מחשוב זמן אמת

פרוייקט אופציה 1

מגישים:  
עמית בניטה, 315949594.

שניר יהודה, 308450154.

*#include* <dos.h>

*#include* <stdio.h>

*#include* <string.h>

*#include* <stdlib.h> *// Added for exit() function*

*#define* ARRSIZE 1000

*#define* BALL\_NUMBER 5

*#define* TARGET\_SIZE 5

*#define* TARGET\_NUMBER 12

*#define* BALL\_MAX\_X 78

*#define* BALL\_MAX\_Y 23

*#define* BALL\_MIN\_X 1

*#define* BALL\_MIN\_Y 1

void interrupt (\*old\_int9)(void);

void interrupt (\*old\_int8)(void);

*// Function prototypes*

void interrupt new\_int9(void);

void interrupt new\_int8(void);

void my\_halt(void);

void init\_interrupts(void);

void restore\_interrupts(void);

void displayer(void);

void receiver(void);

void updater(void);

void update\_target(int i);

void update\_ball(void);

void terminate\_application(void);

char entered\_ascii\_codes[ARRSIZE];

int tail = -1;

char display[2001];

char ch\_arr[ARRSIZE];

int front = -1;

int rear = -1;

int ball\_speed\_x = 1;

int ball\_speed\_y = 1;

int cycle\_time = 25; *// Initial cycle time in ticks (100 ticks = 1 second)*

*typedef* *struct* position {

    int x;

    int y;

} POSITION;

POSITION target\_pos[TARGET\_NUMBER];

POSITION ball\_pos[BALL\_NUMBER];

POSITION ball\_angle[BALL\_NUMBER];

int game\_over;

int block\_position;

int block\_size;

int block\_in\_motion;

int initial\_run = 1;

int tick\_count = 0;

char display\_draft[25][80]; *// Define display\_draft*

int no\_of\_targets; *// Define no\_of\_targets globally*

int current\_stage = 1; *// Define current\_stage and initialize*

void my\_halt() {

    restore\_interrupts(); *// Ensure interrupts are restored before exiting*

*asm* {CLI}

    exit(0);

}

void interrupt new\_int9(void) {

    unsigned char scan; *// Use unsigned char for scan code*

    char ascii = 0;

*static* int ctrl\_pressed = 0; *// To track if Ctrl is pressed*

*asm* {

        IN AL, *0x*60 *// Read scan code from port 0x60*

        MOV BYTE PTR scan, AL

        IN AL, *0x*61 *// Read and acknowledge interrupt*

        OR AL, *0x*80

        OUT *0x*61, AL

        AND AL, *0x*7F

        OUT *0x*61, AL

    }

*// Detect Ctrl key press*

*if* (scan == *0x*1D) { *// Ctrl key down*

        ctrl\_pressed = 1;

    } *else* *if* (scan == *0x*9D || scan == *0x*1D) { *// Ctrl key up*

        ctrl\_pressed = 0;

    }

*// Detect other keys*

*if* (ctrl\_pressed && scan == *0x*2E) { *// C key while Ctrl is pressed*

        my\_halt(); *// Terminate program*

    } *else* *if* (scan == *0x*4B) { *// Left arrow*

        ascii = 'a';

    } *else* *if* (scan == *0x*48) { *// Up arrow*

        ascii = 'w';

    } *else* *if* (scan == *0x*4D) { *// Right arrow*

        ascii = 'd';

    } *else* *if* (scan == *0x*50) { *// Down arrow*

        ascii = 's';

    } *else* *if* (scan == *0x*01) { *// Esc*

        my\_halt(); *// Terminate program*

    }

*// Ensure we do not exceed the bounds of entered\_ascii\_codes*

*if* (ascii != 0 && tail < ARRSIZE - 1) {

        entered\_ascii\_codes[++tail] = ascii;

    }

    (\*old\_int9)(); *// Call the old interrupt handler*

}

void interrupt new\_int8(void) {

    tick\_count++;

*if* (tick\_count >= cycle\_time) { *// Use cycle\_time to control the tick frequency*

        receiver();

        updater();

        displayer();

        tick\_count = 0;

    }

    (\*old\_int8)(); *// Call the old interrupt handler*

}

void init\_interrupts() {

    old\_int8 = getvect(8);

    setvect(8, new\_int8);

    old\_int9 = getvect(9);

    setvect(9, new\_int9);

}

void restore\_interrupts() {

    setvect(8, old\_int8);

    setvect(9, old\_int9);

}

void displayer() {

    char far\* screen = (char far\*)*0x*B8000000;

    int i, j;

    char stage\_message[20];

    int len;

    int start;

*// Clear screen*

*for* (i = 0; i < 25; i++) {

*for* (j = 0; j < 80; j++) {

            screen[(i \* 80 + j) \* 2] = display[i \* 80 + j];

*if* (i == 24 && display[i \* 80 + j] == '=') {

                screen[(i \* 80 + j) \* 2 + 1] = *0x*1A; *// Blue background, light green text*

            } *else* *if* (display[i \* 80 + j] == '\*') {

                screen[(i \* 80 + j) \* 2 + 1] = *0x*1F; *// Blue background, white text*

            } *else* *if* (display[i \* 80 + j] == '#') {

                screen[(i \* 80 + j) \* 2 + 1] = *0x*27; *// Black background, green text*

            } *else* *if* (i == 0) { *// Upper walls*

                screen[(i \* 80 + j) \* 2 + 1] = *0x*4F; *// Red background, white text*

            } *else* {

                screen[(i \* 80 + j) \* 2 + 1] = *0x*70; *// Light gray background, black text*

            }

        }

    }

*// Display current stage number*

    sprintf(stage\_message, "Stage: %d", current\_stage);

    len = strlen(stage\_message);

    start = 0; *// Center the message on the screen*

*for* (i = 0; i < len; i++) {

        screen[(22 \* 80 + start + i) \* 2] = stage\_message[i];

        screen[(22 \* 80 + start + i) \* 2 + 1] = *0x*0F; *// White text on black background*

    }

}

void receiver() {

    char temp;

*while* (tail > -1) {

        temp = entered\_ascii\_codes[tail];

        rear++;

        tail--;

*if* (rear < ARRSIZE)

            ch\_arr[rear] = temp;

*if* (front == -1)

            front = 0;

    }

}

int ball\_hit\_right\_wall() {

*return* (ball\_pos[0].x == BALL\_MAX\_X && ball\_angle[0].x > 0);

}

int ball\_hit\_left\_wall() {

*return* (ball\_pos[0].x == BALL\_MIN\_X && ball\_angle[0].x < 0);

}

int ball\_hit\_block() {

*return* (ball\_pos[0].y >= BALL\_MAX\_Y && ball\_pos[0].x >= block\_position &&

            ball\_pos[0].x <= block\_position + block\_size);

}

int fall\_off() {

*return* (ball\_pos[0].y >= BALL\_MAX\_Y && ball\_angle[0].y > 0 && ball\_hit\_block() == 0);

}

int ball\_hit\_roof() {

*return* (ball\_pos[0].y == BALL\_MIN\_Y && ball\_angle[0].y < 0);

}

void reverse\_angle\_roof() {

    ball\_angle[0].y = -ball\_angle[0].y;

}

void reverse\_angle\_bottom() {

    ball\_angle[0].y = -ball\_angle[0].y;

}

void reverse\_angle\_left() {

    ball\_angle[0].x = -ball\_angle[0].x;

}

void reverse\_angle\_right() {

    ball\_angle[0].x = -ball\_angle[0].x;

}

void terminate\_application() {

    int i, j;

    char\* message = "GAME OVER";

    int len = strlen(message);

    int start = (80 - len) / 2;

*for* (i = 0; i < 25; i++) {

*for* (j = 0; j < 80; j++) {

            display[i \* 80 + j] = ' ';

        }

    }

*for* (i = 0; i < len; i++) {

        display[12 \* 80 + start + i] = message[i];

    }

    displayer();

    delay(3000); *// Display the game over message for 3 seconds*

    printf("Game over\n");

    restore\_interrupts(); *// Ensure interrupts are restored before exiting*

*asm* INT 27;

}

void update\_target(int i) {

*if* ((ball\_pos[0].y == BALL\_MIN\_Y) &&

        (ball\_pos[0].x >= target\_pos[i].x) &&

        (ball\_pos[0].x <= target\_pos[i].x + TARGET\_SIZE)) {

        target\_pos[i].x = -1; *// Mark target as hit*

    }

}

void update\_ball() {

*if* (fall\_off()) {

        terminate\_application();

    }

*if* (ball\_hit\_right\_wall() || ball\_hit\_left\_wall()) {

        reverse\_angle\_left();

    }

*if* (ball\_hit\_block()) {

        reverse\_angle\_bottom();

    }

*if* (ball\_hit\_roof()) {

        reverse\_angle\_roof();

    }

    ball\_pos[0].x += ball\_angle[0].x \* ball\_speed\_x;

    ball\_pos[0].y += ball\_angle[0].y \* ball\_speed\_y;

*if* (ball\_pos[0].x < BALL\_MIN\_X) ball\_pos[0].x = BALL\_MIN\_X;

*if* (ball\_pos[0].x > BALL\_MAX\_X) ball\_pos[0].x = BALL\_MAX\_X;

*if* (ball\_pos[0].y < BALL\_MIN\_Y) ball\_pos[0].y = BALL\_MIN\_Y;

*if* (ball\_pos[0].y > BALL\_MAX\_Y) ball\_pos[0].y = BALL\_MAX\_Y;

}

void updater() {

    int i, j;

    int no\_of\_balls = 1; *// Ensure no\_of\_balls is properly initialized*

    int target\_disp;

    int all\_targets\_hit;

    target\_disp = 80 / TARGET\_NUMBER;

    all\_targets\_hit = 1;

*if* (initial\_run == 1) {

        no\_of\_targets = 12;

*for* (i = 0; i < no\_of\_targets; i++) {

            target\_pos[i].x = (i \* target\_disp) + (target\_disp / 2);

            target\_pos[i].y = 1;

        }

        ball\_pos[0].x = BALL\_MAX\_X / 2;

        ball\_pos[0].y = BALL\_MAX\_Y - 1;

        ball\_angle[0].x = 2;

        ball\_angle[0].y = -2;

        block\_size = 10;

        block\_position = (BALL\_MAX\_X - block\_size) / 2;

        initial\_run = 0;

    }

*if* (block\_in\_motion == 0) {

        char temp;

*while* (front <= rear && front != -1) {

            temp = ch\_arr[front++];

*if* (front > rear)

                front = rear = -1;

*switch* (temp) {

*case* 'a':

*if* (block\_position > BALL\_MIN\_X)

                    block\_position--;

*break*;

*case* 'd':

*if* (block\_position + block\_size < BALL\_MAX\_X)

                    block\_position++;

*break*;

*default*:

*break*;

            }

        }

    }

*for* (i = 0; i < no\_of\_balls; i++)

        update\_ball();

*for* (i = 0; i < no\_of\_targets; i++) {

*if* (target\_pos[i].x != -1) {

            all\_targets\_hit = 0;

*break*;

        }

    }

*if* (all\_targets\_hit) {

        current\_stage++; *// Move to the next stage*

        no\_of\_targets = 12;

*for* (i = 0; i < no\_of\_targets; i++) {

            target\_pos[i].x = (i \* target\_disp) + (target\_disp / 2);

            target\_pos[i].y = 1;

        }

        ball\_pos[0].x = BALL\_MAX\_X / 2;

        ball\_pos[0].y = BALL\_MAX\_Y - 1;

        ball\_angle[0].x = 1;

        ball\_angle[0].y = -1;

        block\_size = 10;

        block\_position = (BALL\_MAX\_X - block\_size) / 2;

*// Decrease cycle time and increase ball speed for the next stage*

*if* (cycle\_time > 10) { *// Ensure the minimum cycle time is not too low*

            cycle\_time -= 3; *// Decrease cycle time gradually*

        }

        ball\_speed\_x += 1;

        ball\_speed\_y += 1;

    }

*// Update targets and display*

*for* (i = 0; i < no\_of\_targets; i++)

        update\_target(i);

*for* (i = 0; i < 25; i++) {

*for* (j = 0; j < 80; j++) {

            display\_draft[i][j] = ' ';

        }

    }

*for* (i = 0; i < no\_of\_targets; i++) {

*if* (target\_pos[i].x != -1) {

*for* (j = 0; j < TARGET\_SIZE; j++) {

                display\_draft[target\_pos[i].y][target\_pos[i].x + j] = '\*';

            }

        }

    }

*for* (i = 0; i < no\_of\_balls; i++) {

        display\_draft[ball\_pos[i].y][ball\_pos[i].x] = '#';

    }

*for* (i = 0; i < block\_size; i++) {

        display\_draft[24][block\_position + i] = '=';

    }

*for* (i = 0; i < 25; i++) {

*for* (j = 0; j < 80; j++) {

            display[i \* 80 + j] = display\_draft[i][j];

        }

    }

}

int main() {

*asm* {

        MOV AH, 3

        INT 10h

    }

    init\_interrupts();

*while* (!game\_over) {

*// Main game loop*

*// The game will continue running until game\_over becomes true*

*// This is a placeholder for the actual game logic*

        delay(10); *// Slow down the loop slightly*

    }

    restore\_interrupts();

*return* 0;

}