**Ducky Run**

Team Member 1: Manreet Kaur and W00410937

Team Member 2: Edmund Agyekum and W00420918

Team Member 3: Joe Andro W00403186

1. **Brief Description of the project and your game**

* The purpose of this project was to create a game using c-code, animation, and DE-10 boards to compile the code with to display on the monitor. The objective of our game is like the Pac-Man where the character runs around away from monster and must eat up all the dots. In our game there is a Duck which must go around and eat up all the boxes. Each box has a set number of points and the score increases based on the color box that the duck eats. Once the duck eats all the boxes, then the game is over, and the user can play again. In order to play our game, the user has to play using the ‘a’, ‘s’, ‘d’, and ‘w’, keys on the keyboard to move the duck and hit each box on the screen. As the duck hits the screen the score adds up and is displayed on the screen as well. The max score the user can get is 420. Once the score reaches 420 the game comes to an end. Interfaces used are the keyboard and monitor to input controls into the game and view the game respectively.

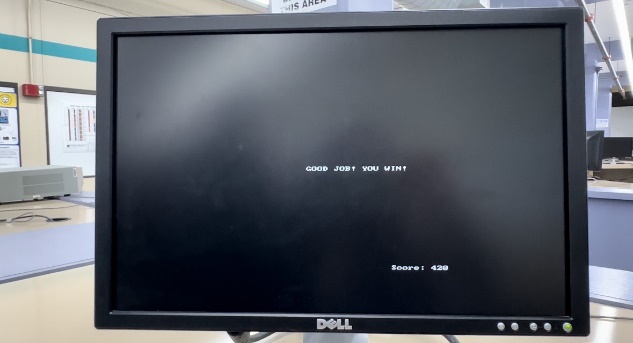
1. **Steps in creating the game:**

* Key steps in creating the game:
  + - 1. Design the Duck figure and the main screens for the game.
      2. Start with the Duck figure.
      3. Get the overall map with the boxes used in the game.
      4. Duck moves up, down, right, and left.
      5. Add the score displayed on the screen.
      6. Add a game over screen.

1. **Task Distribution among team members**
2. Member 1 task to design and write code for the duck figure. As well as designing the map for the game and writing the code for the main screen for the game with all the coordinates for each figure. In addition to completing the presentation and report documentation.
3. Member 2 task was to design a movement system to move the duck across the screen. And when the duck reached the box, the box would disappear, and score will be added as well as creating and implementing the “you win” and “game over” screen upon certain conditions being met in game.
4. Member 3 task assisted with the initial setup to the code.
5. **Software Flowchart**

* Create the initial duck.
* Repeating while loop:
  + Create the text value for the total score.
  + If the score is less that the highest achievable score:
    - Move the duck.
    - Create the map based on the duck's location.
    - Grab any boxes that are in the duck’s location.
    - Show related score on the screen.
  + If the score is equal to the highest achievable score (all boxes have been grabbed):
    - Updated the onscreen score.
    - Create a “you win” screen.
  + If you leave the bounds of the game:
    - Create a “you lose screen.”

1. **Screenshot Verification Screenshots**:

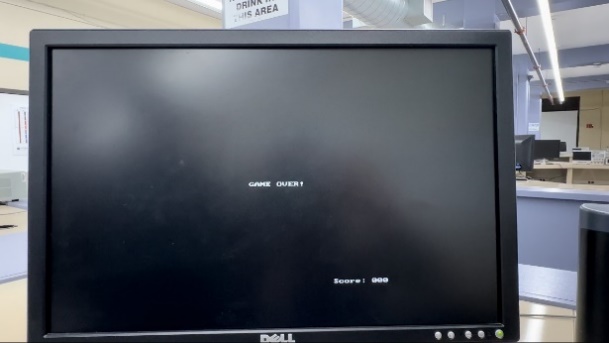


Once all the boxes are collected and the score reaches 420, this screen is shown, which says “Good Job! You win!”.

A computer screen with colorful squares on it

Description automatically generated

The main screen with the different boxes which the duck will get. Each box represents a different number of points.



If the player goes off screen (0,0 or 240, 320), then game will come to an end and display the “Game Over” screen.

1. **Difficulties encountered and Solutions.**

Some challenges that we encountered while writing the code for this project was writing the code for the duck to move it up, down, left, and right. The code was not compiling when we ran it to test it and we solved this problem by breaking it into parts and running it piece by piece so see what we needed to connect. Another challenge that we faced multiple times was dealing with small errors in our code and we solved this problem by going through it one by one to fix the problem.

1. **New Acquired skills/knowledge**

What we learned: how to use the monitor program and animation using c-code by running the code using DE-10 boards. We also learned how to use keyboard keys to control the animation like done for our game.

1. **Code with comments**

**Main game code**

*/\* Duck game final project by Manreet, Joe and Edmund*

*this is the main function used to run the game. It requires all*

*the library files listed below including the custom "map\_box.c" file.*

*\*/*

*//yellow 0xFF00, red 0xFA00, Green 0x1CE0, Blue 0x001F, black 0x0*

*//included libraries and files*

#include "address\_map\_nios2.h"

#include "nios2\_ctrl\_reg\_macros.h"

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "JTAG\_UART.c" *//must add this*

#include "map\_box.c"

*//Function prototypes*

void clear(void);

void ducky(int, int, int, int, short); *//x,y start and end, color*

void moveDucky(void);

void eatBox(void);

void VGA\_text\_display(int, int, char\*);

void game\_over(void);

void you\_win(void);

*//initial location of the main character(duck)*

int duckyLocation1[4] = {56 + 30, 53, 60 + 30, 58};

int duckyLocation2[4] = {50 + 30, 56, 55 + 30, 61};

int duckyLocation3[4] = {50 + 30, 56, 55 + 30, 61};

*//"score:..." shown at the bottom right of screen*

char text[20] = "Score: \0";

*//variables used in multiple functions*

extern int box[17][6]; *//array that includes details on the boxes for the map*

int sumScore = 0; *//total score variable*

int ones, tens, hundreds; *//varaibles for splitting up the score for displaying it on the screen as text*

int main()

{

*//clear the screen*

    clear();

*//X1, Y1, X2, Y2*

*//create the duck at its initial location*

    ducky(duckyLocation1[0], duckyLocation1[1], duckyLocation1[2], duckyLocation1[3], 0xFA00);

    ducky(duckyLocation2[0], duckyLocation2[1], duckyLocation2[2], duckyLocation2[3], 0xFF00); *//first*

    ducky(duckyLocation3[0], duckyLocation3[1], duckyLocation3[2], duckyLocation3[3], 0xFF00); *//second*

*//while loop for running continuously*

    while (1) {

*//separate the score into the ones, tens and hundreds*

        ones = sumScore % 10;

        tens = (sumScore / 10) % 10;

        hundreds = (sumScore / 100) % 10;

*//Place the score values into the "text" character array to be shown on the screen*

        text[9] = ones + '0';

        text[8] = tens + '0';

        text[7] = hundreds + '0';

*//if all the boxes not yet been grabbed*

        if (sumScore < 420) {

            moveDucky(); *//ability to move the duck*

            mapify(); *//create the updated map*

            eatBox(); *//edit any boxed that have been grabbed*

            VGA\_text\_display(50, 50, text); *//show the updated text on the screen*

        }

*//else if all the boxes have been grabbed, show the winning screen*

        else if (sumScore >= 420) {

            text[8] = '2';

            VGA\_text\_display(50, 50, text);

            you\_win();

            exit(0);

        }

    }

    return 0;

}

*//function definition to clear the screen*

void clear() {

*//clear the screen*

    ducky(0, 0, 319, 239, 0); *//0, 319 and 0, 239 and black = 0*

}

*//function definition to create the duck at a specified location*

void ducky(int *x1*, int *y1*, int *x2*, int *y2*, short *pixel\_color*)

{

    int pixel\_ptr, row, col;

    for (row = *y1*; row <= *y2*; row++) *//for each row*

    {

        for (col = *x1*; col <= *x2*; col++) *//for each column*

        {

            pixel\_ptr = FPGA\_ONCHIP\_BASE + (row << 10) + (col << 1);

            \*(short\*)(pixel\_ptr) = *pixel\_color*;

        }

    }

}

*//function definition to grab the keyboard input and move the ducks position based on key input*

void moveDucky() {

    volatile int\* JTAG\_UART\_ptr = (int\*)JTAG\_UART\_BASE; *//must add this*

    char c = get\_jtag(JTAG\_UART\_ptr); *//reading the input from the keyboard*

*//case statement for various inputs*

    switch (c) {

    case 'w':

        clear();

        ducky(duckyLocation1[0], duckyLocation1[1] - 5, duckyLocation1[2], duckyLocation1[3] - 5, 0xFA00);

        duckyLocation1[1] -= 5;

        duckyLocation1[3] -= 5;

        ducky(duckyLocation2[0], duckyLocation2[1] - 5, duckyLocation2[2], duckyLocation2[3] - 5, 0xFF00); *//first*

        duckyLocation2[1] -= 5;

        duckyLocation2[3] -= 5;

        ducky(duckyLocation3[0], duckyLocation3[1] - 5, duckyLocation3[2], duckyLocation3[3] - 5, 0xFF00); *//second*

        duckyLocation3[1] -= 5;

        duckyLocation3[3] -= 5;

        break;

    case 'a':

        clear();

        ducky(duckyLocation1[0] - 5, duckyLocation1[1], duckyLocation1[2] - 5, duckyLocation1[3], 0xFA00);

        duckyLocation1[0] -= 5;

        duckyLocation1[2] -= 5;

        ducky(duckyLocation2[0] - 5, duckyLocation2[1], duckyLocation2[2] - 5, duckyLocation2[3], 0xFF00); *//first*

        duckyLocation2[0] -= 5;

        duckyLocation2[2] -= 5;

        ducky(duckyLocation3[0] - 5, duckyLocation3[1], duckyLocation3[2] - 5, duckyLocation3[3], 0xFF00); *//second*

        duckyLocation3[0] -= 5;

        duckyLocation3[2] -= 5;

        break;

    case 's':

        clear();

        ducky(duckyLocation1[0], duckyLocation1[1] + 5, duckyLocation1[2], duckyLocation1[3] + 5, 0xFA00);

        duckyLocation1[1] += 5;

        duckyLocation1[3] += 5;

        ducky(duckyLocation2[0], duckyLocation2[1] + 5, duckyLocation2[2], duckyLocation2[3] + 5, 0xFF00); *//first*

        duckyLocation2[1] += 5;

        duckyLocation2[3] += 5;

        ducky(duckyLocation3[0], duckyLocation3[1] + 5, duckyLocation3[2], duckyLocation3[3] + 5, 0xFF00); *//second*

        duckyLocation3[1] += 5;

        duckyLocation3[3] += 5;

        break;

    case 'd':

        clear();

        ducky(duckyLocation1[0] + 5, duckyLocation1[1], duckyLocation1[2] + 5, duckyLocation1[3], 0xFA00);

        duckyLocation1[0] += 5;

        duckyLocation1[2] += 5;

        ducky(duckyLocation2[0] + 5, duckyLocation2[1], duckyLocation2[2] + 5, duckyLocation2[3], 0xFF00); *//first*

        duckyLocation2[0] += 5;

        duckyLocation2[2] += 5;

        ducky(duckyLocation3[0] + 5, duckyLocation3[1], duckyLocation3[2] + 5, duckyLocation3[3], 0xFF00); *//second*

        duckyLocation3[0] += 5;

        duckyLocation3[2] += 5;

        break;

    }

*//if the duck goes beyond the bounds of the screen, activate the game over screen*

    if (duckyLocation1[0] < 0 || duckyLocation1[0] > 319 || duckyLocation1[1] < 0 || duckyLocation1[1] > 239) {

        game\_over();

        exit(0);

    }

*// printf("%d\n", duckyLocation1[0]);*

*// printf("%d\n", duckyLocation1[1]);*

}

*//function definition for grabbing the boxes*

void eatBox(void) {

    int i, j;

*//checks if the duck's beak is within the bounds of any box*

    for (i = 0; i < 17; i++){

        if((duckyLocation1[0] >= box[i][0] && duckyLocation1[2] <= box[i][2]) && (duckyLocation1[1] >= box[i][1] && duckyLocation1[3] <= box[i][3])) {

            for (j=0; j<5; j++){

                box[i][j] = 0;

            }

            sumScore += box[i][5];

            printf("eaten\n");

            printf("%d", sumScore);

        }

    }

}

*//function definition for displaying text on the screen*

void VGA\_text\_display(int *x*, int *y*, char\* *text\_ptr*) {

    int offset;

    volatile char\* character\_buffer = (char\*)FPGA\_CHAR\_BASE; *// VGA character buffer*

*/\* assume that the text string fits on one line \*/*

    offset = (*y* << 7) + *x*;

    while (\*(*text\_ptr*))

    {

        \*(character\_buffer + offset) = \*(*text\_ptr*); *// write to the character buffer*

        ++*text\_ptr*;

        ++offset;

    }

}

*//function definition for showing the game over screen*

void game\_over(void) {

    clear();

    char message[30] = "GAME OVER!        \0";

    VGA\_text\_display(35, 29, message);

}

*//function definition for showing the 'you win' screen*

void you\_win(void) {

    clear();

    char message[30] = "GOOD JOB! YOU WIN!\0";

    VGA\_text\_display(35, 29, message);

}

**Map creation code**

*/\*File used to create the map for the duck game.*

*For final project by Manree, Joe and Edmund*

*\*/*

*//necessary libraries*

#include "address\_map\_nios2.h"

#include <stdlib.h>

#include <time.h>

*//function prototypes*

void boxify(int, int, int, int, short); *//x,y start and end, color*

void VGA\_text(int, int, char\*);

int generate\_random(int);

void mapify(void);

*//This is an array of the location, colour and score value for each box in the game*

*//X1, Y1, X2, Y2, colour, score*

int box[17][6] = {

{25, 20, 35, 30, 0xFA00, 20}

,{30, 40, 48, 60, 0x1CE0, 30}

,{100, 100, 150, 150, 0xFF00, 10}

,{300, 10, 310, 20, 0x001F, 50}

,{300, 10, 310, 20, 0xFF00, 10}

,{50, 50, 75, 75, 0xFA00, 20}

,{200, 200, 250, 215, 0x1CE0, 30}

,{300, 200, 320, 240, 0x001F, 50}

,{25, 200, 50, 240, 0xFF00, 10}

,{290, 100, 300, 110, 0xFA00,20}

,{250, 10, 270, 50, 0x1CE0, 30}

,{220, 220, 230, 230, 0xFF00, 10}

,{100, 10, 125, 35, 0x001F, 50}

,{200, 50, 230, 100, 0xFA00, 20}

,{20, 150, 50, 190, 0x1CE0, 30}

,{80, 160, 120, 190, 0xFA00, 20}

,{170, 125, 215, 155, 0xFF00, 10}

};

*//function definition for creating each box at the specified location and color*

void boxify(int *x1*, int *y1*, int *x2*, int *y2*, short *pixel\_color*){

    int pixel\_ptr, row, col;

    srand(time(NULL)); *//random generator seed*

    char ins[20] = "Get the boxes!    \0"; *//instruction to show on screen*

    VGA\_text(35, 29, ins);

    for (row = *y1*; row <= *y2*; row++) *//for each row*

    {

        for (col = *x1*; col <= *x2*; col++) *//for each column*

        {

            pixel\_ptr = FPGA\_ONCHIP\_BASE + (row << 10) + (col << 1);

            \*(short\*)(pixel\_ptr) = *pixel\_color*;

        }

    }

}

*//function definition for creaing the boxes at the various locations*

void mapify(void) {

    int i,j = 0;

    for (i = 0; i < 17; i++) {

        boxify(box[i][j], box[i][j + 1], box[i][j + 2], box[i][j + 3], box[i][j + 4]);

    }

}

*//function definition to show text on the screen*

void VGA\_text(int *x*, int *y*, char\* *text\_ptr*) {

    int offset;

    volatile char\* character\_buffer = (char\*)FPGA\_CHAR\_BASE; *// VGA character buffer*

*/\* assume that the text string fits on one line \*/*

    offset = (*y* << 7) + *x*;

    while (\*(*text\_ptr*))

    {

        \*(character\_buffer + offset) = \*(*text\_ptr*); *// write to the character buffer*

        ++*text\_ptr*;

        ++offset;

    }

}

*//function definition to generate a random number*

int generate\_random(int *range*) {

    int value;

    value = 1 + (int)rand % *range*;

    return(value);

}