TetRLs

In TetRLs you play an unidentified character who is apparently trapped in the laboratory of a mad soviet scientist, forced to participate in some sort of bizarre mind-game experiment that involves a lot of arranging of oddly shaped crates. However, you must balance this task with battling dozens of highly aggressive lab rats that apparently escaped (or were released...).

TetRLs' gameplay is relatively simple for a roguelike, movement keys supported (on Standard US QWERTY keyboards) are: "vi-keys", "wasd", "arrow keys" and "numpad". You may choose to use whichever scheme you wish at any time though use of the arrow keys is discouraged because that scheme does not provide easy access to diagonal movement keys.

To clear each level the player must place a certain number of tetrominos on the board while trying not to get killed by some lab-rats that appear over time. The player may attack and kill the lab rats with a wrench by "bumping" into them, otherwise the level may eventually become to cluttered by rats to be playable.

Should the player get too close to a rat for too long, the rat may bite and injure the player, reducing their HP level. The only way to increase HP after being attacked is to place more tetrominos. Particularly, the player receives a bonus amount of hit points if they

// tetrls.h listing begins:

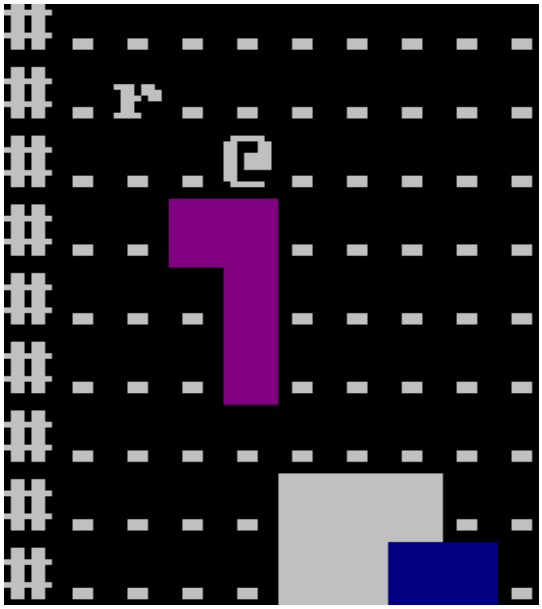
```
#define FLOOR 0
#define WALL 1
#define CLOSED_DOOR 2
#define OPEN_DOOR 3

#define RED_BLOCK 4
#define WHITE_BLOCK 5
#define MAGEN_BLOCK 6
#define BLUE_BLOCK 7
#define GREEN_BLOCK 8
#define YELLOW_BLOCK 9
#define CYAN_BLOCK 10
```

// map.h listing begins:

```
#ifndef __MAP_
#define __MAP_

#include <vector>

using namespace std;

class map
{
  public:
  vector<vector<short int> > data;
  unsigned int height, width;

  map(unsigned int new_h, unsigned int new_w, int fill);

  void resize(int new_h, int new_w);
  void fill(int fill);
};
#endif
```

// crate.h listing begins:

```
#ifndef __CRATE_
#define __CRATE_

#include <vector>
#include "map.h"

using namespace std;

class crate
{
  public:
  vector<map* > shape;
  int x, y;
  int rotation;
  int color;
  string type;

  crate(string temp);
};

#endif
```



// Listings continued on next page…

// crate.cpp listing begins:

```
#include <vector>
#include <fstream>
#include "crate.h"

using namespace std;

  crate::crate(string temp)
  {
    type = temp;
    string filename = "data/" + type + ".tet";
    ifstream fin(filename.c_str());

    fin >> color;

    for(int i=0; i<4; i++)
    {
      shape.push_back(new map(4,4,0));

      for(int t=0; t<4; t++)
      for(int n=0; n<4; n++)
        fin >> shape[i]->data[t][n];
    }

    rotation=0;
  }
```

// mapcpp listing begins:

```
#include <vector>
#include "map.h"

using namespace std;

  map::map(unsigned int new_h, unsigned int new_w, int fill)
  {
    vector<short int> blarg;

    for(unsigned int n=0; n<new_w; n++)
      blarg.push_back(fill);
    for(unsigned int n=0; n<new_h; n++)
      data.push_back(blarg);

    height = new_h;
    width = new_w;

    return;
  }

  void map::fill(int fill)
  {
    for(unsigned int n=0; n<height; n++)
      for(unsigned int i=0; i<width; i++)
        data[n][i] = fill;

    return;
  }
```

// main.cpp listing begins:

```
#include <curses.h>
#include <vector>
#include <fstream>
#include <string>
#include <cstdlib>
```

// Listing continued on next page…

complete an entire line with the tetrominos.

Once the player has placed the proper number of tetrominos, the door on the right of the room will open up and upon reaching it the player will enter the next level (and be unable to return). As levels progress the required tetrominos need to advance increases, as does the rate of lab rat generation.

The game is complete when the player finishes 5 levels (75 tetrominos total).

In addition to the movement keys, you'll need to use a few other keys:

Use the period or the number 5 key to take a turn without moving.

To return a tetromino to it's original location and rotate it use 'R' (That's shift-'r', as just normal 'r' is already used for movement) or 'x'.

To fix a tetromino in place and drop another onto the board use 'D' (That's shift-'d') or the space key.

Press 'Q' to quit

TetRLs was written by John L. Greco for the 2009 7DRL competition.

// Listing continued from previous page

```
#include <ctime>
#include "map.h"
#include "crate.h"
#include "tetrls.h"

using namespace std;

void move_player(int direction);
void load_new_level();
bool is_crate_there(int x, int y);
bool push_crate(int direction);
bool attack(int x, int y);
void plant_piece();
void run_rats();
void create_piece();
void rotate_recenter();
void redraw();
void victory();

struct monster
{
  int x, y;
  int hp;
};

enum direction { UP, RIGHT, DOWN, LEFT , UPLEFT, UPRIGHT, DOWNLEFT, DOWNRIGHT } ;

unsigned int height, width;
map* current_map;
monster* player;
crate* current_crate;
int placed;
int target;
int filled_lines;
int xoffset;
int level_count;

vector<monster* > monsters;

vector<string > message;

int main()
{
  srand(time(NULL));

  initscr();
  raw();
  keypad(stdscr, TRUE);
  noecho();
  curs_set(0);
  start_color();
  use_default_colors();

  init_pair(1, -1, COLOR_RED);
  init_pair(2, -1, COLOR_WHITE);
  init_pair(3, -1, COLOR_MAGENTA);
  init_pair(4, -1, COLOR_BLUE);
  init_pair(5, -1, COLOR_GREEN);
  init_pair(6, -1, COLOR_YELLOW);
  init_pair(7, -1, COLOR_CYAN);

  target = 0;
  level_count = 0;
```

// Listing continued on next page…

// Listing continued from previous page

```
  xoffset = -11;
  player = new monster;
  load_new_level();
  player->hp = 10;

  mvprintw(0,0,"Ah ha!  I see you have finally awaken comrade!\n" \
    "Now we can begin ze experiment!\n\n" \
    "You will find next to you a nice hefty wrench and\n" \
    "a remote control device.  Your task is quite simple,\n" \
    "you must push a certain number of oddly shaped crates\n" \
    "into each room.  Once you have positioned a crate to\n" \
    "your liking you may signal me to drop another(D).\n" \
    "However, once you do zis you may no longer move any\n" \
    "previous crates.\n\nAdditionally, if you wish to have a crate rotated\n");
  mvprintw(12,0,"and reset to its starting location you can use your remote\n" \
    "to signal me(R).\n\n\nOh, and one last zing before we begin comrade!\n" \
    "A significate population of rats seems to have recently\n" \
    "escaped from my lab, if you run into any give zem a good\n" \
    "wack with ze wrench I left with you!  If you happen\n" \
    "to be injured I will dispense a medpack to you whenever\n" \
    "I drop a new crate.");
  refresh();
  getchar();
  erase();

  message.push_back("Now, lets begin! ");

  int key;
  redraw();
  refresh();

  while(key != 'Q')
  {
    if(player->hp <= 0)
      break;

    key = getch();

    switch(key)
    {
      case 'h': case 'a': case '4': case KEY_LEFT:
        move_player(LEFT);
        break;
      case 'l': case 'd': case '6': case KEY_RIGHT:
        move_player(RIGHT);
        break;
      case 'k': case 'w': case '8': case KEY_UP:
        move_player(UP);
        break;
      case 'j': case 's': case '2': case KEY_DOWN:
        move_player(DOWN);
        break;

      case 'y': case 'q': case '7':
        move_player(UPLEFT);
        break;
      case 'u': case 'e': case '9':
        move_player(UPRIGHT);
        break;
      case 'b': case 'z': case '1':
        move_player(DOWNLEFT);
        break;
      case 'n': case 'c': case '3':
```

// Listing continued on next page…

// Listing continued from previous page

```
        move_player(DOWNRIGHT);
        break;

      case '.': case '5':
        break;

      case 'D': case ' ':
        if(player->x >= 4 && player->x <= 7 && player->y >= 2 && player->y <= 5)
          message.push_back("Move out of ze way comrade! ");
        else
          plant_piece();
        break;
      case 'R': case 'x':
        if(player->x >= 4 && player->x <= 7 && player->y >= 2 && player->y <= 5)
          message.push_back("Move out of ze way comrade! ");
        else
          rotate_recenter();
        break;

      default:
        continue;
    }

    redraw();

    run_rats();
    if(rand()%1500 <= player->hp + target)
    {
      for(int i=0; i<20; i++)
      {
        int x = rand()%10 +1;
        int y = rand()%16 +5;

        if(current_map->data[y][x] == FLOOR && !is_crate_there(x,y) && (x !=
player->x && y != player->y))
        {
          monsters.push_back(new monster);
          monsters[monsters.size()-1]->hp = 3;

          monsters[monsters.size()-1]->x = x;
          monsters[monsters.size()-1]->y = y;

          break;
        }
      }
    }
  }
  message.push_back("You die... ");
  redraw();
  getch();

  endwin();

  return 0;
}

void victory()
{
  erase();

  mvprintw(0, 0, "Very good comrade!  You have completed your task!\n\nNow kindly
allow the guards to escort you back to your cell...");
  refresh();
```

// Listing continued on next page…

// Listing continued from previous page

```
  getch();
  endwin();

  exit(0);
}

void move_player(int direction)
{
  int x = player->x;
  int y = player->y;

  switch(direction)
  {
    case UP:
      y--;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
        {
          if(!push_crate(UP))
            return;
        }
        if(attack(x,y)) return;

        player->y = y;
        return;
      }
      break;
    case DOWN:
      y++;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
        {
          if(!push_crate(DOWN))
          return;
        }
        if(attack(x,y)) return;

        player->y = y;
        return;
      }
      break;
    case LEFT:
      x--;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
        {
          if(!push_crate(LEFT))
          return;
        }
        if(attack(x,y)) return;

        player->x = x;
        return;
      }
      break;
    case RIGHT:
      x++;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
```

// Listing continued on next page…

// Listing continued from previous page

```
        {
          if(!push_crate(RIGHT))
            return;
        }
        if(attack(x,y)) return;

        player->x = x;
        return;
      }
      break;
    case UPLEFT:
      x--;
      y--;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
        {
          message.push_back("You don't have enough leverage to push the block "
            "that way");
          return;
        }
        if(attack(x,y)) return;

        player->x = x;
        player->y = y;
        return;
      }
      break;

    case UPRIGHT:
      x++;
      y--;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
        {
          message.push_back("You don't have enough leverage to push the block "
            "that way");
          return;
        }
        if(attack(x,y)) return;

        player->x = x;
        player->y = y;
        return;
      }
      break;

    case DOWNLEFT:
      x--;
      y++;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
        {
          message.push_back("You don't have enough leverage to push the block "
            "that way");
          return;
        }
        if(attack(x,y)) return;

        player->x = x;
        player->y = y;
```

// Listing continued on next page…

// Listing continued from previous page

```
        return;
      }
      break;

    case DOWNRIGHT:
      x++;
      y++;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
        {
          message.push_back("You don't have enough leverage to push the block "
            "that way");
          return;
        }
        if(attack(x,y)) return;

        player->x = x;
        player->y = y;
        return;
      }
      break;
  }

  if(current_map->data[y][x] == CLOSED_DOOR)
    message.push_back("Ze door is locked comrade, complete your task! ");
  else if(current_map->data[y][x] == OPEN_DOOR)
  {
    level_count++;
    if(level_count == 5)
      victory();

    redraw();
    mvprintw(player->y+1, player->x+xoffset, ".");
    for(unsigned int i=0; i<monsters.size(); i++)
      mvprintw(monsters[i]->y+1, monsters[i]->x+xoffset, ".");
    message.push_back("You continue on... ");
    load_new_level();
  }

  return;
}

void run_rats()
{
  for(unsigned int i=0; i<monsters.size(); i++)
  {
    if(rand()%2 != 1)
      continue;

    int x = monsters[i]->x;
    int y = monsters[i]->y;
    int move;

    if(abs(monsters[i]->x-player->x) <= 1 && abs(monsters[i]->y-player->y) <= 1)
    {
      if(abs(monsters[i]->x-player->x)== 1 && abs(monsters[i]->y-player->y) == 1)
      {
        if(monsters[i]->x > player->x)
          move = LEFT;
        else
          move = RIGHT;
      }
```

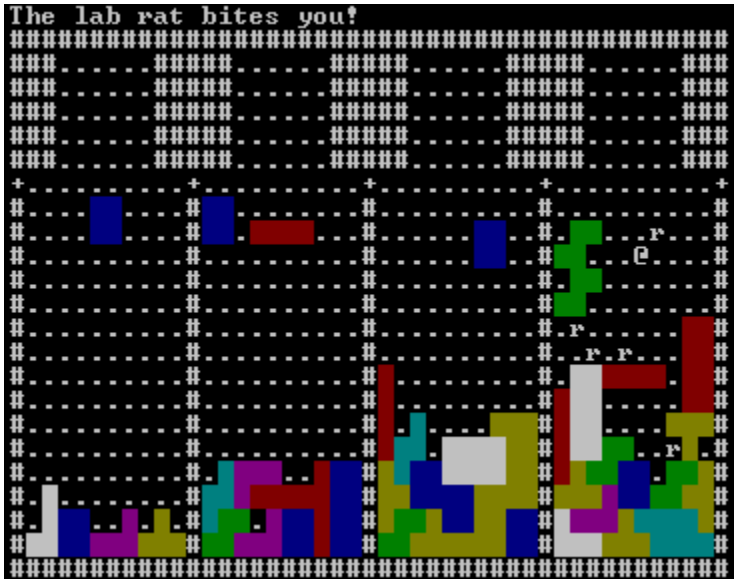
// Listing continued on next page…

// Listing continued from previous page

```
      else
      {
        message.push_back("The lab rat bites you! ");
        move = 9;

        player->hp -= 1;
      }
    }
    else if(abs(monsters[i]->x - player->x) > 4
      && abs(monsters[i]->y - player->y) > 4) // && rand()%4 != 1)
    {
      if(abs(monsters[i]->x - player->x) > abs(monsters[i]->y - player->y))
      {
        if(player->x > monsters[i]->x)
          move = RIGHT;
        else
          move = LEFT;
      }
      else
      {
        if(player->y > monsters[i]->y)
          move = DOWN;
        else
          move = UP;
      }
    }
    else
    {
      move = rand()%4;
    }
    switch(move)
    {
    case UP:
      y--;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
          break;

        monsters[i]->y--;
      }
      break;
    case DOWN:
      y++;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
          break;

        monsters[i]->y++;
      }
      break;
    case LEFT:
      x--;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
          break;

        monsters[i]->x--;
      }
      break;
    case RIGHT:
```

// Listing continued on next page…

// Listing continued from previous page




```
      x++;
      if(current_map->data[y][x] == FLOOR)
      {
        if(is_crate_there(x, y))
          break;

        monsters[i]->x++;
      }
      break;
    default:
      break;
    }

    if(monsters[i]->y <= 5)
      monsters[i]->y++;
  }
}

bool attack(int x, int y)
{
  for(unsigned int t=0; t<monsters.size(); t++)
    if(monsters[t]->x == x && monsters[t]->y == y)
    {
      monsters[t]->hp--;
      if(monsters[t]->hp <= 0)
      {
        message.push_back("You kill the lab rat with your wrench! ");

        delete monsters[t];
        monsters.erase(monsters.begin()+t);
      }
      else
        message.push_back("You hit the lab rat with your wrench ");
      return true;
    }
  return false;
}

bool is_crate_there(int x, int y)
{
  if(x < current_crate->x || y < current_crate->y || x > current_crate->x+4
    || y > current_crate->y+4)
    return false;

  for(int n=0; n<4; n++)
  for(int i=0; i<4; i++)
  {
    if(current_crate->shape[current_crate->rotation]->data[n][i])
      if(i+current_crate->x == x && n+current_crate->y == y)
        return true;
  }

  return false;
}

bool push_crate(int direction)
{
  int x = 0;
  int y = 0;

  switch(direction)
  {
    case UP:
```

// Listing continued on next page…

// Listing continued from previous page

```
      y--;
      break;
    case DOWN:
      y++;
      break;
    case RIGHT:
      x++;
      break;
    case LEFT:
      x--;
      break;
  }

  int offset_x = current_crate->x + x;
  int offset_y = current_crate->y + y;


  for(int n=0; n<4; n++)
  for(int i=0; i<4; i++)
  {
    if(current_crate->shape[current_crate->rotation]->data[n][i])
    {
      if(current_map->data[n+offset_y][i+offset_x] != FLOOR)
        return false;

      for(unsigned int t=0; t<monsters.size(); t++)
        if(monsters[t]->x == i+offset_x && monsters[t]->y == n+offset_y)
        {
          message.push_back("The lab rat gets in the way! ");
          return false;
        }
    }
  }

  current_crate->x = offset_x;
  current_crate->y = offset_y;

  return true;
}

void plant_piece()
{
  if(current_crate->y <= 5)
    return;

  for(int n=0; n<4; n++)
  for(int i=0; i<4; i++)
  {
    if(current_crate->shape[current_crate->rotation]->data[n][i])
      current_map->data[current_crate->y+n][current_crate->x+i] = current_crate-
>color + 3;
  }

  int lines = 0;

  for(unsigned int n=0; n<current_map->height;n++)
  {
    int count = 0;
    for(unsigned int i=0; i<current_map->width; i++)
      if(current_map->data[n][i] >= 4)
        count++;
    if(count == 10)
      lines++;
```

// Listing continued on next page…

// Listing continued from previous page

```
  }

  if(lines > filled_lines)
  {
    message.push_back("Excellent, keep up the good work comrade! ");
    player->hp += (lines - filled_lines)*5;

    filled_lines = lines;
  }

  player->hp += 2;
  placed++;

  if(placed == target)
  {
    current_map->data[6][11] = OPEN_DOOR;
    message.push_back("Vary good comrade! Proseed to ze exit. ");
  }
  else
    create_piece();

  return;
}

void create_piece()
{
  int num = rand() % 7;

  string peice;

  switch(num)
  {
    case 0: peice = "i"; break;
    case 1: peice = "j"; break;
    case 2: peice = "l"; break;
    case 3: peice = "o"; break;
    case 4: peice = "s"; break;
    case 5: peice = "t"; break;
    case 6: peice = "z"; break;
  }

  delete current_crate;
  current_crate = new crate(peice);
  current_crate->x = 4;
  current_crate->y = 2;

  return;
}

void rotate_recenter()
{
  current_crate->rotation++;
  if(current_crate->rotation == 4)
    current_crate->rotation = 0;

  current_crate->x = 4;
  current_crate->y = 2;

  return;
}

void load_new_level()
{
```

// Listing continued on next page…

// Listing continued from previous page

```
  ifstream fin("data/std.map");

  fin >> height >> width;

//  delete current_map;
//  delete current_crate;

  player->x = 1;
  player->y = 6;

  monsters.clear();

  current_map = new map(height, width, FLOOR);

  for(unsigned int n=0; n<height; n++)
    for(unsigned int i=0; i<width; i++)
      fin >> current_map->data[n][i];

  create_piece();

  placed = 0;
  target += 5;

  xoffset += 11;

  filled_lines = 0;

  return;
}

void redraw()
{
  void *dumb;

  for(unsigned int n=0; n<current_map->height; n++)
  for(unsigned int i=0; i<current_map->width; i++)
  {
    switch(current_map->data[n][i])
    {
      case FLOOR:
        mvprintw(n+1, i+xoffset, ".");
        break;
      case WALL:
        mvprintw(n+1, i+xoffset, "#");
        break;
      case CLOSED_DOOR:
        mvprintw(n+1, i+xoffset, "+");
        break;
      case OPEN_DOOR:
        mvprintw(n+1, i+xoffset, "-");
        break;
      case RED_BLOCK:
        mvprintw(n+1, i+xoffset, " ");
        mvchgat(n+1, i+xoffset, 1, COLOR_PAIR(1), 1, dumb);
        break;
      case WHITE_BLOCK:
        mvprintw(n+1, i+xoffset, " ");
        mvchgat(n+1, i+xoffset, 1, COLOR_PAIR(2), 2, dumb);
        break;
      case MAGEN_BLOCK:
        mvprintw(n+1, i+xoffset, " ");
        mvchgat(n+1, i+xoffset, 1, COLOR_PAIR(3), 3, dumb);
        break;
```

// Listing continued on next page…

// Listing continued from previous page

```
      case BLUE_BLOCK:
        mvprintw(n+1, i+xoffset, " ");
        mvchgat(n+1, i+xoffset, 1, COLOR_PAIR(4), 4, dumb);
        break;
      case GREEN_BLOCK:
        mvprintw(n+1, i+xoffset, " ");
        mvchgat(n+1, i+xoffset, 1, COLOR_PAIR(5), 5, dumb);
        break;
      case YELLOW_BLOCK:
        mvprintw(n+1, i+xoffset, " ");
        mvchgat(n+1, i+xoffset, 1, COLOR_PAIR(6), 6, dumb);
        break;
      case CYAN_BLOCK:
        mvprintw(n+1, i+xoffset, " ");
        mvchgat(n+1, i+xoffset, 1, COLOR_PAIR(7), 7, dumb);
        break;

      default:
        break;
    }
  }

  for(int n=0; n<4; n++)
  for(int i=0; i<4; i++)
  {
    if(current_crate->shape[current_crate->rotation]->data[n][i])
    {
      mvprintw(n+current_crate->y+1, i+current_crate->x+xoffset, " ");
      mvchgat(n+current_crate->y+1, i+current_crate->x+xoffset, 1,
        COLOR_PAIR(current_crate->color), current_crate->color, dumb);
    }
  }

  for(unsigned int i=0; i<monsters.size(); i++)
  {
    mvprintw(monsters[i]->y+1, monsters[i]->x+xoffset, "r");
  }

  mvprintw(10, 60, "        ");
  mvprintw(10, 60, "HP: %d", player->hp);
  mvprintw(11, 60, "                ");
  mvprintw(11, 60, "Placed: %d/%d", placed,target);

  mvprintw(player->y+1, player->x+xoffset, "@");

  for(int n=0; n<COLS; n++)
    mvprintw(0, n, " ");

  while(message.size() != 0)
  {
    string output = "";
    unsigned int width;
    int total = 0;
    if(COLS < 80)
      width = 80;
    else
      width = COLS;

    output += message[0];
    total += message[0].length();
    message.erase(message.begin());

    while(message.size() != 0)
```

// Listing continued on next page…

// Listing continued from previous page

```
    {
      if(total + message[0].length() > width - 10)
        break;

      output += message[0];
      total += message[0].length();
      message.erase(message.begin());
    }

    if(message.size() != 0)
      output += " - MORE -";

    for(unsigned int n=0; n<width; n++)
      mvprintw(0, n, " ");

    mvprintw(0, 0, (char* )output.c_str());

    refresh();

    if(message.size() != 0)
    {
        getch();
    }
  }

  touchwin(stdscr);
  refresh();

  return;
}
```

Additional data files:
The following files will need to be typed exactly as they appear saved with the following names and stored in a directory called ‘data’

i.tet

```
1
0 0 0 0
0 0 0 0
1 1 1 1
0 0 0 0

0 1 0 0
0 1 0 0
0 1 0 0
0 1 0 0

0 0 0 0
0 0 0 0
1 1 1 1
0 0 0 0

0 1 0 0
0 1 0 0
0 1 0 0
0 1 0 0
```

j.tet

```
2
0 0 0 0
1 1 1 0
0 0 1 0
0 0 0 0

0 1 1 0
0 1 0 0
0 1 0 0
0 0 0 0

1 0 0 0
1 1 1 0
0 0 0 0
0 0 0 0

0 1 0 0
0 1 0 0
1 1 0 0
0 0 0 0
```

l.tet

```
3
0 0 1 0
1 1 1 0
0 0 0 0
0 0 0 0

1 1 0 0
0 1 0 0
0 1 0 0
0 0 0 0

0 0 0 0
1 1 1 0
1 0 0 0
0 0 0 0

0 1 0 0
0 1 0 0
0 1 1 0
0 0 0 0
```

o.tet

```
4
0 0 0 0
0 1 1 0
0 1 1 0
0 0 0 0

0 0 0 0
0 1 1 0
0 1 1 0
0 0 0 0

0 0 0 0
0 1 1 0
0 1 1 0
0 0 0 0

0 0 0 0
0 1 1 0
0 1 1 0
0 0 0 0
```

s.tet

```
5
0 0 0 0
0 1 1 0
1 1 0 0
0 0 0 0

1 0 0 0
1 1 0 0
0 1 0 0
0 0 0 0

0 0 0 0
0 1 1 0
1 1 0 0
0 0 0 0

1 0 0 0
1 1 0 0
0 1 0 0
0 0 0 0
```

t.tet

```
6
0 0 0 0
1 1 1 0
0 1 0 0
0 0 0 0

0 1 0 0
0 1 1 0
0 1 0 0
0 0 0 0

0 1 0 0
1 1 1 0
0 0 0 0
0 0 0 0

0 1 0 0
1 1 0 0
0 1 0 0
0 0 0 0
```

z.tet

```
7
0 0 0 0
1 1 0 0
0 1 1 0
0 0 0 0

0 1 0 0
1 1 0 0
1 0 0 0
0 0 0 0

0 0 0 0
1 1 0 0
0 1 1 0
0 0 0 0

0 1 0 0
1 1 0 0
1 0 0 0
0 0 0 0
```

map.std

```
23 12
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 0 1 1 1
2 0 0 0 0 0 0 0 0 0 0 2
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1
```