```
// CS 355                        Date: 11 29 2022, Sunday

#include <ncurses.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>


#define DELAY 400//multiplied by val of at least 5.
#define WOOD_BASE 1
#define DIRT_BASE 2
#define FRUIT_BASE 3
//block author: Navin
void fillScrn(void);
int getDirect(char input, int direction);
bool collision(int x, int y, int direction, int check);
void moveSnake(int passedDir);
//block author: Navin and Logan
//global vars
int points; //score
int trophyX = 7;//trophy coordinates, to be passed
int trophyY = 3;
int trophyPts;
int trophyTimeStmp = 0;//will use as a timer
int wipeX = -2;//used to note if old fruit is to be removed
int wipeY = -2;//-1 does nothing, -2 adds new fruit, any other value wipes stored value location and adds new fruit
int sizeToIncrease = 0;
static int xCoord[2000]; //[0] is head, if lenghth needs to be > 2000 something is very wrong
static int yCoord[2000];
int length = 5;//can keep track of number of elements populated
bool trophyAlreadyHit = false;

int main(int argc, char *argv[])
{
    //block author: All, code here is a bit mixed for who did it
    srand(time(NULL));
    bool win = FALSE;
    //snake is technically a set max length
    //hold//int xCoord[LINES/2]; //[0] is head
    //hold//int yCoord[LINES/2];
    xCoord[0] = 7;
    xCoord[1] = 6;
    xCoord[2] = 5;
    xCoord[3] = 4;
    xCoord[4] = 3;
    //now y coords
    yCoord[0] = 7;
    yCoord[1] = 7;
    yCoord[2] = 7;
    yCoord[3] = 7;
    yCoord[4] = 7;
    //will only need to track head's direction,can represent with ints
    //0=left, 1=up. 2=right,3=down
        int direction = (rand() % 3)+1; //(max - min + 1) + min (3-1+1)+1
        int life=3;         //number of lives the game player have
    //screen setup
    initscr();
    noecho();
    curs_set(FALSE);
    timeout(1);
    //block author: Logan and Navin
    //begin prompt
    move(5,5);
    addstr("Press 's' to continue, or wait for game to start");//user prompt
    move(6,5);
    addstr("Use 'W', 'A', 'S', and 'D' to move");//user prompt
    move(7,5);
    addstr("Hit 'q' to quite");//user prompt
    move(0,0);//reset position
    for(int i = 0; i < 5000; i++)
    {
        usleep(5);
        if(getch() == 's')
        {
            i = 6000;//break loop
        }
    }
    //end beggining screen, proceed to color check and game
    //block author: Logan
    clear(); // Clear the screen of all
    // previously-printed characters
    if(has_colors() == FALSE)//prevents unlikely errors, porbably never going to get triggered
    {
        endwin();
        printf("No color");
        exit(1);
```

```
}
else
{
    start_color();
    init_pair(DIRT_BASE, COLOR_WHITE, COLOR_GREEN);//declare color pairs for later
    init_pair(WOOD_BASE, COLOR_BLACK, COLOR_RED);
    init_pair(FRUIT_BASE, COLOR_RED, COLOR_BLACK);

    clear();//wipe screen
    fillScrn();//make border, it doesn't render each time, only at begining
    int currDelay = 0;
    while(life > 0)
    {
        //get new direction
        char input = getch();
        if(input == 'q')//player wants to quit, but life variable should stay local so it is checked here
        {
            life = 0;
        }
        direction = getDirect(input, direction);
        //------------
        //block author: Logan
        if(collision(xCoord[0], yCoord[0], direction, currDelay))//freeze snake, update values, and proceed to next life
        {
            sleep(4);//flashing was requiring too many additional system resources, so just going to freeze-frame the snake
            attron(COLOR_PAIR(DIRT_BASE));
            for(int i = 0; i < length; i++)//safe to wipe all now that we are pre checking, may start at 0
            {
                mvaddch(yCoord[i], xCoord[i], ' ');
            }
            attroff(COLOR_PAIR(DIRT_BASE));
            refresh();
            life--;//update life counter
            if(life > 0)//can still continue with fresh spawn
            {
                sizeToIncrease = 0;
                length = 5;//reset snake
                xCoord[0] = 7;
                xCoord[1] = 6;
                xCoord[2] = 5;
                xCoord[3] = 4;
                xCoord[4] = 3;
                //now y coords
                yCoord[0] = 7;
                yCoord[1] = 7;
                yCoord[2] = 7;
                yCoord[3] = 7;
                yCoord[4] = 7;
                direction = (rand() % 3)+1; //(max - min + 1) + min (3-1+1)+1 RESETS DIRECTION
            }
        }
        //block author: Logan
        else if(length > LINES + COLS)//win condition, will work like bowling, get an additional life to run and get more points
        {
            life++;
            //reset snake
            sizeToIncrease = 0;
            length = 5;//reset snake
            xCoord[0] = 7;
            xCoord[1] = 6;
            xCoord[2] = 5;
            xCoord[3] = 4;
            xCoord[4] = 3;
            //now y coords
            yCoord[0] = 7;
            yCoord[1] = 7;
            yCoord[2] = 7;
            yCoord[3] = 7;
            yCoord[4] = 7;
            direction = (rand() % 3)+1; //(max - min + 1) + min (3-1+1)+1 RESETS DIRECTION
        }
        //block author: Logan
        else
        {
            //---------
            if(currDelay >= (DELAY/length))//delay clamp here
            {
                currDelay = 0;
                //wipe snake from screen
                for(int i = 0; i < length; i++)
                {
                    attron(COLOR_PAIR(DIRT_BASE));
                    mvhline(yCoord[i], xCoord[i], ' ', 1);//fills in dirt
                    attroff(COLOR_PAIR(DIRT_BASE));
                }
                //now move the snake
                moveSnake(direction);
```

```
                    //now to draw new snake
                    for(int i = 0; i < length; i++)
                    {
                        mvaddch(yCoord[i], xCoord[i], 'S');
                    }
                    //draw fruit
                    if(wipeX == -1 && wipeY == -1)//nothing to do
                    {
                        //continue as normal
                    }
                    else if(wipeX == -2 && wipeY == -2)//need to add new fruit, snake wiped old
                    {
                        attron(COLOR_PAIR(FRUIT_BASE));
                        mvaddch(trophyY, trophyX, (char)(trophyPts+48));
                        attroff(COLOR_PAIR(FRUIT_BASE));
                        wipeX = -1; //set flags
                        wipeY = -1;
                        trophyAlreadyHit = false;
                    }
                    else//wipe old fruit, then add new NEED TO CHECK THIS MORE OFTEN
                    {
                        //wipe old
                        attron(COLOR_PAIR(DIRT_BASE));
                        mvaddch(wipeY, wipeX, ' ');
                        attroff(COLOR_PAIR(DIRT_BASE));
                        //new fruit
                        attron(COLOR_PAIR(FRUIT_BASE));
                        mvaddch(trophyY, trophyX, (char)(trophyPts+48));
                        attroff(COLOR_PAIR(FRUIT_BASE));
                        wipeX = -1; //set flags
                        wipeY = -1;
                        trophyAlreadyHit = false;
                    }

                    //---------
                    refresh();
                }
                trophyTimeStmp--;
                usleep(1); // Shorter delay between movements
                currDelay++;
            }
        }
    }
    //end screen
    clear();
    move(10,10);
    printw("Your combined score is: %d", points);//like printf, but for ncurses
    refresh();
    sleep(10);

    endwin();
    printf("Your combined score is: %d\n", points);//send score to console as well, in case player wants to see it later
}
//block author: Logan
int getDirect(char input, int direction)//converts keystroke to numerical direction representation
{
    if(input == 'w')//up
    {
        return 1;
    }
    else if(input == 'd')//right
    {
        return 2;
    }
    else if(input == 's')//down
    {
        return 3;
    }
    else if(input == 'a')//left
    {
        return 0;
    }
    return direction;//no change
}
//block author: Logan
void fillScrn()//initial environment display
{
    //clear brushes
    attroff(COLOR_PAIR(DIRT_BASE));
    attroff(COLOR_PAIR(WOOD_BASE));
    //do base color
    attron(COLOR_PAIR(DIRT_BASE));
    for (int y = 0; y < LINES; y++)
    {
        mvhline(y, 0, ' ', COLS);
    }
    attroff(COLOR_PAIR(DIRT_BASE));
```

```
    //Now Horozontal lines
    attron(COLOR_PAIR(WOOD_BASE));
    mvhline(0, 0, '-', COLS);
    mvhline(LINES-1, 0, '-', COLS);
    //Now Vertical lines
    mvvline(0, 0, '|', LINES);
    mvvline(0, COLS-1, '|', LINES);
    attroff(COLOR_PAIR(WOOD_BASE));
}
//block author: Logan
//returns 0 for none, 1 for end game collision
bool collision(int x, int y, int direction, int check)
{
    srand(time(NULL));
    //checks what will be the new location
    if(direction == 0)//left
    {
        x = x -1;
    }
    else if(direction == 1)//up
    {
        y = y - 1;
    }
    else if(direction == 2)//right
    {
        x = x + 1;
    }
    else if(direction == 3)//down
    {
        y = y + 1;
    }
    //block author: Vincent and Logan
    int chAsInt = mvinch(y, x);//outputs an int. Going to call a few times, best to save it
    if(((chAsInt & A_CHARTEXT) == '-') || ((chAsInt & A_CHARTEXT) == '|') || ((chAsInt & A_CHARTEXT) == 'S'))//checks for 2 border chars, or
snake
    {
        return TRUE;//has hit self or border
    }
    else if(trophyTimeStmp <= 0)
    {
        wipeX = trophyX;//no longer flag values
        wipeY = trophyY;
        //make new trophy
        trophyPts = (rand() % 9) + 1; //calculate new trophy's val (9 - 1 + 1) + 1
        trophyX = rand() % (COLS-2) + 1; //get new coords, random. Formula is really (COLS - 1 + 0 + 1) + 0, simplified. (max - min + 1) +
min, where max = COLS - 2
        trophyY = rand() % (LINES-2) + 1;//need -2, as it goes 0 -> LINES-1
        //ensure snake is not being hit by trophy
        while((mvinch(trophyY, trophyX) == 'S'))
        {
            trophyX = rand() % (COLS-2) + 1; //get new coords, random. Formula is really (COLS -2 - 1 + 0 + 1) + 0, simplified. (max - min +
1) + min, where max = COLS - 2
            trophyY = rand() % (LINES-2) + 1;
        }
        trophyTimeStmp = 1000 * (rand() % (9) + 1); //(max - min + 1) + min (9-1+1)+1, converted to seconds as it will need fewer calculations
(maybe)
    }
    else if((((chAsInt & A_CHARTEXT) == '1' ||(chAsInt & A_CHARTEXT) == '2' ||(chAsInt & A_CHARTEXT) == '3' ||(chAsInt & A_CHARTEXT) == '4' ||
(chAsInt & A_CHARTEXT) == '5' ||(chAsInt & A_CHARTEXT) == '6' ||(chAsInt & A_CHARTEXT) == '7' ||(chAsInt & A_CHARTEXT) == '8' ||(chAsInt &
A_CHARTEXT) == '9' ||(chAsInt & A_CHARTEXT) == 'o')) && !trophyAlreadyHit)
    {
        //snake hit fruit
        wipeX = -2;
        wipeY = -2;
        //fruit here
        points = points + trophyPts;//trophy reached, update score //can display value of fruit, but will need to update collision.
        sizeToIncrease = sizeToIncrease + trophyPts;//SIZE INCREASE IS GETTING TOO HIGH
        trophyAlreadyHit = true;
        //make new trophy
        trophyPts = (rand() % 9) + 1; //calculate new trophy's val (9 - 1 + 1) + 1
        trophyX = rand() % (COLS-2) + 1; //get new coords, random. Formula is really (COLS - 1 + 0 + 1) + 0, simplified. (max - min + 1) +
min, where max = COLS - 2
        trophyY = rand() % (LINES-2) + 1;//need -2, as it goes 0 -> LINES-1
        //ensure snake is not being hit by trophy
        while((mvinch(trophyY, trophyX) == 'S'))
        {
            trophyX = rand() % (COLS-2) + 1; //get new coords, random. Formula is really (COLS -2 - 1 + 0 + 1) + 0, simplified. (max - min +
1) + min, where max = COLS - 2
            trophyY = rand() % (LINES-2) + 1;
        }
        trophyTimeStmp = 1000 * (rand() % (9) + 1); //(max - min + 1) + min (9-1+1)+1, converted to seconds as it will need fewer calculations
(maybe)
    }
    return FALSE;//clear, or fruit
}
//block author: Logan
void moveSnake(int passedDir)//moves snake, self explanatory
```

```
{
    if(sizeToIncrease > 0)//have size to add, shift array SEG FAULT IN HERE
    {
        length++;
        sizeToIncrease = sizeToIncrease-1;
    }
    for(int i = length-1; i >= 0; i--)
    {
        if(i == 0)//head
        {
            if(passedDir == 0)//left
            {
                xCoord[i] = xCoord[i]-1;
            }
            else if(passedDir == 1)//up
            {
                yCoord[i] = yCoord[i]-1;
            }
            else if(passedDir == 2)//right
            {
                xCoord[i] = xCoord[i]+1;
            }
            else if(passedDir == 3)//down
            {
                yCoord[i] = yCoord[i]+1;
            }
        }
        else
        {
            xCoord[i] = xCoord[i-1];
            yCoord[i] = yCoord[i-1];
        }
    }
}
```