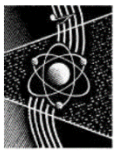


THE UNIVERSITY of York
Department of Electronics

Introduction to Programming in C (ELE00002C) Assignment Report (Undergraduate)

Examination Number: Y3834764

Content

Requirements……2

Analysis……2

Specifications……3

Design……3

Implementation Report……7

Verification and Testing……8

User Manual……9

Requirements

I am required to design a computer game for my assignment which includes launching an object through the program window, avoiding obstacles, several difficulties and environmental effects such as gravity and wind. This program is written in C, using several libraries including the *graphics_lib.h*, the graphical library which is provided by the university, and is expected to be executable only in Microsoft Windows.

Analysis

Since the *graphics_lib.h* is expected to be used, the whole program should be like a command window for display introductions and for user to input the data, and a graphical window which will show some shapes and images that related to the game, as well as the result of user's input. The content in the command window should be straightforward since all the user controls will be done here. In the graphical window, all the shape should be in the right place and movements should be reasonable.

For launching an object, initial velocity and angle are needed. I assume the unit for velocity is m/s and using degree for user input the angle. The unit of gravity and wind speed is m/s^2 and m/s in correspond to the unit of the velocity. Wind direction is ranging from 0 °to 225 °in respect of the ground. Obstacles are individual blocks which are squares with a side length of 30 centimetres.

The whole program starts with an introduction of the game, then let the user select the difficulty level, from 1 to 5. Higher difficulty level will result in more obstacles, but more chance for getting high score. After that, it should initialize an 800x600 graphical window then the user should get the information of the wind direction and speed then adjust the initial position before launching the object. Then user is expected to input the velocity and the angle for the object according to the wind and the position of the blocks, which their position are randomly generated. After the object is launched and user gets the score, the program will enter next level.

The angle that user selected is ranging from 0 °to 90 °, in the program this has to be converted into radius, using **Angle * (180 / π)**. For the object, the velocity on the horizontal and vertical direction is derived from the initial speed that user gave. Horizontal velocity is **user input velocity * cos (Angle)** and vertical velocity is **user input velocity * sin (Angle)**. Both will be effect by wind speed and the direction of the wind. The final horizontal velocity is **user input velocity * cos (Angle) + wind speed * sin (wind angle)** and the vertical velocity is **user input velocity * sin (Angle) + wind speed * cos (wind angle)**. Velocity on the horizontal direction remains constant from now on but the one on the vertical direction will be effected by the gravity, using **vertical velocity + gravity * time**. The whole movement will stop when hitting the ground.

Since it is a simple game, my user could be normal players, who probably have no knowledge on programming at all but have some experience in real life of throwing out a ball or something else and that's why I use degree when input the angle rather than radius but will later be converted into radius for the program itself.

Specification

Mandatory:	Optional/Extra:
1. Intro	1. Generate obstacles at random position
2. Ask user for difficulty level	2. A timer
3. Initialize graphical window	3. Create 'portal blocks'
4. Generate wind speed and direction	4. BGM will change when time is running out
5. Draw man, ground and obstacles	5. An object instead of a projectile line
6. Ask user for velocity and angle	6. Create a local *.txt file for leader board
7. Draw the projectile line	
8. Calculate and display score	

Design

A timer is implanted in this program, using multithreading to make the timer and the program work together. An integer a = 60 will minus 1 after each second using Sleep(1000). At the first 30 seconds, using PlaySound play sound file *looping_radio_mix.wav* . When a = 30, message appear on the screen '30 seconds left', play sound file 'MP_DM_COUNTDOWN_30_1.wav'. Sound will stop when a = 0, message '10 scores penalty awarded'.

The timer is packed into function thread() and will not be executed until user is ready.

Once the user execute the program, instructions will show up in the command screen:

Welcome to Bendy Theft Ball I
After the welcome message, you are able to adjust Bendy's position
Press <-- or --> to move the Bendy
Then press ENTER to continue
After that, you need to set the initial velocity and angle (0 °~90 °) of a ball in Bendy's hand.
Wind will affect the ball's motion, information will be given.
Score will be given by which slot the ball is in multiplied by the difficulty level.
Hitting the left side of any block will cause score invalid.
Both top and bottom of the block are accessible, 5 points will be awarded if hitting the inner right side of the block
You have 1 minute after the intro.
10 points penalty will be awarded if running out of time.
Get ready.
Countdown will start after the difficulty level is selected.

There's a 1.2 second pause between each message and a 2 seconds pause before the last message is shown, contained in function welcome_msg(void).

After the welcome message is shown, the program would ask the user for difficulty level:

Input difficulty level (1~5):

User is expected to input an integer from 1 to 5, if user's input is not an integer, or exceed the expected range, program would keep asking user for the level until the user give an expected input. This integer will be stored as diff_level.

Before the graphical window is initialized, call srand(void) function from stdlib.h for generating a random number. In generate_wind(double *wind_speed, double *wind_direction, double *wind_angle) function, double wind_speed is a random number ranging from 0~24, indicating the wind speed, double wind_angle, ranging from 0~225 representing the angle and double wind_direction, is the radius form of angle, will be used in further calculation. Command window provides information about the wind direction and the speed for user.

The graphical window will be set as 640x480. Background colour is light blue, and a brown rectangle, starting from (0,400) end at (640,480) as ground.

The stickman figure (called 'Bendy') consists several basic shapes: 2 filled circles with radius 17 and 22, as head and upper body, a 44x37 rectangle as body and two 13x24 rectangles as legs.

The exact position of shapes are:

Head: Centre of circle (92, 307)

Upper Body: Centre of circle (92, 345)

Body: Start (70, 341) End (70+width, 341+height)

Left leg: Start (78, 376) End (78+width, 376+height)

Right leg: Start (94, 376) End (94+width, 376+height)

Position of obstacles are randomly generated but have to be limited in a reasonably range. Quantity of the obstacles depends on the difficulty level. Using for loop to generate obstacles, loop for 'diff_level' times.

Score section on the ground, marked with number 10, 20, 30, 20, 10. Higher score meaning tighter section. Using yellow short line to separate each sector.

100 width for 10 scoring sector, 85 width for 20, 20 width for 30 score.

Markers position: (200, 400) (300, 400) (385, 400) (420, 400) (505, 400) (605, 400)

These should be packed into function score_section(void).

While the graphical window are initialized, the timer will start counting with the sound. Before hitting enter, user could adjust the position of Bendy using ← (return value 77) and → (return value 75) on the keyboard. When pressing arrow keys, using copy_array function to store the information of old position, cover it with light blue shapes, then show the new position of Bendy in black.

When hitting enter, we require user to input the value of velocity and the angle for the ball is about to be thrown. Using InputCheck function to keep asking user for the appropriate value until user does. Angle is limited between 0° and 90°.Convert the

angle into radius using angle / 180 * π then dissolve the velocity value into vel_x and vel_y. vel_x = velocity * cos (angle), vel_y = velocity * sin(angle). On the horizontal direction, dissolve wind speed as well, final vel_x and vel_y should be affected by the wind separately.

Move to Bendy's hand (105, 330), assign these coordinate to x_position, y_position. The movement of the position is explained in the Analysis section. During the movement, keep assign the old position to x_position, y_position for covering the old ball then draw the new ball using new values. In this loop, obstacle_col() function is used to find whether the ball hits the blocks. When the edge hitting the block, ball will drop to the ground immediately, scoring 0. As the ball hit the ground, play sound 'score.wav'

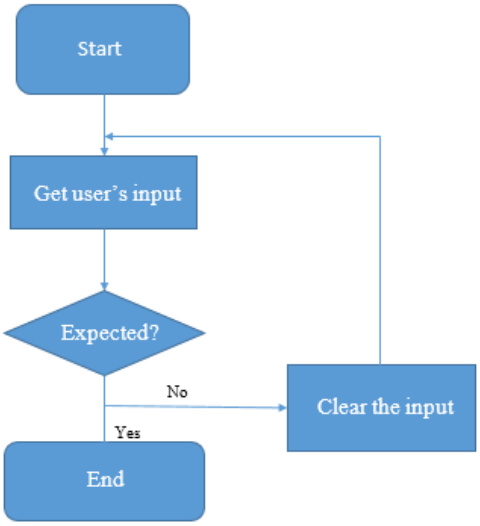
The final score depends on where the ball is. If hitting the block, total score is 0 no matter which scoring part the ball hit. 10 scores penalty for not finish the round in 1 minute. The actually score will be multiplied by the difficulty level. Then press any key to quit.

Variables table:

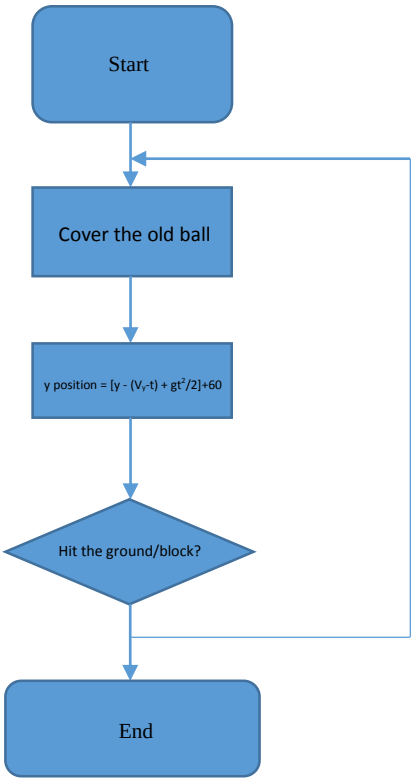
Type	Name	Description
int	a	timer
int	total_score	The result of this round
int	x[5]	Arrays contain all x position of Bendy
int	y[5]	Arrays contain all y position of Bendy
int	input_validation	For validating the input
int	diff_level	Difficulty level
int	wind_speed	Wind Speed
int	wind_angle	Wind angle
int	wind_direction	Wind angle in radius form
int	ob_gen_counter	counter to generate the obstacles
int	flag	flag to tell whether the timer thread is working or not
int	old_x[5]	old x position of Bendy
int	old_y[5]	old y position of Bendy
double	vel	user input velocity
double	vel_x	velocity on horizontal direction
double	vel_y	velocity on vertical direction
double	angle	user input angle
int	x_position, y_position	position of the ball

Algorithms:

InputCheck:



Ball's Movement:



Implementation Report

List of header files used in the program:

File name	Usage
stdio.h	use of printf, scanf
stdlib.h	use of srand
conio.h	use of getch
time.h	use of Sleep
windows.h	use of PlaySound
process.h	use of multithreading
math.h	use of calculation
graphics_lib.h	use of graphical functions
bendy.h	split the program into functions

List of functions in bendy.h

Function	Type	Usage
draw_bendy	void	Draw the Bendy figure
draw_new_bendy	int	Draw the new Bendy during the movement
cover_bendy	int	Cover the old Bendy during the movement
copy_array	int	Copy every data from an array into another array
generate_obstacle	void	Generate an obstacle at random position
InputCheck	int	Input validation
score_section	void	Draw the score section
generate_wind	void	Generate 2 integers indicate wind speed and angle
welcome_msg	void	Show the welcome message

List of sound file used

File name	Usage
looping_radio_mix.wav[1]	Background Music, At first 30 seconds
MP_DM_COUNTDOWN_30_1.wav[2]	Background Music, 30 seconds countdown
score.wav[2]	Sound effect when scoring
wind.wav[2]	Wind sound effect, mixed with looping_radio_mix.wav and MP_DM_COUNTDOWN_30_1.wav

While I was doing the implementation, I compromised on the obstacle colliding detecting system, the ball will go through the blocks from top or bottom side, but still will drop to the ground when hitting the left or the inner right side of the block. By adjusting the scoring system, user still get 0 score if hits the left side, but will get 5 scores award when hitting the inner right side (as I found it's very difficult to do). Everything else is designed as expected.

---

1 Extracted from The Stanley Parable, Copyright (c) Galactic Cafe 2013 All rights reserved.

2 Extracted or captured from Grand Theft Auto V Copyright (c) Rockstar Games Inc. 2015 All rights reserved.

Verification and Testing

Stage 1: Expect input and output

Input	Expected input	Expected output	Result
Difficulty level	1, 2, 3, 4, 5	Into next step	√
Keyboard Control	<-- or -->	Bendy moves around horizontally	√
Velocity	Every real number	Ask user for angle	√
Angle	0~90, integer	Ball will be thrown	√

Stage 2: Unexpected input and expect output

Input	Unexpected input	Expected output
Difficulty level	Any number larger than 5 or less than 0, any non-integer input	Program notify the user to input a proper number
Keyboard Control	Any key rather than <--, --> or Enter	Nothing happened
Angle	Any number larger than 90 or less than 0, any non-integer input	Program notify the user to input a proper number

Stage 3: System Requirements

Due to compiler and environment issue, this program can only run in Windows 7. Windows 8 and higher version is not compatible with this program. As I tried to rewrite and rebuild this program on Windows 8 and Window 10, same problem occurs as the program will crash after the graphical window is initialized. Further solution is to take out every function block to make a frame then compile in higher version of Windows, then add up every single functions into it. Due to the limitation of times, I didn't realize this solution but will have further trial.

Stage 4: Known Bugs

Bug	Status
No message shown when input difficult level out of the range	Not fixed
'Shade' behind Bendy	Debugged
Ball drops to the ground to early	Debugged
Timer keeps counting after score is given	Not fixed
Lag caused by keeping refreshing scoring section	Not fixed
Bendy and the ball will 'take the obstacle apart'	Not fixed
Ball keeps dropping when hitting the ground	Debugged

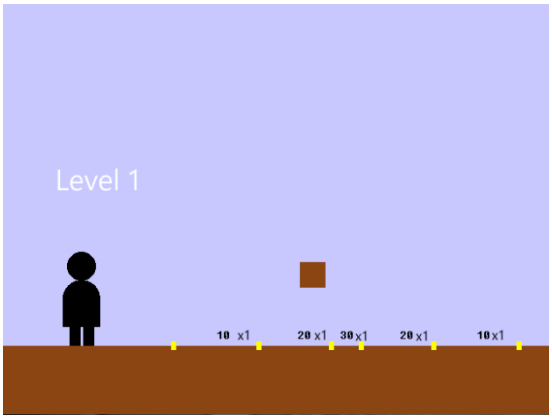
User Manual

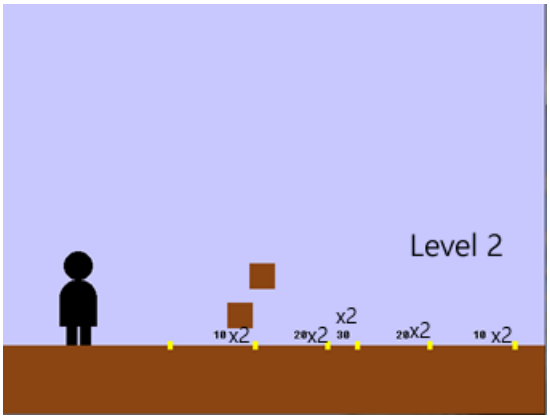
Installation: Copy the file to wherever you like, be sure ‘data’ ‘music’ folders, libportaudio-2.dll portmidi.dll are in the same folder where the exe file is.

System requirements: Windows 7 exclusively, as the system will crash in Windows 8, 8.1 and 10. *

Instructions: There will be an intro when you start this game but a clear instructions is here:

You have 5 difficulty levels (1~5 as easy ~ hard). Level 1 has 1 obstacle, level 2 has 2 obstacles, and so on. But the score you get will be multiplied by the level.





A clear message is shown in the terminal window of information of the wind.

```
Wind speed: 16
Wind direction : ↗ Angle: 34
```

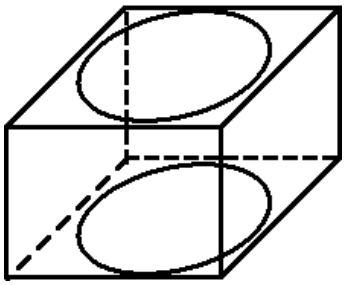
*Copyright Microsoft Coperation & Facebook Inc.

Please note: Keep the terminal window activated (i.e. click the terminal window after the graphical window is shown).

Before hitting enter, you are able to move around Bendy using ← and →.

After hitting enter, you have to enter the velocity (m/s) and angle (ᵒ) for a ball you are about to throw. Think through about these as wind would affect ball's movement. And remember, you only have 1 minute after you entered the difficulty level otherwise you will be given 10 points penalty!

Each obstacle has a shape like this →



As you can go through the top and bottom of each block. By doing that you can hit the inner side of this block, you will be awarded 5 extra points.

BUT! Hitting the left side of the block will clear your score!

So this is just a simple game, don't expect too much, have some simple fun!