University of York

C Programming for MSc Assessment 2022-2023

Exam Number: Y3916929

Requirements:

Explicit Requirements:

To design and build a computer game that is a version of the game of catch. The player controls an on-screen stick figure to catch a ball thrown by a computer-simulated thrower, and the goal is to catch the ball before it reaches the ground. The game must consider the following aspects.

- the flight of the ball must be subject to the effect of gravity;
- the simulated (computer) thrower varies the speed, the angle and the rate at which the balls are thrown
- the game should keep a score of the number of successful catches over a set number of throws;
- the use of sound in the game;
- making use of both the mouse and keyboard to control the game;
- making good use of C features such as function–s, structs and pointers;
- making good use of comments, formatted code and meaningful variable names but avoiding global variables.

Implicit Requirements:

The requirements that are implicit in this assignment are as follows:

- The program must be compatible with the Windows operating system and its API.
- The program must have proper error handling and logging capabilities.
- The program must have a user-friendly interface that is consistent with the Windows UI conventions.
- The program should be able to handle file and directory operations in Windows.
- The program should be able to handle Windows-specific system calls and libraries.
- The program must be able to run on multiple versions of Windows.
- The program should be able to handle the specific encoding used in Windows.
- The program should be able to integrate with other Windows features and applications.
- The program should be able to handle and process graphics using the appropriate Windows libraries.
- The program should be able to handle and process MIDI music and audio using the appropriate Windows libraries.
- The program should be able to handle and process audio and video in a way that is consistent with the capabilities of the Windows operating system.
- The program should be able to handle and process input from MIDI devices in a way
-
- that is consistent with the capabilities of the Windows operating system.

Analysis:

Aim:

To implement a black box approach, we need to test the program's inputs and outputs without looking at the code.

The user inputs are:

- Number of balls
- Keyboard input for changing the number of balls
- Initial x and y positions of the ball
- Gravity constant

Outputs:

- Ball's new x and y positions
- Animation of multiple objects (balls, stick figures)
- Sound effects (if audio is implemented)

Functional Requirements:

- The program should be able to animate multiple objects (balls, stick figures)
- The program should allow for the user to change the number of balls being animated.
- The program should be able to update the position of the balls based on a projectile formula.
- The program should be able to handle user input from the keyboard.

Outputs:

The outputs of the code include:

- Ball's new x and y positions: The program calculates the new position of the ball using the projectile formula and updates the ball's position on the screen.

- Animation of multiple objects: The program animates multiple objects, including balls and stick figures. The animation is based on the ball's new x and y positions and the stick figure's x and y position.
- Sound Effect: The code includes an amio.h library, which is used for audio input and output. If this library is implemented, the program will output sound effects.
- Text to screen: The code does not include any specific functions for outputting text to the screen. But it can be done using the standard library functions such as printf(), fprintf() and etc.

In summary, the program outputs a graphical animation of multiple objects, including balls and stick figures, and sound effects with those libraries which are implemented. The program also uses the standard library functions for outputting text to the screen.

Analysis of Process:

1. Initialization: The program defines constants for the window size, initial ball position, and gravity. It also creates a struct for the ball, which includes variables for the new and y positions, velocity in the x and y directions, and colour.
2. Ball hit boundary: The program checks if the ball has hit the boundary of the window by comparing the ball's new x and y positions to the defined XMAX, XMIN, YMAX, YMIN constants. If the ball has hit the boundary, the function returns true, otherwise it returns false.
3. Draw stick person: The program draws a stick person based on the position of the person and whether they are the thrower or catcher. It uses the circle and line functions to draw the head, body, legs, and arms of the stick person.
4. Update ball position: The program updates the position of the ball using the projectile formula. It uses the time, velocity in the x direction, and gravity constant to calculate the new y position of the ball. It also increments the x position of the ball by 1.
5. Set random ball values: The program sets random values for the velocity in the x and y directions of the ball using the rand_number function.

Mathematical Formulas:

1. Projectile formula: The program uses the following formula to calculate the new y position of the ball. Also we are using an increment operator by the position of 1 and calculate the Y position based on this projectile formula :

   y = (Initial Y position - (vy * time) + (GRAVITY * time^2)/2)

2. Circle formula: This formula is used to check if the ball is in inside a circle of radius of 15 units around the hand of the catcher.

   distance = sqrt((x2-x1)^2 + (y2-y1)^2)

Specification:

- The program is a demonstration of how multiple objects are animated in a C program, such as the catcher stickman, thrower stickman and the ball which is launched in a projectile which is supposed to be caught by the catcher stickman based on the arrow keys.
- The program uses a graphics library to display the animation on the screen.
- The program must draw a moveable stickman and animate random positions for balls.
- The program must be able to produce 3 chances for the user to drop the catch.
- The program will make the game restart after 3 missed catches and the user gets to choose the difficulty option again

User Interface Design:

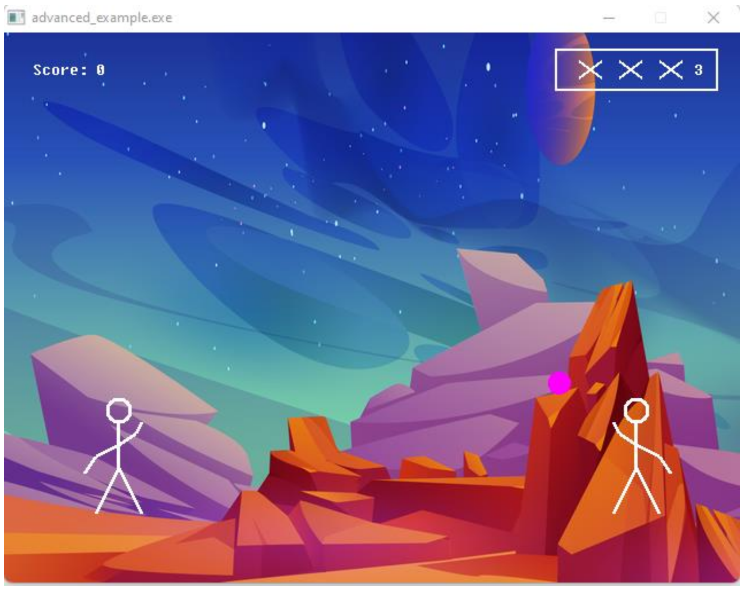


Image 1. UI of the game

The information required from the user is from two stages, one is during the actual gameplay and the other is during the starting of the game. The user is suggested to either select a "TRIAL" version of the game where there are no score or chances. The user can practise the game as it is and once they get a clear picture of what it is, they can proceed the actual game. Based on the decision of the user, the program navigates to either the game directly or to the "TRIAL". If the user chooses to play the game directly, they are led to a screen where they can select the difficulty option from the given two "EASY" or "DIFFICULT". If the user selects "EASY" option, the game will be a simple game of catch where the catcher has to just move around the stick man using arrow keys and catch the ball. Whereas if the user selects the "DIFFICULT" option, after the projectile of the ball is calculated, the ball is made to vanish for a few seconds and reappear again when it is near the catcher. This makes the reaction time for the catcher significantly less and makes the game more challenging for the user to play.

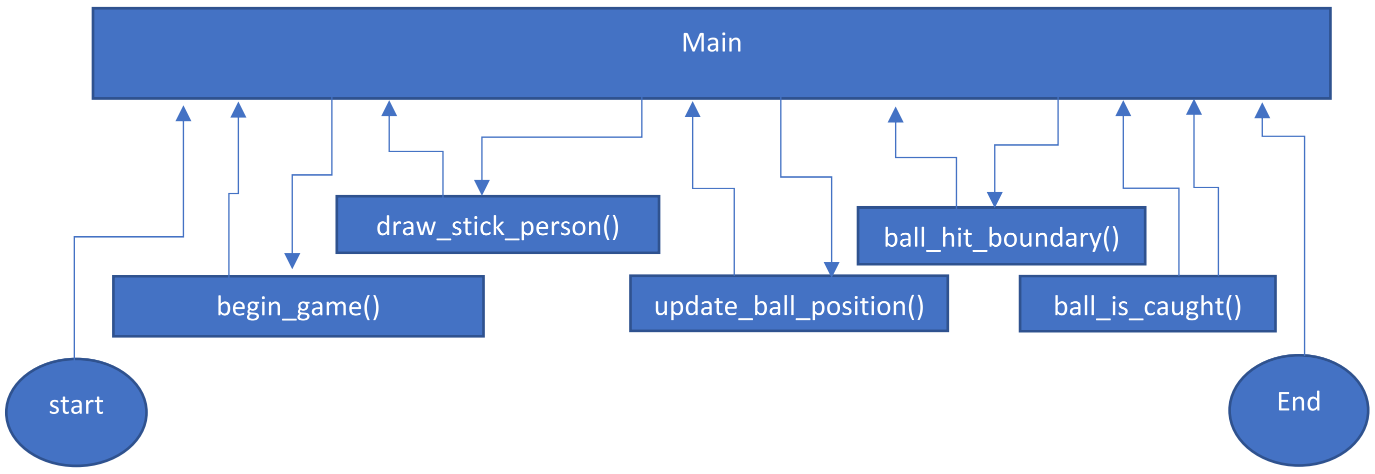
Pixel Ranges:

- The animation will be drawn within the defined XWINDOW and YWINDOW pixel ranges.
- XWINDOW is defined as 640 pixels and YWINDOW is defined as 480 pixels.
- The program will also check the X and Y positions of the ball and compare them to the defined XMAX, XMIN, YMAX and YMIN pixel ranges to check if it hits the boundary.
- The program uses the circle and line functions to draw the head, body, legs, and arms of the stick figure. The head_x, head_y, and head_radius variables define the position and size of the head.

User Input Validation:

The user input is expected to be numerical values. The input for the stick man should be keyboard commands, while the keyboard input for changing the position of the stickman should be a valid directional keyboard commands. Input validation is done by the program when the arrow keys is pressed by the user, the program checks if the input matches the predefined set of inputs which are the directional keys. In general, it is important to validate user input to ensure that it is in the correct format and that it falls within the acceptable range of values. This will help prevent errors and crashes in the program and ensure that the user has a positive experience with the game.

Structure Diagram.



Function Table.

Function name	Return Type	Arguments	Description
ball_hit_boundary()	Bool	XY coordinates(integers)	Checks if the ball has hit the boundary and returns true or false.
draw_stick_person()	Void	XY coordinates(integers)	Draws a basic stick person based on the position of te thrower and catcher.

update_ball_position()	Void	XY coordiantes(integers)	Updates the position of the ball based on the projectile formula.
set_random_ball_value()	Void	XY coordinates(integers)	Sets random values for the x and y velocities of the ball.
Main()	Int	None	The main function that calls the other smallish functions to animate the objects and play music.

Variable table:

Variable Name	**Type**	**Range**
XWINDOW	#define	640
YWINDOW	#define	480
XMAX	#define	XWINDOW - RADIUS
XMIN	#define	RADIUS
YMAX	#define	YWINDOW - RADIUS
BALL_INITIAL_X_POSITION	#define	120
BALL_INITIAL_Y_POSITION	#define	340
GRAVITY	#define	9.8
Ball	Ball	struct
Person	Person	struct
ball->x_new	double	0 <= x_new <= XMAX
ball->y_new	double	0 <= y_new <= YMAX
ball->vx	double	20 <= vx <= 40
ball->vy	double	40 <= vy <= 60
ball->color	uint8_t	0 <= color <= 255

person->head_x	double	XMIN <= head_x <= XMAX
person->head_y	double	YMIN <= head_y <= YMAX
person->head_radius	double	0 <= head_radius
Thrower	bool	true or false

Pseudocode:

```
Define struct Ball {
    double x_position, y_position;
    double x_velocity, y_velocity;
    uint8_t color;
}

Define struct Person {
    double    head_x_position;
    double    head_y_position;
    double    head_radius;
}

Define function ball_hit_boundary(ball)
    if (ball's x_position is out of bounds or ball's y_position is out of bounds)
        return true
    else
        return false

Define function draw_stick_person(person, thrower)
    Draw circle for head using person's head position and radius
```

```
   Draw line for body using person's head position and body size
   Draw lines for legs and hands using person's coordinates

Define function update_ball_position(ball)
   Calculate time using ball's x position and velocity
   Update ball's y position using velocity, time and gravity
   Update ball's x position

Define function set_random_ball_values(ball)
   Set random values for ball's x and y velocity

Define main()
   Initialize graphics window and audio
   Initialize person and ball variables
   Loop
      Begin game
      Check for keyboard events
      Update ball position
      Draw stick person and ball
      Check if any ball hit boundary
      If  ball is caught
   End loop
   Close audio
```

Testing and Verification:

One of the key features in the projectile of the ball was to make sure that the ball movement was in a clean smooth projectile. Initially the ball movement was recorded as a staircase approach where the ball was moving in a straight line for a certain time and then increased the Y axis value and this was repeated in a loop. This was happening due to the data type assigned to the function Update_ball_position(). Initially it was assigned as int and what happened was it kept rounding off to the nearest integer. To solve this problem, I had to change the data type from int to double. What this achieved is that it did not approximate the values of the function and printed it using the graphics library as it is and it made the movement of the ball much smoother. Also, the implementation of difficulty settings in the game was a challenge. As it was just a simple game of catch, it was quite repetitive after a while, so I tried out an idea of making thing a bit more interesting. The initial idea was just to increase the number of balls and increase the speed of the balls, but that felt a bit lack-lustre. So the final idea which I personally feel that is innovative is to make the ball disappear after the projectile is initiated. Which will make it harder for the player to catch the ball as the user would have to just predict the projectile of the ball.

Logical Flow:

The code begins with the creation of two structures, one for the Ball and one for the Person. The Ball structure holds information about the position, speed, and color of the ball, while the Person structure has details about the head's location and size of the stick person. The code then defines multiple functions that are utilized throughout the game. The ball_hit_boundary function checks if the ball has hit the edge and gives back a true or false value. The draw_stick_person function is accountable for drawing the stick person on the screen. The update_ball_position function updates the ball's position based on the projectile formula and the set_random_ball_values function sets random values for the velocity of the ball. The main function of the code is responsible for initiating the graphics window, audio, and variables for the number of balls and the person. The code enters a loop where it constantly checks for keyboard events, updates the ball's position, draws the stick person and the ball, and checks if any ball has hit the boundary. Once the game ends, the audio and graphics window are closed. The game takes into consideration the physics of the thrown ball, including gravity, and the player's ability to move the stick person on the screen to catch the ball. The game also includes the option to change the number of balls, as well as additional features such as bitmap and audio examples. In general, the code is well-structured and organized, making use of functions and structs to separate different tasks and make the code more readable. The use of object-oriented programming concepts makes the code easy to understand and maintain. The game has been updated in November 2022 by adding audio example, adding the ability to change the number of balls, adding keyboard example.

User Manual:

Intended User:

The Intended user of this game is interested in the language of C and is interested in developing various programs in the C language. The user can also be a gaming enthusiast who is interested in just experiencing the game and keeping track of the score and making the game a bit easier or difficult depending on how the experience goes. Also, the user is required to have a basic understanding of how to operate a computer, including navigating the file system and running programs and additionally, knowledge of how to play the game would be required, such as understanding how to control the stick figure and catching the ball.

Frequently asked questions:

1 How can I install the game on my computer?

A: You can install the game by downloading the source code from the internet or from the developer, installing a C compiler and IDE, installing the necessary libraries and frameworks, and building and running the program.

2 Is the game compatible with MacOS?

A: It depends on the libraries and frameworks used in the game. Some libraries may not be available for MacOS, in that case, the game may not be compatible with MacOS. And also, there is no code blocks IDE available in MacOS.

3 What are the minimum system requirements to run the game?

A: The specific system requirements will depend on the programming languages, libraries, and frameworks used to develop the game. In general, you will need a computer with a supported operating system, such as Windows or Linux, a C compiler and IDE, a graphics library, an audio library, a library for handling input from the keyboard, and a library for handling the physics of projectile motion.

Here are some general installation instructions for the game,

- Download the source code for the game from the internet or from the developer.
- Install a C compiler and IDE on your computer, such as GCC or Clang and Visual Studio Code or Code::Blocks.
- Install the necessary libraries and frameworks that are used in the game, such as graphics library, audio library, input library, and physics library.
- Open the source code in the IDE and build the program.
- Run the program and the game will start.
- here are some general system requirements that may be necessary to run the game:
- A computer with a supported operating system, such as Windows or Linux.
- A C compiler and IDE to build and run the program.
- Graphics library or library that provides support for drawing graphics on the screen.
- Audio library or library that provides support for playing audio files.
- A library that provides support for handling input from the keyboard.
- A library that provides support for handling the physics of projectile motion.
- A library that provides support for the animation of the ball and stick person.
- A library that provides support for the random number generator.
- A library that provides support for the math operations.
- A library that provides support for the data types.

Here is a basic user manual for the game:

- Start the game by running the program on your computer.
- The game will open a window with an on-screen stick person and a ball.
- The objective of the game is to guide the stick person to catch the ball before it hits the ground.

- Use the left and right arrow keys to move the stick person left and right to catch the ball.
- The computer-simulated thrower will throw the ball and it will move across the screen.
- If the stick person successfully catches the ball before it hits the ground, the player will earn points. If the ball hits the ground, the player loses a life.
- The game is over when the player loses all their lives.

Bibliography

- https://stackoverflow.com/questions/48157930/cannon-projectile-program-using-c-programming
- https://www.geeksforgeeks.org/projectile-motion/
- https://homework.study.com/explanation/write-a-complete-program-that-computes-the-duration-of-a-projectiles-flight-and-its-height-above-the-ground-when-it-reaches-the-target-the-required-program-displays-instructions-to-the-program-user-a.html
- Cameron Harwood in Nov 2022 – advanced_example.zip