Game project

By Harry Bentham

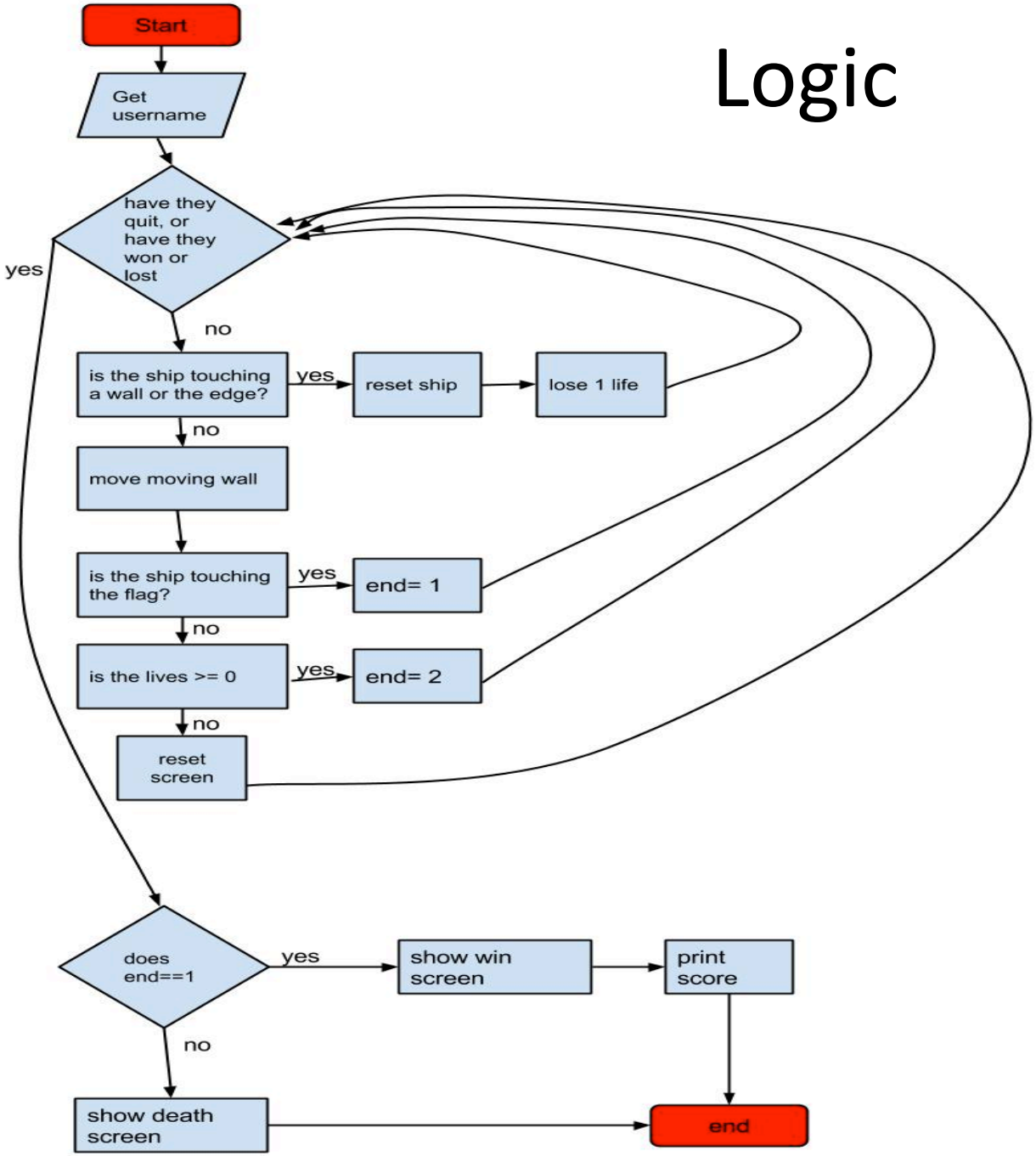
Analysis

- We decided we wanted the game to be:
  - A space themed game
  - Have a moving ship.
  - A maze starting in the top left corner and ending in the bottom right
  - Collisions with walls or screen edge that takes you back to the beginning of the maze and lose a life
  - The game ends when you reach the flag.
  - If the ship is guided successfully around the maze to the finishing point then a message of success should be displayed.
  - Your score is determined by an algorithm, which combines time taken to complete, number of lives remaining and the speed the game was running at.
  - Game speed can be selected (1 to 5) before you move the spaceship.
  - To make the game more interesting we decided to add moving walls (either horizontally or vertically)
  - The game should calculate a score on successful completion and store the score alongside the other high scores if the score is within the top five.

Analysis continued

- Additional features of the game include:
  - Five of our wall move up and down or left and right
  - There are 5 different difficulty settings (easy, medium, hard, very hard, über hard) which change the speed at which the walls move,
  - Each player starts with 5 lives, which reduce upon collision with walls or screen edges.
  - At the end of the game, splash screens appear to tell you whether you have won or lost.

Logic


Start
Get username
have they quit, or have they won or lost
yes
no
is the ship touching a wall or the edge?
yes
reset ship
lose 1 life
no
move moving wall
is the ship touching the flag?
yes
end= 1
no
is the lives >= 0
yes
end= 2
no
reset screen
does end==1
yes
show win screen
print score
no
show death screen
end

Pseudo Code and C code

Initialise variables and screen definition.
Display “game starting”
Ask user to enter their initials.

```
printf("Game starting...\n");
    printf("please enter your initials (max 3 characters)\n");
    scanf("%s",&playername);
```

Loop until user quits, user reaches the flag or user looses all 5 lives.

```
while(quit==FALSE && end==0){
```

If the movement buttons are pressed at any time add or subtract from the x and y co-ordinates of the character.

(if up is pressed subtract from y co-ordinate)

(if down is pressed add to y co-ordinate)

(if left is pressed subtract from x co-ordinate)

(if right is pressed add to x co-ordinate)

```
if(keystates[SDLK_UP])
    ship.y-=5;
if(keystates[SDLK_DOWN])
    ship.y+=5;
if(keystates[SDLK_LEFT])
    ship.x-=5;
if(keystates[SDLK_RIGHT])
    ship.x+=5;
if(keystates[SDLK_RETURN])
    end=1;
```

If the ship moves outside the screen reset its position and take a life away.

```
if(ship.x<0){
    ship.x=spawnx;
    ship.y=spawny;
    lives-=1;
    }
else if(ship.y+ship.h>SCREEN_HEIGHT){
    ship.x=spawnx;
    ship.y=spawny;
    lives-=1;
    }

if(ship.y<0){
    ship.y=spawny;
    ship.x=spawnx;
    lives-=1;
    }
else if(ship.x+ship.w>SCREEN_WIDTH){
    ship.y=spawny;
    ship.x=spawnx;
    lives-=1;
    }
```

Pseudo Code and C code continued

```
for(i=ship.x; i<ship.x+ship.w; i++){
    for(o=ship.y; o<ship.y+ship.h; o++){
        for(wallno=0; wallno<sizeof(wallx)/sizeof(int); wallno++){
            if(i>wallx[wallno] && i<wallx[wallno] + wallw[wallno] && o>wally[wallno] && o<wally[wallno] + wallh[wallno])
{
                ship.y=spawny;
                ship.x=spawnx;
                wallhit=1;
                }
                else{
                //do nothing;
                }
            }
        }
    }
if(wallhit==1){
    lives-=1;
    wallhit=0;
```

If the ship's area coincides with the perimeter of any of the walls, reset its position and take a life away.

Pseudo Code and C code continued

If the ship collides with the finish flag, user has won.

```
for(i=ship.x; i<ship.x+ ship.w; i++){
        for(o=ship.y; o<ship.y+ship.h; o++){
            if(i>flagbox.x && i<flagbox.x+flagbox.w && o>flagbox.y && o<flagbox.y + flagbox.h){
                end=1;
            }
        }
    }
```

Re-draw screen.

```
// Apply background to screen
        CCSS_apply_surface(0, 0, background, screen);

        for(wallno=1; wallno<sizeof(wally)/sizeof(int); wallno++){
        // apply walls
        CCSS_apply_surface(wallx[wallno], wally[wallno], wallpic, screen);
        }


        // Apply our character
        CCSS_apply_surface(ship.x, ship.y, character, screen);
        //Apply our flag
        CCSS_apply_surface(580, 420, flag, screen);
        // Built Wall
        //CCSS_print(400, 0, font, text_color, screen, "Position %d-%d", ship.x, ship.y);
        CCSS_print(400, 0, font, text_color, screen, "Lives = %d,",lives);
        // Update screen
        SDL_Flip( screen );
        ticks = SDL_GetTicks() - starttick;
        if(ticks < 1000 / FRAMES_PER_SECOND){
            SDL_Delay((1000/FRAMES_PER_SECOND)-ticks);
        }
    }
```

Pseudo Code and C code continued

If the ship collides with the finish flag, user has won.

```
for(i=ship.x; i<ship.x+ ship.w; i++){
        for(o=ship.y; o<ship.y+ship.h; o++){
            if(i>flagbox.x && i<flagbox.x+flagbox.w && o>flagbox.y && o<flagbox.y + flagbox.h){
                end=1;
            }
        }
    }
```

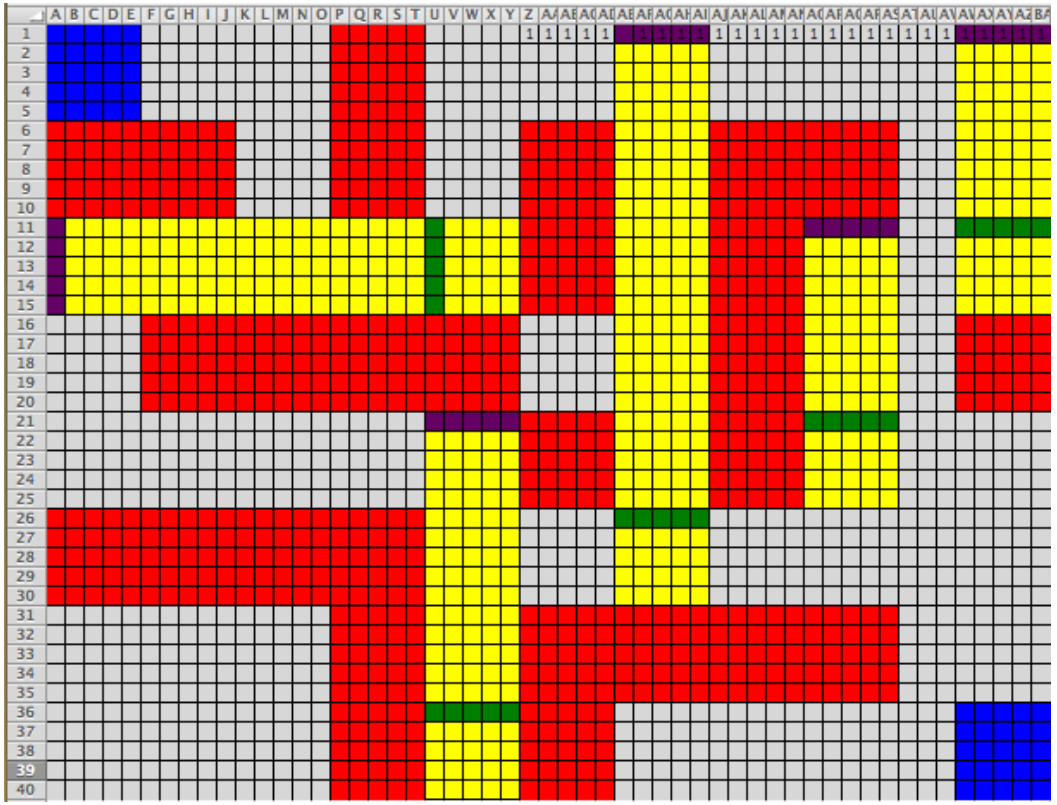
Re-draw screen.

```
// Apply background to screen
        CCSS_apply_surface(0, 0, background, screen);

        for(wallno=1; wallno<sizeof(wally)/sizeof(int); wallno++){
        // apply walls
        CCSS_apply_surface(wallx[wallno], wally[wallno], wallpic, screen);
        }


        // Apply our character
        CCSS_apply_surface(ship.x, ship.y, character, screen);
        //Apply our flag
        CCSS_apply_surface(580, 420, flag, screen);
        // Built Wall
        //CCSS_print(400, 0, font, text_color, screen, "Position %d-%d", ship.x, ship.y);
        CCSS_print(400, 0, font, text_color, screen, "Lives = %d,",lives);
        // Update screen
        SDL_Flip( screen );
        ticks = SDL_GetTicks() - starttick;
        if(ticks < 1000 / FRAMES_PER_SECOND){
            SDL_Delay((1000/FRAMES_PER_SECOND)-ticks);
        }
    }
```

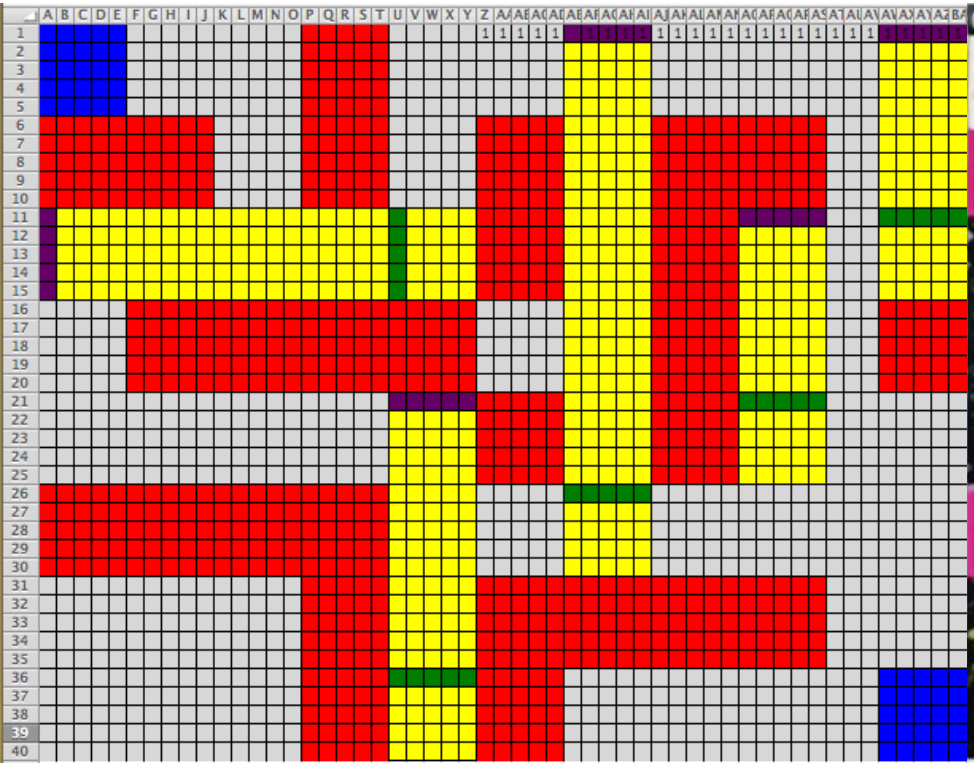
Level design

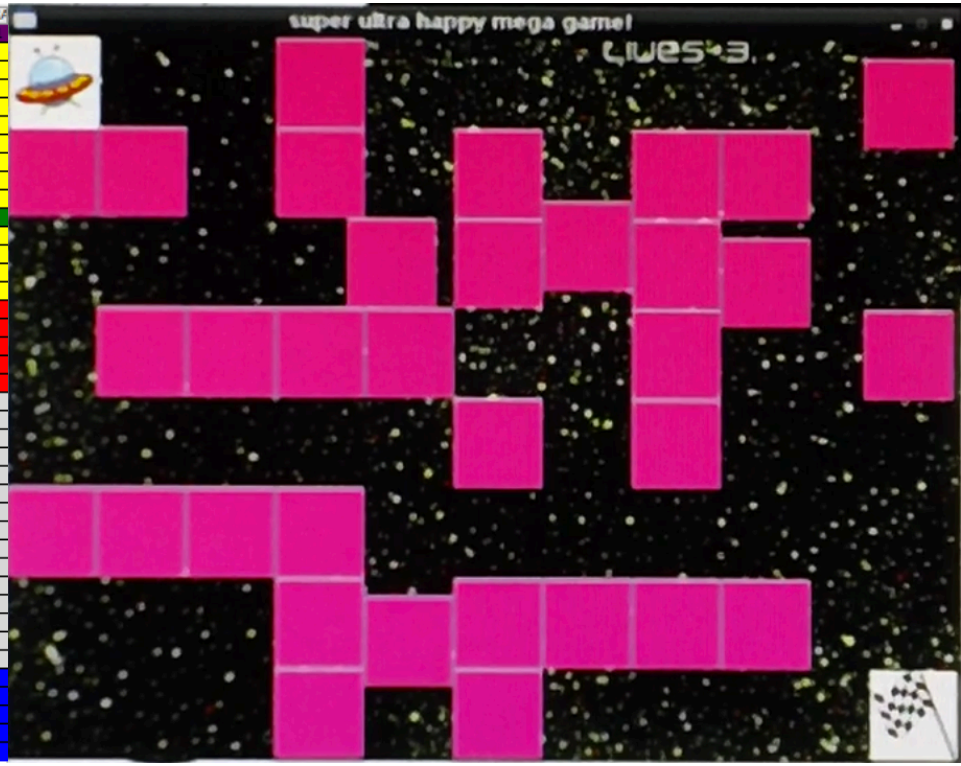
Key

- Red= static wall
- Yellow = 60 x 60 moving wall path
- Blue = start/ finish
- Grey = area where the spaceship can move
- Purple = moving wall starting point/ where wall changes direction
- Green = where wall changes direction



Finished Result





Development

Getting started and approach:

- First we got the template program
- Then compiled it to make sure that we had all the resources available and configured :
  - To compile and run the program you need to have installed:
    - All of the files from the github page
    - sudo apt-get install libsdl-dev
    - sudo apt-get install libsdl-image1.2-dev
    - sudo apt-get install libsdl-gfx1.2-dev
    - sudo apt-get install libsdl-ttf2.0-dev
  - Then compile the library by:
    - cd./lib/ccss
    - make Makefile all
    - cd ../../
  - Then compile the game
    - gcc game.c ./lib/ccss/ccss.a -o game.o -lSDL -lSDL_image -lSDL_gfx -lSDL_ttf
- We then read the code to understand the logic:
  - Initialisation of the screen and different variables.
  - Loop until exit pressed
  - Clean up
  - Return 0

Test and success criteria

- We tested at each stage of the development, adding features and checking that they worked. We also checked that existing features still worked.
- An example of this was when we had one wall we checked that collisions worked correctly from each side before extending to multiple walls.
- Conditions tested
  - Collisions with screen edges, success if ship returned to it's predetermined spawn point and lost a life.
  - Collisions with walls, both static and dynamic, success if ship returned to it's predetermined spawn point and lost a life.
  - On collision spaceship returns to the spawn point and the number of lives is reduces.
  - If you failed to navigate to the end flag and loose all 5 lives the Death screen is shown
  - If you succeeded to navigate to the end flag Success screen is shown
  - Selecting difficulty level 1 to 5 changes moving wall speed
  - All paths through the maze are possible, except for the impossible path.
  - Scores are calculated correctly.

Evidence of success

For evidence of successful test results see the video Computing report.mp4 in the file

Some tests did fail

We seemed to have trouble reading and writing from files. The high scores table unfortunately was unsuccessful after lots of tweaking; we kept getting a segmentation fault. We found that we were reading the high scores into the array, deleting them from the file and only writing the score that was just achieved.