + 33;

robot->true\_y = OVERALL\_WINDOW\_HEIGHT-50;

robot->width = ROBOT\_WIDTH;

robot->height = ROBOT\_HEIGHT;

robot->direction = 0;

robot->angle = 0;

robot->currentSpeed = 0;

robot->crashed = 0;

robot->auto\_mode = 0;

printf("Press arrow keys to move manually, or enter to move automatically\n\n");

}