C programming Assignment
Graphics Game Report

Student Exam ID: Y3852394

CONTENTS

1. Assignment abstract

This C programming assignment is required to design a graphical game to entertain bank customers waiting to see personal banker, applying with some properly physics behaviors displayed by a coin following invisible projectile trace. As required, the player seeks to avoid at least one physical obstacle and the path of the object should conclude influence of gravity.

Besides, considering improving playability, a rewarding and challenging experience must be included. There are three different piggy banks pointing to different scores and a restriction, which have either random position or height every time the user starts the game. To encourage player to play again, rewarding the player with score and assuring the game is not too complicated to go on should be considered.

There also should be at least one extended feature be applied into the game, for instance, adding musical soundtrack when the game is proceeding, designing animation for coin or applying additional game mechanics. As suggestion, in terms of ensuring users to get the score in formal game, illustration can be designed as a different button from start in starting window.

From a top view perspective, the game also implied some other key requirements. Firstly, it's definitely edited by C programming language and the game development software is CodeBlocks operated under Linux. Then, the game functions, libraries, head files are customised by any possible approaches that can be used from lab exercises. Because the game is provided for bank customers, bank theme is expected to have.

As following, this report will analyze problems of this game, including limitation to be improved in future. At the same time, specification covering design and features in detail will be introduced. Last but not the least, there will also be an evaluation from designer's perspective.

2. Problem analysis

2.1 Description of the game

As for the size of game window, it should be 800*640, which limits all operations have to happen in this range. The first step of the game is to allow users to choose the colours for a stick man who represents themselves. Typing valid number from 0 to 10 in terminal, which represents different colours, can change colour for stick man. Any invalid typing will cause black to draw the stick man. After choosing colour, pressing 'ENTER' on keyboard causes the interface to draw stick man, targets as well as the restriction. Positions of stick man is stable, decided by coordinate of the centre of face, which is (80,480). However, the height of restriction with width of 40 and position of target are random.

In main function of this game, three main performances of stick man are defined to realise moving right, left and throwing the coin. As the designer is thinking, movements of stick man to go left and go right are commanded by 'left' and 'right' keyboard. Due to the obstacle and left frame, the movement of stick man is just allowed from 50 to 360 pixels. Typing 'T' will start logic of throwing the coin, which needs mouse click to throw the coin. By applying with the direction and velocity of the coin using a power bar instead of asking users to input number manually in the terminal, the coin will follow an exact projectile trace after left mouse button has been clicked. The only way to get scores is to throw coins into the piggy bank in three opportunities. To inform users of how many velocities, angles they attempted, information at top right corner is updating once the indicator on power bar is moving.

At the end of the game, if users hit the target, the interface will draw a ‘congratulation’ interface including score and the analysis including the maximum, minimum and mean distance of trace thrown. Otherwise, the interface will just draw the analysis. Without any other operation, the game will start again in 5 seconds.

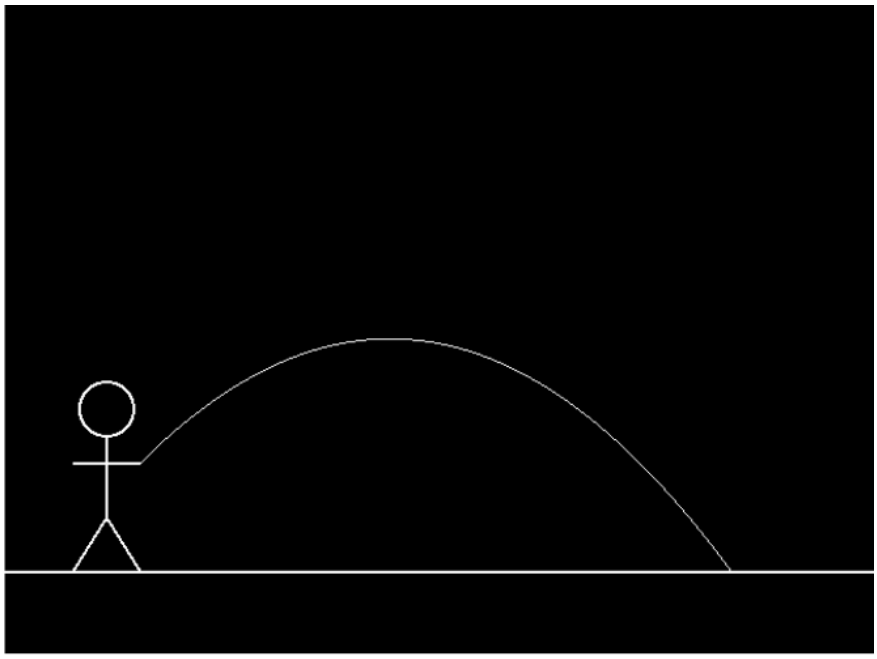
2.2 General approach

As for general approaches to creating this game, drawing basic elements such as background and stick man constituted by different colour and geometric figures is where to start firstly. By means of functions from <**graphics_lib.h**> provided by university, drawing a smile face and body of the stick man manually becomes possible and easy, which only needs angles and length for circle as well as rectangle.

At the same time, knowing how to draw a projectile is also a key point. Some physics knowledge such as free fall law is used here and transmitted into C language. Meanwhile, rather than draw a curved line, approximating a line by drawing a lot of tiny straight line segments using differential thought is realised by a loop. In order to create an animation to make throwing coin closer to real, a light cyan rectangle at position of precious coin, which is the same principle as stick man’s movements is employed. Updating only position of coin decreases storage burden in case of splash screen meanwhile shorten the length of code.

2.3 Physics laws and models applied

For the sake of corresponding to actual scenario, the program should follow mathematics formulae and physics law. In this game, projectile model is used when throwing the coin to target and power bar model is used when deciding angle and initial velocity.



As users give one velocity and angle, what the program should work out are horizontal velocity and vertical velocity separately. Assuming no air resistance, ***Newton’s second law ($P_y = P_{y0} - V_y t + at^2/2$) and Speed-distance-time triangle ($P_x = P_{x0} + V_x t$)*** plays an indispensable role in calculating every coordinate that projectile goes across. Considering effect of gravity, ***$a = g = 9.81 m/s^2$.***

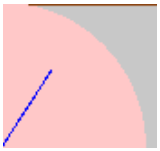
To use two essential formulae in C language, some variables need declaring:

(P_x ,P_y) is current location of the coin

(P_{x0} ,P_{y0}) is original location of the coin, the same as position of stick man's right hand

V_x and **V_y** are horizontal velocity and vertical velocity respectively in m/s

t is time in seconds



The core formula of power-bar model is distance between two points $|AB| = \sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}$. With the assistance of square roots function **sqrt()**, the distance between end point and original point of indicator is able to be solved. **(x_1 , y_1)** and **(x_2 , y_2)** are coordinate of bottom left corner and positions of mouse button click respectively. In the meantime, using anti-trigonometric function **atan2()** can work out angle between indicator and horizontal plane are able to be calculated. To ensure the coin could step over restriction due to random height of restriction, the initial velocity is 1.2 times as much as length of indicator on power bar, ranging from 0-135.8.

2.4 Special limitation

As required, using GFX library is one of the special condition that wouldn't be required outside of the assignment. Normally, double-clicking exe.file will start the game immediately. But this game is related to choosing colours in terminal, which means this cannot be realised even cause this game is stuck somewhere.

3. Specification

3.1 User inputs and outputs

User input is one of the essential aspects that games should satisfy. The game might get console input from the keyboard by pressing some keys or from the mouse only or combining them together. All types of user inputs should be permitted and should never exceed the game's expected range. In this game, operation using mouse button and keyboard are inserted and used in different occasions.

In starting interface, program recognises mouse click and the interface will update only when mouse click valid text. In game process, when users are choosing colour for stick man, program realises numbers typed in terminal to give an appropriate colour for stick man. Apart from this, all inputs should be from in game window. Moving stick man needs 'left' and 'right' button on keyboard and pressing once could let stick man move 20 pixels. In terms of convenience in case that users switch between terminal and the game window, the initial velocity and angle can be controlled by way of moving the indicator on power bar after pressing 'T' on keyboard. When user decide the direction, clicking left mouse button, the animation of coin will release.

As to the output, it may come from vary aspects. It could contain graphics, musics and up-to-date text information for different velocity, angle and score with the aid of game window. In the initialised window, illustration button and start button lead to different interface. The illustration button will give users a tutorial, similar to real game, ending until users get scores. For letting users know that they get score, the game window will give a congratulation reminder and the real game will begin then. Game-over window is drawn at the end of the game, giving minimum, maximum, mean distance as well as final score according to the logic function. After that, the welcome window should pop up any relative information and the player could therefore play again. During the game, player can click 'X' icon on the top right of the terminal window to quit it.

3.2 Library

System-provided <**math.h**> is essential for developing the algorithm as well as formulae.

<graphics_lib.h> which is provided by university has complete information and functions used to write graphical program and decreases difficulty to draw figures out and design the whole program.

Function Name	Parametres	Description
GFX_InitWindow()	X: width of window in pixels Y: height of window in pixels	Initialising a 800*640 graphics window
GFX_CreateEventQueue()	(none)	Called before registering any events
GFX_RegisterDisplayEvents()	(none)	Registering display events with a queue
GFX_InitFont()	(none)	Initialising the graphics font system
GFX_InitKeyboard()	(none)	Initialising the keyboard
GFX_RegisterKey-boardEvents()	(none)	Registering keyboard events with a queue
GFX_InitMouse()	(none)	Initialising the mouse
GFX_RegisterMouseEvent()	(none)	Registering mouse events with a queue
GFX_InitBitmap()	(none)	Initialising the bitmap structure
GFX_SetBackgroundColour()	Colour: Black	Black background in welcome window
GFX_SetColour()	Colour: 13 colours are defined in li-brary.	Set different colour pen to draw figures or write texts on the game window
GFX_DrawText()	X: start x-coordinate Y: start y-coordinate constant char: text as string able to be written on the screen	Write text using format set up in library on the screen. Used when drawing reminders, designing starting and over window and up-dating score, velocity and angle.
GFX_UpdateDisplay()	(none)	Display figures since last update
GFX_WaitForEvent()	(none)	In starting interface and 3 performances of stick man to instruct the program to wait un-til an event occurs
GFX_IsEventMouseButton()	(none)	Check if mouse button is clicked
GFX_GetMouseButton()	MOUSE_BUTTON_LEFT	Identify which button was used
GFX_GetMouseCoordinates()	(&x,&y) (&x_mouse,&y_mouse)	Once mouse is clicked, get coordinates of the mouse at the point when the event occurred

GFX_DrawFilledRectangle()	upper_left_x and upper_left_y: coordinates of upper left corner lower_right_x and lower_right_y: coordinates of lower right corner fillcolour: chosen from 16 colours defined in library	Sky and ground are defined into different colours separately using two light cyan and light grey rectangles. To update data for score, velocity, angle even the stick man once moved, a rectangle with background colour is drawn.
GFX_DrawFilledCircle()	centre_x = 0, centre_y = 640 radius = 80 fillcolour: Lightmagenta	This is just used in drawing the one quarter circle for power bar
GFX_DrawCircle()	X and Y: x_position and y_position radius: 20 thickness: 4	Draw the head of stick man
GFX_RandNumber()	Int: low range int: upper range	Both restriction height and x-coordinate of target are random
GFX_PauseFor()	Int: 5000/3000	Wait for period in milliseconds
GFX_LoadBitmap()	Name of 6 bitmaps	Load pictures into program
GFX_MoveTo()	X: initial_pos_x Y: initial_pos_y	Draw projectile from coordinate of right hand after moving the stick man
GFX_DrawLine()	xstart, ystart: start coordinate xend, yend: end coordinate thickness: 4	Draw stick man's body
GFX_DrawLineTo()	X and Y = pos_x and pos_y thickness: 1	Due to changing initialised point, this allows to draw a tiny line to points calculated, playing a role in drawing projectile
GFX_DrawBitmap()	Image: name of 6 bitmaps x, y: coordinate of the centres of images	Declaring 6 bitmaps: welcome.png, blue_target.png, green_target.png, red_target.png, coin.png, congratulation.png Drawing bitmap on screen
GFX_FreeBitmap()	6 bitmaps to delete to release memory	Delete an image from program memory that's no longer in use (saving memory)
GFX_DrawArc()	centre_x, centre_y: centre of circle from which the arc is derived radius: circle from the arc is derived angle_start, angle_end: angle in degrees at which the arc should start and end thickness: line thickness = 2	Draw face of the stick man implied into appropriate radius and on the basis of centre of the circle

GFX_GetKeyPress(&key)	ALLEGRO_KEY_LEFT ALLEGRO_KEY_RIGHT ALLEGRO_KEY_T	Pointers to integer to store the index of the key that has been pressed. The names to identify keys are in the format of ALLEGRO_KEY [key].

Besides, to make coding tidier, for different figures or parts used a lot of times in this game for some reason, they are created as functions, which can be invoked by a line instead of all codes. 12 functions are declared at beginning and defined after main function, which are:

Function Name	Parametres	Description
void changecolour**()**	int user_input1	Get numbers users input and give appropriate colour to draw stick man
double Mean**()**	double values**[]** int num_values	Calculate mean distance of 3 trying
double draw_projectile_path**()**	double initial_pos_x double initial_pos_y int ***proj_x int restriction_height int x_target int ***proj_y	Draw coin following a projectile trace based on position of right hand. Strop drawing the projectile based on position of the x-coordinate of target
void draw_target**()**	int x_target	Draw bitmaps as 3 targets
int score_board**()**	int x int distance int y	Return appropriate score when hitting different target
void draw_stick_man**()**	double x_position double y_position	Draw stick man based on position of centre of his head
void draw_restriction**()**	int restriction_height	Draw restriction based on random heighest point
double maximum**()**	double array1**[]** int size	Acquire maximum distance in three trying
double minimum**()**	double array2**[]** int length	Acquire minimum distance in three trying
void illustration**();**	int x_position, int y_position int user_input1 int initial_pos_x**,** int initial_pos_y int mark**,** int key int proj_x**,** int proj_y char string_mark**[**11**]**	Illustration logic – exit when user hits the target and enter the real game
void over_interface**()**	int mark**,** int max**,** int min double mean_distance char string_mark**[**11**]**	Draw over interface which give analysis of the game to users
double draw_power_bar**()**	double ***velocity	Draw the power bar used to adjust initial velocity and launch angle

3.3 Design

In this part, more specific expansion over what this program can do and what reaction of this game will give after some operation users do is introduced. When the program is set ready to compile in CodeBlocks, after being built and run, the welcome interface appears, giving three options which are **Illustration, Cancel, Start**. **Cancel** is used to quit the game in welcome interface.

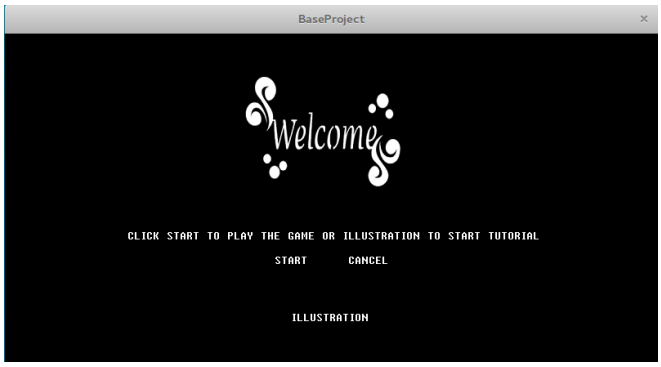


Illustration button will start one round as a tutorial with more reminders to tell users what to do. Hitting head of the piggy bank wouldn't cause addition of score but the coin will vanish. Only back of piggy bank triggers that.

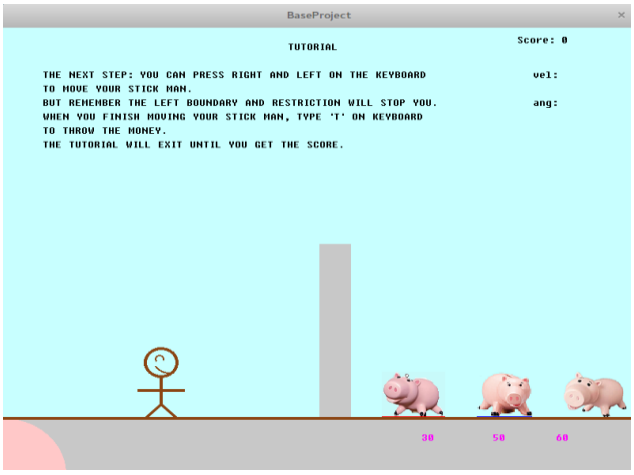
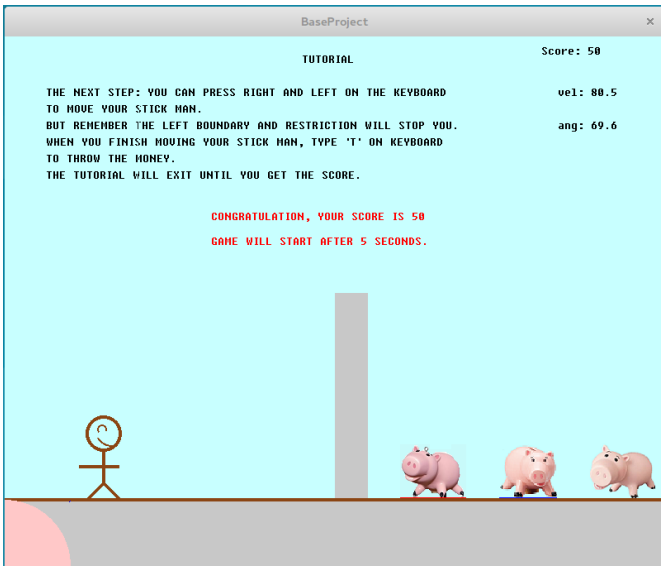
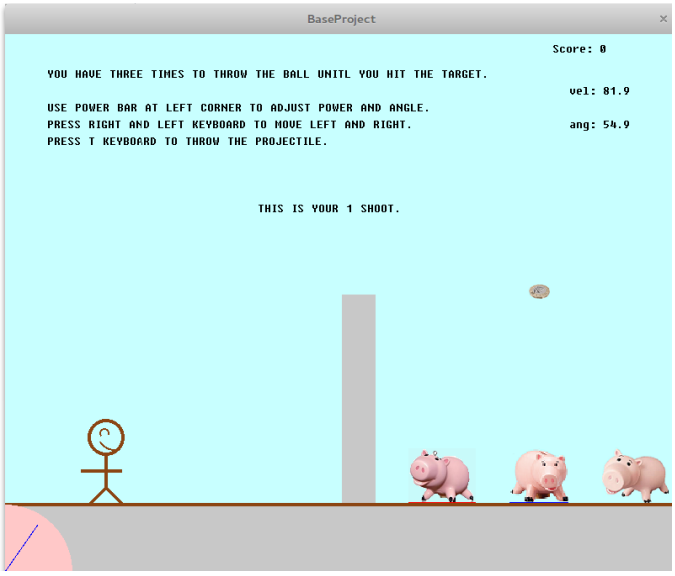


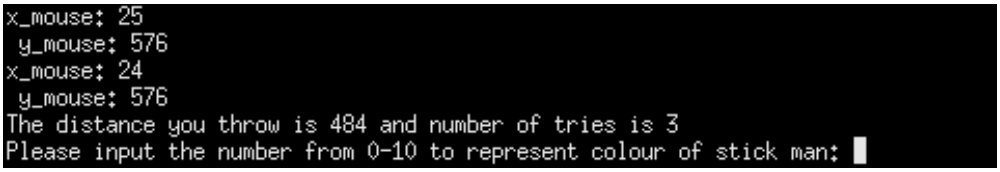
Illustration draws congratulation texts until users hit the target and exit in 5 seconds to game interface which is the same as the window after clicking **Start** in welcome window directly.



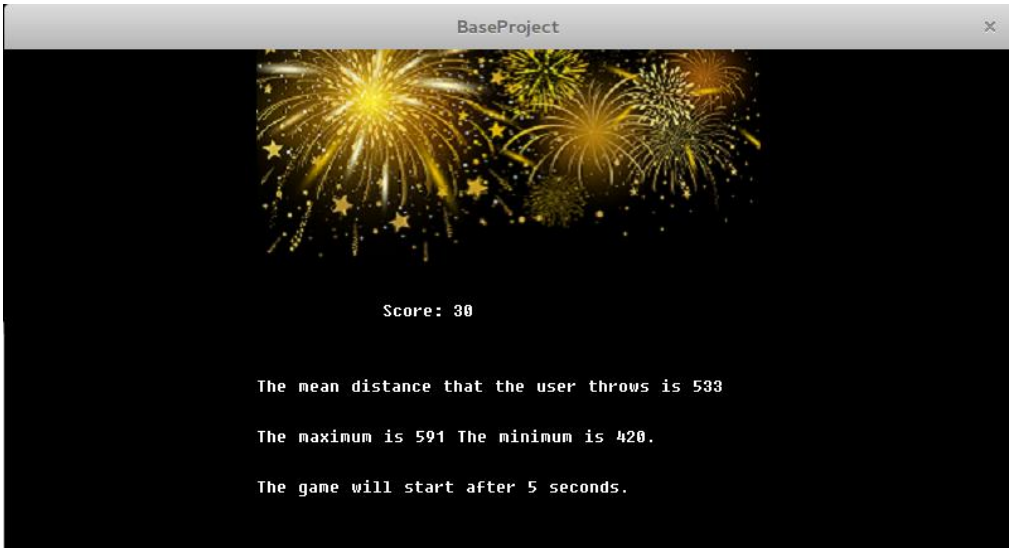
In the real game where users just have three opportunities to throw the coin, after adjusting position of stick man and typing ‘T’ on keyboard, power bar at bottom left corner is permitted to employ. Once left mouse button is clicked, the coin will appear and be thrown from right hand of the stick man. The targets represents scores which are 30, 50, 60 from left to right respectively.



What the terminal shows are the question of choosing colours, mouse position using power bar together with distance for current throwing, number of trying.



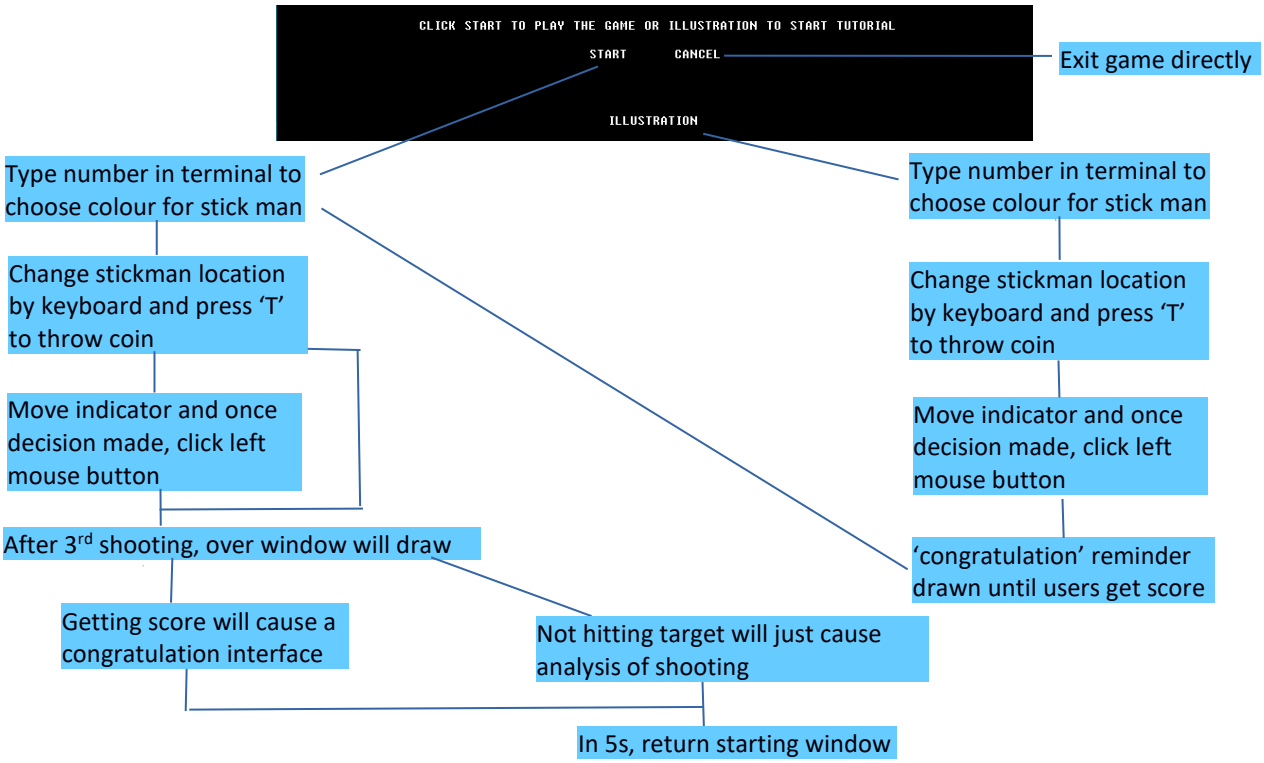
Having run out of 3 chance, the over window will pop up and give analysis of user’s throwing.



3.4 Extended features

In order to make the game more enjoyable, restriction has different height and target has different positions every time users start a new round. Animation is imported into the game as well. Coin follows an invisible projectile trace and correspondent score will be added only when coin hits back of piggy banks. Meanwhile, considering redundancy of exchanging between terminal and game window while determining initial velocity and angle, power bar is designed, which is only needs mouse control.

3.5 Testing (shown by flowchart)



As shown in flowchart above, in welcome window, users have three choices to test the program. If programmers want to test whether restriction and frame can stop throwing coin, choosing to click Illustration button in welcome window is suggested. In this window, users are able to try infinite times as they want as long as they don't hit the target. Otherwise, users can click Start in welcome window directly to test logic of throwing coin in three times, which is the main function of this program.

3.6 Customers satisfaction

The target audience of this graphics game should be bank customers who are not familiar or experienced with programming, but still would like to kill time when they are waiting for their personal bankers. Considering to amuse customers, the game is designed not so difficult to handle and achieve score. Besides, instead of restarting the game by running the program again after one round, the game will return to welcome window automatically. By clicking Start in welcome window, customers can start the game again.

4. Evaluation

4.1 Bugs and suggestions for improvement

Problems caused by lack of memory should be declared firstly. Due to a lot of arrays and pointers used in this game, there are some possibilities that the interface could flash into black except data updated by means

of the rectangle with background colour. To avoid this, moving mouse slowly is available to decrease probability of the bug. The memory problem should be developed when this game is really loaded into bank app. For now, if this happens when playing game, users can restart the game.

Acquiring score is only allowed when coin hits back of the piggy bank. Although precision accuracy has tested as high as possible, piggy banks probably are erased by projectile trace and won't be drawn again. In the future, there should be a solution to solve this in case that users do not know where to hit in second and third shooting because part of targets disappears.

4.2 Overall quality

In this graphics game, based on satisfying assignment requirements, some extended features are added in and reasonable logic coding as well as beautiful interface are considered. Knowledge learnt from all programming lab is used including functions, loops and pointers, arrays, which improves connection among different components of the program. Instead of resorting to python or other programming facility, this program is designed in CodeBlocks individually. To sum up, although there are still some bugs needed solving in the future, this game can run as designers thought and with a lot of functions this game can do, users should like it.

5. Reference

Library form: https://www.elec.york.ac.uk/internal_web/meng/yr1/modules/Introduction_to_Programming/Labs/documentation/graphics__lib_8c.html

6. Appendix (C Source Code)

```
/*
 *  A program to demonstrate simple graphical operations
 */

/*  This line allows the compiler to understand the
 *  graphics functions
 */
 #include <graphics_lib.h>
 #include <math.h>
 #define pi 3.14159265
 /* If the file uses the graphics library, it should be included into the src
folder */

/* Function prototypes */
void changecolour(int user_input1);
double Mean(double values[],int num_values);
double draw_projectile_path(double initial_pos_x, double initial_pos_y, int
*proj_x, int restriction_height, int x_target, int *proj_y);
void draw_target(int x_target);
int score_board(int x, int distance, int y);
void draw_stick_man(double x_position,double y_position);
void draw_restriction(int restriction_height);
double maximum(double array1[],int size);
double minimum(double array2[],int length);
void illustration(int x_position, int user_input1, int y_position,  int ini-
tial_pos_x, int initial_pos_y, int mark, int key, int proj_x, int proj_y, char
string_mark[11]);
void over_interface(int mark, double mean_distance, int max, int min, char
string_mark[11]);
double draw_power_bar(double *velocity);

/* Main function starts here */
int main(void)
{
    /* Declare variables for the x and y positions */
    double x_position;
    double y_position;

    /* Declare variable for choosing colour */
    int user_input1;

    /* Declare variables for initial position of stick man */
    double initial_pos_x;
    double initial_pos_y;

    /* Declare variable for pointer of keyboard */
    int key;

    /* Declare variable for left x-coordinate of blue target */
    int x_target;

    /* Declare variables to save maximum and minimum distance */
    int max;
    int min;

    /* Declare pointers to represent pos_x and pos_y from projectile function */
    int proj_x;
```

```
    int proj_y;

    /* Declare variable to save the score into and set its initial value */
    int mark = 0;

    /* Declare the variable to represent highest point of restriction */
    int restriction_height;

    /* Declare arraies to store letters */
    char string_mark[11];
    char string_count[25];

    /* Flag to symbol when we want to end */
    int count = 0;

    /* Flage to symbol how many shootings are tying */
    int num_distance = 0;

    /* Declare an array storing distance of 3 shooting */
    double data_values[3];

    /* Store the value from function of draw_projectile */
    double mean_distance;

    /* Declare variables of welcome window */
    /* Mouse postion where to click */
    int x_start;
    int y_start;

    /* Flage to symbol when to exit the welome window */
    int done = 0;

    /* Declare a variable to represent left mouse button */
    int button;

    /* Set initial position of the stick man */
    x_position = 80;
    y_position = 480;

    /* Set initial position of the hand */
    initial_pos_x = 110;
    initial_pos_y = 520;

    /* Open a 800 pixels wide by 640 pixels high graphics window */
    GFX_InitWindow(800, 640);

    /* Ensure to give a random number every time users start the game */
    srand(time(NULL));

    /* Set up event queue,keyboard and mouse */
    GFX_CreateEventQueue();
    GFX_InitFont();
    GFX_InitMouse();
    GFX_RegisterMouseEvents();

    /* Initialise Bitmap Images, which goes after GFX InitWindow but before any
other BITMAP functions */
    GFX_InitBitmap();

    /* Set the background colour */
    GFX_SetBackgroundColour(BLACK);
```

```
    /* Draw the start interface */
    /* Draw welcome bitmap */
    BITMAP image5 = GFX_LoadBitmap("welcome.png");
    GFX_DrawBitmap(image5,390,120);

    /* Set characters' colour */
    GFX_SetColour(WHITE);

    /* Draw welcome text*/
    GFX_DrawText(150,240,"CLICK START TO PLAY THE GAME OR ILLUSTRATION TO START
TUTORIAL");
    GFX_DrawText(330,270,"START");
    GFX_DrawText(420,270,"CANCEL");
    GFX_DrawText(350,340,"ILLUSTRATION");

        /* Make figures written before draw */
    GFX_UpdateDisplay();

    /* Set a while loop for starting the game */
    while(!done)
    {
        /* Wait for an event */
        GFX_WaitForEvent();

        /* Process mouse events */
        if (GFX_IsEventMouseButton())
        {
            /* Check if left mouse button click */
            GFX_GetMouseButton(&button);
            if (button == MOUSE_BUTTON_LEFT)
            {
                /* Get click coordinates and save the value of x and y */
                GFX_GetMouseCoordinates(&x_start,&y_start);

                /* When user clicks start */
                if (x_start > 320 && x_start < 380 && y_start > 270 && y_start <
290)
                {

                    done = 1;

                }

                /* When user clicks illustration */
                else if (x_start > 340 && x_start < 460 && y_start > 340 &&
y_start < 360)
                {
                    illustration(x_position, user_input1, y_position, ini-
tial_pos_x, initial_pos_y, mark, key, proj_x, proj_y, string_mark);

                    GFX_ClearWindow();

                    done = 1;
                }

                /* When user clicks cancel */
                else if (x_start > 410 && x_start < 470 && y_start > 270 &&
y_start < 290)
                {
                    GFX_CloseWindow();
```

```
                }
            }
        }
    }

    /* Release memory of bitmap and clear window to exit starting window */
    GFX_FreeBitmap(image5);
    GFX_ClearWindow();

    /* Random x coordinate for the target */
    x_target = GFX_RandNumber(560,611);

    /* Random height for restriction */
    restriction_height = GFX_RandNumber(250,520);


    /* Fill grey to the ground and the lightcyan to the background */
    GFX_DrawFilledRectangle(0,560,800,640,LIGHTGRAY);
    GFX_DrawFilledRectangle(0,0,800,560,LIGHTCYAN);

    /* Draw a quarter of the circle as power bar */
    GFX_DrawFilledCircle(0,640,80,LIGHTMAGENTA);

    /* Set a display queue */
    GFX_RegisterDisplayEvents();

    /* Writing the reminder on the window */
    GFX_InitFont();
    GFX_SetColour(BLACK);
    GFX_DrawText(50,40,"BACK TO THE TERMINAL TO SET THE COLOUR OF YOUR STICK
MAN.");
    GFX_DrawText(670,60,"vel:");
    GFX_DrawText(670,100,"ang:");
    GFX_DrawText(650,10,"Score: 0");

    /* Make figures written before draw */
    GFX_UpdateDisplay();

    /* make the colour selection above case insensitive. */
    printf("Please input the number from 0-10 to represent colour of stick man:
");
    scanf("%d",&user_input1);

    /* Draw a rectangle to hide th text before */
    GFX_DrawFilledRectangle(50,40,600,60,LIGHTCYAN);

    /* Draw the reminder on the top of the main interface */
    GFX_DrawText(50,40,"YOU HAVE THREE TIMES TO THROW THE BALL UNITL YOU HIT THE
TARGET.");
    GFX_DrawText(50,80,"USE POWER BAR AT LEFT CORNER TO ADJUST POWER AND ANGLE.
");
    GFX_DrawText(50,100,"PRESS RIGHT AND LEFT KEYBOARD TO MOVE LEFT AND RIGHT.
");
    GFX_DrawText(50,120,"PRESS T KEYBOARD TO THROW THE PROJECTILE. ");

    /* Set the colour and draw the stick man using the function */
    changeColour(user_input1);
    printf("\n");

    /* Display targets, restriction and stick man */
    draw_target(x_target);
```

```
    draw_restriction(restriction_height);
    draw_stick_man(x_position,y_position);

    /* Make figures written before draw */
    GFX_UpdateDisplay();

    /* Initialise keyboard */
    GFX_InitKeyboard();
    GFX_RegisterKeyboardEvents();

    /* Define 3 movements using do-while loop */
    do
    {
        /* Tell user how many shoots they did */
        sprintf(string_count, "THIS IS YOUR %d SHOOT.", count+1);
        GFX_SetColour(BLACK);
        GFX_DrawText(300, 200, string_count);

        /* Wait for an event */
        GFX_WaitForEvent();

        /* Move left */
        if (GFX_IsEventKeyDown())
        {

            GFX_GetKeyPress(&key);
            if (key == ALLEGRO_KEY_LEFT)
            {
                if (x_position > 50)
                {
                    /* Draw a rectangle to hide the previous stick man */
                    GFX_DrawFilledRectangle(x_position-40,y_position-30,x_posi-
tion+30,y_position+80,LIGHTCYAN);

                    /* Do the move of going left */
                    x_position -= 20;

                    /* Set the colour and draw the stick man again */
                    changeColour(user_input1);
                    draw_stick_man(x_position,y_position);

                    /* Refresh the coordinates of initial pos_x and pos_y */
                    initial_pos_x = x_position + 30;
                    initial_pos_y = y_position + 40;

                    /* Change the variable into string */
                    GFX_DrawFilledRectangle(700,10,800,30,LIGHTCYAN);

                    /* Change the variable into string */
                    GFX_SetColour(BLACK);
                    sprintf(string_mark, "Score: %d", mark);
                    GFX_DrawText(650,10,string_mark);

                    /* Make figures written before draw */
                    GFX_UpdateDisplay();
                }
            }

        /* Move right */
           else if (key == ALLEGRO_KEY_RIGHT)
            {
```

```
                if (x_position < 360)
                {
                    /* Draw a rectangle to hide the previous stick man */
                    GFX_DrawFilledRectangle(x_position-40,y_position-30,x_posi-
tion+30,y_position+80,LIGHTCYAN);

                    /* Do the move of going right */
                    x_position +=  20;

                    /* Set the colour and draw the stick man again */
                    changeColour(user_input1);
                    draw_stick_man(x_position,y_position);

                    /* Refresh the coordinates of initial pos_x and pos_y */
                    initial_pos_x = x_position + 30;
                    initial_pos_y = y_position + 40;

                    /* Change the variable into string */
                    GFX_DrawFilledRectangle(700,10,800,30,LIGHTCYAN);

                    GFX_SetColour(BLACK);
                    sprintf(string_mark, "Score: %d", mark);
                    GFX_DrawText(650,10,string_mark);

                    /* Make figures written before draw */
                    GFX_UpdateDisplay();
                }
            }

        /* Throw coin */

           else if (key == ALLEGRO_KEY_T)
            {
                /* Draw the projectile from the hands */
                changeColour(user_input1);
                data_values[num_distance] = draw_projectile_path(initial_pos_x,
initial_pos_y, &proj_x, restriction_height, x_target, &proj_y);

                /* Get the data into the array storing it into the position of
num_distance */
                printf("The distance you throw is %.01f and number of tries is
%d\n",data_values[num_distance], num_distance + 1);

                /* Calculate the score user gets */
                mark += score_board(x_target,proj_x,proj_y);

                /* Draw the filled rectangle to hide the previous mark and time
of shoot */
                GFX_DrawFilledRectangle(700,10,800,30, LIGHTCYAN);
                GFX_DrawFilledRectangle(300,200,500,220,LIGHTCYAN);

                GFX_SetColour(BLACK);

                /* Change the variable into string */
                sprintf(string_mark, "Score: %d", mark);
                GFX_DrawText(650,10, string_mark);

                /* Redraw the part hidded by the projectile */
                GFX_DrawFilledRectangle(400,restriction_height,440,558,LIGHT-
GRAY);
```

```
                /* Redraw all elements which can be erased */
                GFX_DrawText(50,40,"YOU HAVE THREE TIMES TO THROW THE BALL UNITL
YOU HIT THE TARGET.");
                GFX_DrawText(50,80,"USE POWER BAR AT LEFT CORNER TO ADJUST POWER
AND ANGLE. ");
                GFX_DrawText(50,100,"PRESS RIGHT AND LEFT KEYBOARD TO MOVE LEFT
AND RIGHT. ");
                GFX_DrawText(50,120,"PRESS T KEYBOARD TO THROW THE PROJECTILE.
");
                GFX_DrawFilledRectangle(0,560,800,640,LIGHTGRAY);
                GFX_DrawFilledCircle(0,640,80,LIGHTMAGENTA);

                /* Make figures written before draw */
                GFX_UpdateDisplay();

                /* Update the value in the array */
                num_distance += 1;

                /* Calculate the mean distance that the user throws */
                mean_distance = Mean(data_values,num_distance);

                /* Calculate the minimum and the maximum */
                min = minimum(data_values,num_distance);
                max = maximum(data_values,num_distance);

                /* Count how many times users tried */
                count += 1;

            }

        }

    /* Leave the loop in 3 trying */
    } while (count < 3);

    /* Wait for 3 second to print out the records */
    GFX_PauseFor(3000);

    /* Draw the over_interface and restart it */
    over_interface(mark, mean_distance, max, min, string_mark);

    return 0;
}

/* Set a change colour function */
void changeColour(int user_input1)
{
    switch(user_input1)
    {
    // Case is used to select the colour
    case 0:
        GFX_SetColour(RED);
        break;
    case 1:
        GFX_SetColour(LIGHTBLUE);
        break;
    case 2:
        GFX_SetColour(LIGHTGREEN);
        break;
    case 3:
        GFX_SetColour(BLUE);
```

```
            break;
        case 4:
            GFX_SetColour(GREEN);
            break;
        case 5:
            GFX_SetColour(BROWN);
            break;
        case 6:
            GFX_SetColour(LIGHTMAGENTA);
            break;
        case 7:
            GFX_SetColour(DARKGRAY);
            break;
        case 8:
            GFX_SetColour(WHITE);
            break;
        case 9:
            GFX_SetColour(CYAN);
            break;
        case 10:
            GFX_SetColour(MAGENTA);
            break;
        default:
            GFX_SetColour(BLACK);
        }

}


/* Function of drawing the projectile */
double draw_projectile_path(double initial_pos_x, double initial_pos_y, int
*proj_x, int restriction_height, int x_target, int *proj_y)
{
    /* Declare variables for velocities of x,y and initial velocity, angle and
final position and gravity */
    double vel_x,vel_y,initial_angle,initial_velocity,pos_x,pos_y,fi-
nal_x,pre_pos_x;
    float n = 9.81;
    float time;
    int gravity;
    double velocity;

    /* Insert a bitmap following the trace of projectile */
    BITMAP image4 = GFX_LoadBitmap("coin.png");

    /* Move the line on the power bar to decide the power and the angle */
    initial_angle = draw_power_bar(&velocity);
    initial_velocity = velocity;

    /* Set the initial position for the projectile and gravity */
    pos_x = initial_pos_x + 2;
    pos_y = initial_pos_y;

    /* Set the value for previous pos_x */
    pre_pos_x = pos_x;

    /* Give initial value of pos_y */
    pos_y = 0;

    /* Change type of gravity into integer */
    gravity = (int) n;
```

```
    /* Using math to separate velocity to x and y direction */
    vel_x = initial_velocity * cos(initial_angle/180 * 3.1415926);
    vel_y = initial_velocity * sin(initial_angle/180 * 3.1415926);

    /* Move to initial point of the projectile */
    GFX_MoveTo(initial_pos_x, initial_pos_y);

    /* Set a while-loop to draw invisible projectile */
    while (pos_x <= 800 && pos_y <= 560)
    {
        /* Function to draw the projectile */
        /* Draw a filled rectangle to hide previous coin */
        GFX_DrawFilledRectangle(pre_pos_x-13,pos_y-
9,pre_pos_x+13,pos_y+12,LIGHTCYAN);
        GFX_SetColour(LIGHTCYAN);

        /* Calculate the coordinates of projectile and bitmap */
        time = (pos_x - initial_pos_x) / vel_x;
        pos_y = initial_pos_y - (vel_y * time) + (gravity * time * time)/2;
        GFX_DrawLineTo(pos_x,pos_y,1);
        GFX_DrawBitmap(image4,pos_x,pos_y);

        /* Make figures written before draw */
        GFX_UpdateDisplay();

        /* Give value to previous pos_x */
        pre_pos_x = pos_x;
        pos_x += 2;

        /* If the coordinates of projectile are in the range of restriction,
stop throwing */
        if (pos_x+10 >= 400 && pos_x-10 <= 440 && pos_y+6 >= restriction_height
&& pos_y <= 560)
        {
            break;
        }

        /* If the projectile hits the frame, stop throwing */
        else if (pos_y == 0 || pos_x == 800)
        {
            break;
        }

        /* If the projectile hits the blue target, stop throwing */
        else if (pos_x+10 >= x_target && pos_x+10 <= x_target+70 && pos_y+9 >=
490 && pos_y+9 <= 560)
        {
            break;
        }

        /* If the projectile hits the red target, stop throwing */
        else if (pos_x+10 >= x_target-120 && pos_x+10 <= x_target-40 && pos_y+9
>= 490 && pos_y+9 <= 560)
        {
            break;
        }

        /* If the projectile hits the green target, stop throwing */
        else if (pos_x+10 >= x_target+110 && pos_x+10 <= x_target+189 && pos_y+9
>= 490 && pos_y+9 <= 560)
```

```
        {
            break;
        }



    }

    /* Calculate the final x position of the projectile */
    final_x = vel_x * time;

    /* use the final_x and pos_x, pos_y into the main function */
    *proj_x = pos_x;
    *proj_y = pos_y;

    /* Clear the final coin */
    GFX_DrawFilledRectangle(pre_pos_x-13,pos_y-9,pre_pos_x+13,pos_y+12,LIGHT-
CYAN);

    /* Return final x-coordinate of coin used in main function */
    return final_x;

    /* Manually free the memory used by the image once done with it */
    getchar();
    GFX_FreeBitmap(image4);

    /* Make figures written before draw */
    GFX_UpdateDisplay();
}

/* Function of drawing stick man */
void draw_stick_man(double x_position, double y_position)
{
    /* Draw a circle at x_position, y_position with radius 20 and line thickness
2 */
    GFX_DrawCircle(x_position, y_position, 20, 4);

    /* Draw the ground */
    GFX_DrawLine(0,560,800,560,4);

    /* Draw body of the stick man which are four lines */
    /* Draw lines according to the coordinate of the circle*/
    GFX_DrawLine(x_position-30,y_position+40,x_position+30,y_position+40,4);
    GFX_DrawLine(x_position,y_position+20,x_position,y_position+60,4);
    GFX_DrawLine(x_position,y_position+60,x_position-20,y_position+80,4);
    GFX_DrawLine(x_position,y_position+60,x_position+20,y_position+80,4);

    /* Draw face of the stick man */
    GFX_DrawArc(x_position-2,y_position-5,5,500,240,2);
    GFX_DrawArc(x_position+10,y_position-5,20,70,80,2);

    /* move the contents of the screen buffer to the display */
    GFX_UpdateDisplay();

}

/* Function of drawing target */
void draw_target(int x_target)
{
    /* Draw three targets randomly on the ground representing different score */
    GFX_DrawFilledRectangle(x_target,550,x_target+70,560,BLUE);
```

```
    GFX_DrawFilledRectangle(x_target-120,550,x_target-40,560,RED);
    GFX_DrawFilledRectangle(x_target+110,550,x_target+189,560,GREEN);

    /* Load the image into a variable of type BITMAP to represent blue target */
    BITMAP image1 = GFX_LoadBitmap("blue_target.png");
    BITMAP image2 = GFX_LoadBitmap("green_target.png");
    BITMAP image3 = GFX_LoadBitmap("red_target.png");

    /* Draw the image to the window, the x,y position is the centre of the im-
age. */
    /* blue_target */
    GFX_DrawBitmap(image1,x_target+35,525);

    /* green_target */
    GFX_DrawBitmap(image2,x_target+149,525);

    /* red target */
    GFX_DrawBitmap(image3,x_target-80,525);

    /* Make figures written before draw */
    GFX_UpdateDisplay();

    /* Manually free the memory used by the image once done with it */
    getchar();
    GFX_FreeBitmap(image1);
    GFX_FreeBitmap(image2);
    GFX_FreeBitmap(image3);

    /* Make figures written before draw */
    GFX_UpdateDisplay();


}

/* Function of calculating mean distance */
double Mean(double values[],int num_values)
{
    /* Declare variables */
    double average;
    int current_value;

    /* Initialise the average to zero*/
    average = 0;

    /* Calculate the sum of all of the values */
    for (current_value = 0; current_value < num_values; current_value ++)
    {
        average += values[current_value];
    }

    /* Divide by the number of values, if there are any */
    if (num_values > 0)
    {
        average /= current_value;
    }

    /* Return average and used in main funciton */
    return average;
    }

/* Function to symbolize score users get hitting different targets */
```

```
int score_board(int x, int distance, int y)
{

    /* When the projectile is hitting the blue target */
    if (distance+10 >= x && distance <= x+50 && y+9 >= 490 && y+9 <= 530)
    {
        return 50;
    }

    /* When the projectile is hitting the red target */
    else if (distance+10 >= x-60 && distance <= x-20 && y+9 >= 490 && y+9 <=
530)
    {
        return 30;
    }

    /* When the projectile is hitting the green target */
    else if (distance+10 >= x+120 && distance <= x+150 && y+9 >= 490 && y+9 <=
530)
    {
        return 60;
    }

    /* When the user didn't hit the target */
    else
    {
        return 0;
    }


}

/* Function for drawing restriction */
void draw_restriction(int restriction_height)
{
    /* Draw the restriction to restrict stick_man position before this */
    GFX_DrawFilledRectangle(400,restriction_height,440,558,LIGHTGRAY);

    /* Make restriction draw */
    GFX_UpdateDisplay();

}

/* Function for defining how power bar is working */
double draw_power_bar(double *velocity)
{
    /* Declare the coordinates of mouse position */
    int x_mouse;
    int y_mouse;
    double y_distance;
    double x_distance;

    /* Declare line length and angle between line and x-axial */
    float length;
    float angle;
    float initial_velocity;

    /* val is used to change radius into degree */
    int val = 180.0 / pi;

    /* Declare arrays to store letters of velocity and angle */
```

```
    char string_vel[15];
    char string_ang[10];

    /* Initialise the mouse and turn on mouse events. */
    GFX_InitMouse();
    GFX_RegisterMouseEvents();

    /* Initialise the font so you can draw text */
    GFX_InitFont();

    /* While loop will recognize coordinates of mouse and draw a line */
    while(1)
       {
        /* Pause the program until an event occurs */
        GFX_WaitForEvent();

        /* Once an event has occurred, check whether to move the mouse */
        if (GFX_IsEventMouseMoved())
        {
            /* Get the position of the mouse, store the values in x_mouse1 and
y_mouse1 */
            GFX_GetMouseCoordinates(&x_mouse, &y_mouse);

            /* If the line is in the power bar, draw the indicator */
            if (x_mouse >= 1 && x_mouse <= 81 && y_mouse >= 561 && y_mouse <=
640)
            {
                /* Clear previous line */
                GFX_DrawFilledRectangle(0,563,80,640,LIGHTGRAY);
                GFX_DrawFilledCircle(0,640,80,LIGHTMAGENTA);

                /* CLear previous velocity and angle */
                GFX_DrawFilledRectangle(670,60,750,140,LIGHTCYAN);

                /* Make figures written before draw */
                GFX_UpdateDisplay();

                /* Display the mouse position in terminal*/
                printf("x_mouse: %d \n y_mouse: %d \n", x_mouse, y_mouse);

                /* Calculate angle between the line and the frame */
                y_distance = 640 - y_mouse;
                x_distance = x_mouse;
                angle = val * atan2(y_distance,x_distance);

                /* Calculate length between original point and mouse location */
                length = sqrt((x_mouse)*(x_mouse) + (640 - y_mouse)*(640 -
y_mouse));

                /* Change the length into the initial velocity the velocity is
in range of 0 - 135.8 */
                initial_velocity = 1.2 * length;

                /* Display the velocity and angle on the main interface */
                GFX_SetColour(BLACK);

                /* Change variables into string */
                sprintf(string_vel,"vel: %.1f",initial_velocity);
                sprintf(string_ang,"ang: %.1f",angle);

                /* Display velocity and angle */
```

```
                GFX_DrawText(670,60,string_vel);
                GFX_DrawText(670,100, string_ang);

                /* Draw line corresponding to coordinates of mouse */
                GFX_SetColour(BLUE);
                GFX_DrawLine(0,640,x_mouse, y_mouse,1);

                /* Display velocity and angle decided by users on screen */
                GFX_UpdateDisplay();

            }
        }

        /* If event is not moved, check if event is mouse clicked. If so, break
*/
        else if (GFX_IsEventMouseButton())
        {
            break;
        }
    }

    /* Use these two values to define the projectile in draw_projectile function
*/
    *velocity = initial_velocity;
    return angle;
}

/* Function to calculate maximum distance in 3 shooting */
double maximum(double array1[], int size)
{
    /* Declare variables */
    int current_values;
    double largest;

    /* Store the largest value of this array */
    for (current_values = 0; current_values < size; current_values ++)
    {
        if (largest < array1[current_values])
            largest = array1[current_values];
    }

    return largest;
}

/* Function to calculate minimum distance in 3 shooting */
double minimum(double array2[], int length)
{
    /* Declare variables */
    int current_values;
    double smallest;

    /* Store the smallest value of this array */
    for (current_values = 0; current_values < length; current_values ++)
    {
        if (smallest > array2[current_values])
            smallest = array2[current_values];
    }

    return smallest;
}
```

```
/* Function to define the starting interface */
void illustration(int x_position, int user_input1, int y_position,  int ini-
tial_pos_x, int initial_pos_y, int mark, int key, int proj_x, int proj_y, char
string_mark[11])
{
    /* Declare variable in this function */
    char string_hit[40];
    double velocity;
    int x_target;
    int restriction_height;

    /* Check if the projectile to hit the target */
    int done = 0;

    /* Set the random position for x position of the target */
    x_target = GFX_RandNumber(560,611);


    /* Random height for restriction */
    restriction_height = GFX_RandNumber(250,520);

    /* Display the first step */
    /* Fill grey to the ground and the cyan to the background */
    GFX_DrawFilledRectangle(0,560,800,640,LIGHTGRAY);
    GFX_DrawFilledRectangle(0,0,800,560,LIGHTCYAN);

    /* Draw a quarter of the circle to decide the angle of projectile */
    GFX_DrawFilledCircle(0,640,80,LIGHTMAGENTA);

    /* Writing the reminder on the window */
    GFX_RegisterDisplayEvents();
    GFX_InitFont();
    GFX_SetColour(BLACK);
    GFX_DrawText(360,20,"TUTORIAL");
    GFX_DrawText(50,60,"FIRSTLY, BACK TO TERMINAL TO SET THE COLOUR OF STICK
MAN.");
    GFX_DrawText(670,60,"vel:");
    GFX_DrawText(670,100,"ang:");
    GFX_DrawText(650,10,"Score: 0");

    /* Make figures written before draw */
    GFX_UpdateDisplay();

    /* make the colour selection above case insensitive. */
    printf("Please input the number from 0-10 to represent colour of stick man:
");
    scanf("%d",&user_input1);


    /* Display the second step */
    /* Draw a rectangle to hide the reminder for first step */
    GFX_DrawFilledRectangle(50,60,600,80,LIGHTCYAN);

    /* Draw the reminder on the top of the main interface */
    GFX_SetColour(BLACK);
    GFX_DrawText(50,60,"THE NEXT STEP: YOU CAN PRESS RIGHT AND LEFT ON THE KEY-
BOARD  ");
    GFX_DrawText(50,80,"TO MOVE YOUR STICK MAN.");
    GFX_DrawText(50,100,"BUT REMEMBER THE LEFT BOUNDARY AND RESTRICTION WILL
STOP YOU.");
```

```
    GFX_DrawText(50,120,"WHEN YOU FINISH MOVING YOUR STICK MAN, TYPE 'T' ON KEY-
BOARD ");
    GFX_DrawText(50,140,"TO THROW THE MONEY.");
    GFX_DrawText(50,160,"THE TUTORIAL WILL EXIT UNTIL YOU GET THE SCORE.");

    /* Draw the stick man using the function */
    changeColour(user_input1);

    /* Draw the target using the function */
    draw_target(x_target);

    /* Draw components of the main interface */
    draw_restriction(restriction_height);
    draw_stick_man(x_position,y_position);

    /* Show how much score for different target */
    GFX_SetColour(MAGENTA);
    GFX_DrawText(x_target+20,580,"50");
    GFX_DrawText(x_target-70,580,"30");
    GFX_DrawText(x_target+100,580,"60");

    /* Make figures written before draw */
    GFX_UpdateDisplay();

    /* Set up event queue, keyboard */
    GFX_InitKeyboard();
    GFX_RegisterKeyboardEvents();

    /* Start a do-while loop to define 3 movements */
    do
    {
        /* Wait for an event */
        GFX_WaitForEvent();

        if (GFX_IsEventKeyDown())
        {

            GFX_GetKeyPress(&key);

            /* Move left */
            if (key == ALLEGRO_KEY_LEFT)
            {
                if (x_position > 50)
                {
                    /* Draw rectangle to hide the previous stick man */
                    GFX_DrawFilledRectangle(x_position-40,y_position-30,x_posi-
tion+30,y_position+80,LIGHTCYAN);

                    /* Do the move of going left */
                    x_position -= 20;

                    /* Set the colour and draw the stick man again */
                    changeColour(user_input1);
                    draw_stick_man(x_position,y_position);
                    initial_pos_x = x_position + 30;
                    initial_pos_y = y_position + 40;

                    /* Draw the filled rectangle to hide the previous mark */
                    GFX_DrawFilledRectangle(700,10,800,30,LIGHTCYAN);

                    /* Change the variable into string */
```

```
                    GFX_SetColour(BLACK);
                    sprintf(string_mark, "Score: %d", mark);
                    GFX_DrawText(650,10,string_mark);

                    /* move the stick man to redisplay */
                    GFX_UpdateDisplay();
                }
            }
            /* Move right */

            else if (key == ALLEGRO_KEY_RIGHT)
            {
                if (x_position < 360)
                {
                    /* Draw a rectangle to hide the previous stick man */
                    GFX_DrawFilledRectangle(x_position-40,y_position-30,x_posi-
tion+30,y_position+80,LIGHTCYAN);

                    /* Do the move of going right */
                    x_position +=  20;

                    /* Set the colour and draw the stick man again */
                    changeColour(user_input1);
                    draw_stick_man(x_position,y_position);
                    initial_pos_x = x_position + 30;
                    initial_pos_y = y_position + 40;

                    /* Draw the filled rectangle to hide the previous mark */
                    GFX_DrawFilledRectangle(700,10,800,30,LIGHTCYAN);

                    /* Change the variable into string */
                    GFX_SetColour(BLACK);
                    sprintf(string_mark, "Score: %d", mark);
                    GFX_DrawText(650,10,string_mark);

                    /* move the stick man to redisplay */
                    GFX_UpdateDisplay();
                }
            }

            /* Throw coin */

            else if (key == ALLEGRO_KEY_T)
            {
                /* Draw the projectile from the hands */
                changeColour(user_input1);
                draw_projectile_path(initial_pos_x, initial_pos_y, &proj_x, re-
striction_height, x_target, &proj_y);

                /* Calculate the score user gets */
                mark += score_board(x_target,proj_x,proj_y);

                /* Change the variable into string */
                GFX_DrawFilledRectangle(700,10,800,30,LIGHTCYAN);

                /* Change the variable into string */
                GFX_SetColour(BLACK);
                sprintf(string_mark, "Score: %d", mark);
                GFX_DrawText(650,10,string_mark);

                /* Redraw a quarter of the circle */
```

```
                GFX_DrawFilledCircle(0,640,80,LIGHTMAGENTA);

                /* move projectile path to the display */
                GFX_UpdateDisplay();

                /* Check if the projectile hits the target */
                /* If user didn't get mark, keep shooting */
                if (mark == 0)
                {
                    /* Redraw elements which have possibilities to be erased */
                    GFX_SetColour(BLACK);
                    GFX_DrawText(50,60,"THE NEXT STEP: YOU CAN PRESS RIGHT AND
LEFT ON THE KEYBOARD  ");
                    GFX_DrawText(50,80,"TO MOVE YOUR STICK MAN.");
                    GFX_DrawText(50,100,"BUT REMEMBER THE LEFT BOUNDARY AND RE-
STRICTION WILL STOP YOU.");
                    GFX_DrawText(50,120,"WHEN YOU FINISH MOVING YOUR STICK MAN,
TYPE 'T' ON KEYBOARD ");
                    GFX_DrawText(50,140,"TO THROW THE MONEY.");
                    GFX_DrawText(50,160,"THE TUTORIAL WILL EXIT UNTIL YOU GET
THE SCORE.");
                    draw_restriction(restriction_height);
                    GFX_DrawFilledRectangle(0,560,800,640,LIGHTGRAY);
                    GFX_DrawFilledCircle(0,640,80,LIGHTMAGENTA);

                    GFX_UpdateDisplay();


                    done = 0;

                }

                /* If user gets mark, stop the turorial and tell the user mark
*/
                else if (mark != 0)
                {
                    /* Change the variable into string */
                    GFX_SetColour(RED);

                    sprintf(string_hit, "CONGRATULATION, YOUR SCORE IS %d",
mark);
                    GFX_DrawText(250,210,string_hit);
                    GFX_DrawText(250,240,"GAME WILL START AFTER 5 SECONDS.");

                    GFX_UpdateDisplay();

                    done = 1;

                }

            }
        }

    /* Leave the loop when the user hits the target */
    } while(done == 0);

    /*Wait for 5 seconds to exit tutorial */
    GFX_PauseFor(5000);
}

/* Function for drawing the over interface */
```

```
void over_interface(int mark, double mean_distance, int max, int min, char
string_mark[11])
{
    /* Declare arrays used to display words in game window */
    char string_mean_distance[50];
    char string_max_min[40];

    /* Clear all the contents first */
    GFX_ClearWindow();

    /* Set characters' colour */
    GFX_SetColour(WHITE);

    /* Draw the over interface */
    /* If the user didn't get any mark */
    if (mark == 0)
    {
         /* Draw the mean distance */
        sprintf(string_mean_distance, "The mean distance that the user throws is
%.0lf",mean_distance);
        GFX_DrawText(300,240,string_mean_distance);

        /* Draw the minimum and the maximum */
        sprintf(string_max_min,"The maximum is %d The minimum is %d.",max,min);
        GFX_DrawText(300,380,string_max_min);

        /* Draw reminder telling users that they didn't hit the target */
        GFX_DrawText(300,200,"Sorry you didn't hit the target.");

        /* Wait for 5s to restart the game again */
        GFX_DrawText(300,340,"The game will restart after 5 seconds.");
        GFX_UpdateDisplay();
        GFX_PauseFor(5000);
        GFX_CloseWindow();
        main();
    }

    /* If the user gets marks */
    else
    {
        /* Draw firework to congratulate users to get score */
        BITMAP image6 = GFX_LoadBitmap("congratulation.png");
        GFX_DrawBitmap(image6,400,80);

        /* Draw the score */
        /* Change the variable into string */
        sprintf(string_mark, "Score: %d", mark);
        GFX_DrawText(300,200, string_mark);


        /* Draw the mean distance */
        sprintf(string_mean_distance, "The mean distance that the user throws is
%.0lf",mean_distance);
        GFX_DrawText(200,260,string_mean_distance);

        /* Draw the minimum and the maximum */
        sprintf(string_max_min,"The maximum is %d The minimum is %d.",max,min);
        GFX_DrawText(200,300,string_max_min);

        /* Wait for 5s to restart the game again */
        GFX_DrawText(200,340,"The game will start after 5 seconds.");
```

```
        GFX_UpdateDisplay();

        GFX_PauseFor(5000);
        GFX_CloseWindow();

        /* Release memory of bitmap */
        GFX_FreeBitmap(image6);

        /* Restart the game again from welcome window */
        main();

    }
}
```