[rows, columns] TOP LEFT (1,1)
Jonah
Lian
Thomas

Algorithms

main_module.c (IN: Void ; OUT: Void)

Declare global arrays (this needs to happen before the main fn w/ like the headers and shit)
3x 2D arrays (display_level, mine_level, timmy_level; 8 rows by 10 column grid)
********note : this is 0x7 and 0x9 start at 0 when passing to arrays.
/timmy_level : Glif TIM_LOC
3 possible values -
1) Timmy hasn't been on it (TIM_NOT_BEEN)
2) Timmy is on it (TIM_ON)
3) Timmy has been on it (TIM_BEEN)
As timmy moves around, if he goes into a cell value 1, change to value 2 If cell is safe and give a point. Once he leaves that cell, change value to 3. If he goes into a cell value 3, change to value 2 but do not give a point. Once he leaves that cell, change value to 3.
display_level: TIMMY [1], SAFE [2], EMPTY [3], FLAG [4], MINE [5], FL_MINE [6], EXPLODE [7]
mine_level: SAFE [2], EMPTY [3], FLAG [4], MINE [5], FL_MINE [6]

Declare global variable quit_flag and initialize it to false 0
Notes: quit flag can have values 0=FALSE, 1=TRUE, 3=RESTART
Declare timmys_location global array
Declare scoring variables
Flags_count
Mines_count
Score_count

while(quit_flag == FALSE || quit_flag == RESTART)
Welcome message, directions
Choose (//switch)
1) move
2) Plants flag
3) earn points
4) Start Game
5) Quit

If start selected
Choose (//switch)
1) Easy 6
2) Moderate 11
3) Hard 16
4) Impossible 20

[rows, columns] TOP LEFT (1,1)
Jonah
Lian
Thomas

Run configuration module

Gameplay module

Clear screen()

starting_configuration_module.c (IN: level; OUT: void)

Initialize all array coordinates to empty on all levels
Assign to safe spaces to certain cells on mine_level
Put timmy in upper left hand corner of timmy_level and initialize other tiles.
Upper left hand corner a safe cell
If user selected ...
Easy: set number of flags_count/mines_count to 6
Moderate: set number of flags_count/mines_count to 11
Hard: set number of flags_count/mines_count to 16
Impossible: set number of flags_count/mines_count to 20
Seed rand
For (i = number of mines; i > 0; i--)
Assignment loop
temploc_C=Rand column placement of mine
temploc_R=Rand row placement of mine
If current coordinate is empty fill mine array with a mine at given coordinate
Else rerun assignment loop
Replace all empty cells in the mine level with safe cells
On display level initialize tiles given other levels
draw_board function
Update: mines
flags
score
draw surface level glifs
Timmy
Safe spaces
Calculate adj for each safe space
And draw adj
Initialize score_count to 0;

gameplay_module.c (IN: void; OUT: void)
(remember to have a debug to check if this coordinate is the same as the timmy_level 2 value for every while loop pass)
While (quit_flag == FALSE)

[rows, columns] TOP LEFT (1,1)

Jonah

Lian

Thomas

```
user_input = getchar()
Switch (user_input) note: each case should be terminated with a break
        case 'y':  move(-1, -1)          //move up left
        case 'u':  move(-1, 0)           //move up
        case 'i':   move(-1, 1)          //move up right
        case 'h':  move(0, -1)           //move left
        case 'k':  move(0, 1)            //move right
        case 'n':  move(1, -1)           //move down left
        case 'm': move(1, 0)             //move down
        case ',':   move(1, 1)           //move down right
        case 'Y': plant_flag(-1, -1)     //plant up left
        case 'U': plant_flag(-1, 0)      //plant up
        case 'I':  plant_flag(-1, 1)     //plant up right
        case 'H': plant_flaf(0, -1)      //plant left
        case 'K': plant_flag(0, 1)       //plant right
        case 'N': plant_flag(1, -1)      //plant down left
        case 'M': plant_flag(1, 0)       //plant down
        case '<':  plant_flag(1, 1)      //plant down right
        case 'Q': (don't break here)
        case 'q': quit_sequence
        default: print error message (or just flush buffer)
```

Cannot place flag on SAFE, FL_MINE, FLAG, or Outside the grid

Check on display level for FLAG, SAFE, or FL_MINE

Check if grid number is within grid

If mine, give 2 points

If not, minus 1 point

Remove 1 flag from counter

Update flag counter display

move_module.c

For up_left, Input: a=-1, b=-1

```
Void move(int a, int b)
{
        int temp_row, temp_col, adjacent;
        temp_row = timmys_location[0] + a;
        temp_col = timmys_location[1] + b;
        if (check_out_of_grid(temp_row, temp_col) == OUT_OF_BOUNDS)
                write_message(15, You can't move there!);
        else if (check_out_of_grid(temp_row, temp_col) == WIN_ZONE)
                win();
        else if (check_out_of_grid(temp_row, temp_col) == IN_FIELD)
```

[rows, columns] TOP LEFT (1,1)
Jonah
Lian
Thomas

```
	{
	switch ( mine_level[temp_row, temp_col])
	{
		case Glif MINE:
		case Glif FL_MINE:
			death();
			Break;
		case Glif SAFE:
			update_score  +1;
			Move Timmy on display level and mine level to up left;
		case Glif FLAG:
			timmy_level[timmys_location[0], timmys_location[1]] = TIM_LOC
TIM_BEEN;
			timmy_level[temp_row, temp_col] = TIM_LOC TIM_ON;
			adjacent = adj(temp_row, temp_col);
			if (mine_level[timmys_location[0], timmys_location[1]] == FLAG)
				{
				show_glif(Glif FLAG, temp_row, temp_col, 0);
				display_level[timmys_location[0], timmys_location[1]] = FLAG;
				}
			if (mine_level[timmys_location[0], timmys_location[1]] == SAFE)
				{
				show_glif(Glif SAFE, temp_row, temp_col, adj);
				display_level[timmys_location[0], timmys_location[1]] = SAFE;
				}
			show_glif(Glif TIMMY, temp_row, temp_col, adj);
			display_level[temp_row, temp_col] = TIMMY;

			timmys_location[0] += a;
			timmys_location[1] += b;
			break;
		default:
			write_message("Error! The program fucked up\n");
			break;
	}
	}
}
```

plant_module.c
The following code is an example for the rest of the plant modules, which are:

plant_flag(a,b)

[rows, columns] TOP LEFT (1,1)
Jonah
Lian
Thomas

```
{
        int temp_row, temp_col;
        temp_row = timmys_location[0] + a;
        temp_col = timmys_location[1] + b;

        if (check_out_of_grid(temp_row, temp_col) == OUT_OF_BOUNDS)
                write_message(15, You can't move there!);
        else if (check_out_of_grid(temp_row, temp_col) == WIN_ZONE)
                win();
        else if (check_out_of_grid(temp_row, temp_col) == IN_FIELD)
        {
        switch (mine_level[temp_row + a, temp_column + b] )
        {
                case Glif SAFE:
                        if cell is empty in display level do following, else break;
                        show_glif(Glif FLAG, temp_row, temp_column, adj);
                        change mine level and display level cell type to flag
                        update_flags - 1;
                        update_score -1;
                        break;
                case Glif FLAG:
                        write_message("Cannot put a flag there you lose it.");
                        update_flags -1;
                        If FLAG<0 write_message("no more flags.");
                        break;
                case Glif MINE:
                        show_glif(Glif FL_MINE, temp_row, temp_column, adj);
                        change mine level and display level cell type to fl_mine
                        update_mines(-1);
                        update_flags -1;
                        update_score +2;
                        break;
                case Glif FL_MINE:
                        write_message("Cannot put a flag there you lose it.");
                        update_flags -1;
                        break;
                case 'q':
                        write_message("play again? \n Pressure too much, eh?\n");
                        death();
                        break;
                Default:
                        break;
        }
```

[rows, columns] TOP LEFT (1,1)
Jonah
Lian
Thomas

```
        }
}
```

check_adj_module.c

```
int adj(row, column)
{
        int count=0;
        if (Glif MINE == mine_level[row - 1, column - 1])
                count++;
        if (Glif MINE == mine_level[row - 1, column])
                count++;
        if (Glif MINE == mine_level[row - 1, column + 1])
                count++;
        if (Glif MINE == mine_level[row, column - 1])
                count++;
        if (Glif MINE == mine_level[row, column + 1])
                count++;
        if (Glif MINE == mine_level[row + 1, column - 1])
                count++;
        if (Glif MINE == mine_level[row + 1, column])
                count++;
        if (Glif MINE == mine_level[row + 1, column + 1])
                count++;

        return count;
}
```

endgame_module.c

```
void death(void)
        display timmy's death
        write_message ("play again?\n We're gonna need another Timmy!\n");
        quit _sequence();

void win(void)
        display timmy winning
        Ask about playing again
        quit _sequence();

void quit_sequence(void)
        While input is not correct scan
                If 'y', 'Y', 'yes', 'YES', or 'Yes'
                        quit_flag = RESTART;
                        clear screen
```

[rows, columns] TOP LEFT (1,1)
Jonah
Lian
Thomas

```
                    break;
            If ‘n’, ‘N’, ‘no’, ‘NO’, or ‘No’
                    quit_flag = TRUE;
                    clear screen
                    Break;
            Else
                    Display error message;
```

check_out_of_grid_module.c

This function is given a row and column and checks if the cell is within the playing area it returns

4 - OUT_OF_BOUNDS
5 - IN_FIELD
6 - WIN_ZONE

```
int check_out_of_grid(row, column)
{
        if (row < 0 || column < 0 || row > 7 || column > 9)
                return OUT_OF_BOUNDS;
        else if (column == 9)
                return WIN_ZONE;
        else
                return IN_FIELD;
}
```

messages.h

```
#include <string.h>
#include <stdio.h>

char error_move[25] = “
char error_flag[25] = “Cannot place a flag there\n you’ll lose it”;
Play_again[25] = “Play again? \nWe’re gonna need another Timmy!”;
```