|  |
| --- |
|  |

|  |
| --- |
|  |
| Projectile Bank Game Report |
| Exam No: Y3856186 |

**ABSTRACT:**

This report contains the methods used to reach the desired outcome of creating the Projectile game. The criteria given was to create a program that uses angle and velocity manipulation to draw a projectile path and implement it inside a game. As well as this a preferred outcome was to have the program related to the user, hence piggy banks and coins. The report summaries how by investigating real life physics it can be transformed into a computerised code. There are also other synopses which describe various features of the final project as well as evaluating the completion of the project.

ABSTRACT

PROBLEM ANALYSIS

SPECIFICATION

EVALUATION

**PROBLEM ANALYSIS:**

1.Description:

The game consists of four different screens. These are:

* Main Menu
* Actual Game
* Game-over
* Win/Completion screen

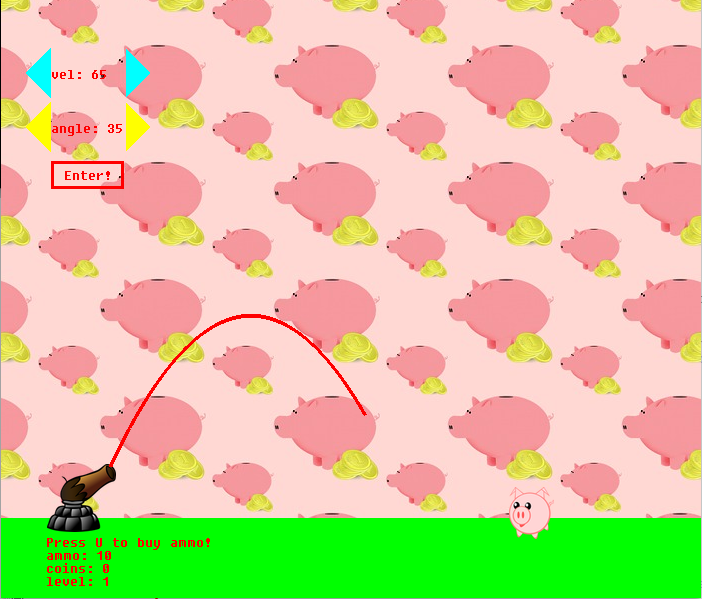
The objective is to aim your cannon using velocity and angle manipulation at the piggy bank. When the piggy bank is hit you receive an amount of coins for ammunition as well as this you level up (or advance to the next harder level). As you progress the game becomes more challenging as you will face a wall to prevent easier aiming. If the player runs out of ammunition they will be directed to a game-over screen where they have the chance to replay from level one. However, if the player completes all the levels (maximum level is two) they will be redirected to a congratulating screen as they have finished the game.

Figure 2:

Figure 1:

2.Laws of physics:

To implement the laws of physics, different variables needed to be taken into account. Such as time, the initial X and Y positions, the angle of the projectile, its velocity and most importantly gravity. Gravity is the most important variable as any object in order to be a projectile must be under the force of gravity. As shown in the free body diagram of a projectile below.

Figure 3:

The second most important variables are the velocity and the angles. By using the initial resultant velocity in calculations through trigonometry the X and Y velocity components are formed, which in return are used to give the height, distance and time of the volley to reach the ground. These work in relation with gravity as the object rises and falls, as shown later. [Ref: A]

Figure 4:

Initial V

X Velocity

Y Velocity

U

Therefore:

**Both the X and Y velocities are initial.**

The following maths will be used to create the parabolic path of the projectile:

For working out the positions of the projectile, air resistance is ignored to assume that the horizontal velocity stays constant. So, distance can be calculated from:

Where:

– new position

– initial position

– distance calculated

This leads to the following equation:

– Distance

– Velocity

– Time

As shown by the free body diagram of a projectile the only force applied to it would be gravity. Therefore: . Using Newton’s 2nd law of motion the Y position or height can be calculated through the equation:

Where:

– new position

y – initial position

– gravity

– time

Note: First equation with X is used to calculate horizontal distance whilst Y represents height.

[Ref: B]

3.Limitations:

For such a design to be added to the code one assumption had to be made. This is:

* To assume zero air resistance

Air resistance varies with the speed and size of the projectile, it increases until it is equal to the force of the object thrown, this is according to Newton’s third law of motion as each force has an equal and opposite force. As well as this in most projectiles (especially small ones e.g. cannonballs) air resistance is negligible, whilst it would make the total distance of the volley decrease, that would be by an insignificant amount. Therefore, it is assumed to be 0.

**SPECIFICATION:**

1.The Code:

The code has been split into two sectors. The Initialising section and the Game section. The Initialising section contains the overall setup of the game where different variables are created as well as calling functions for later use in the actual gameplay. The Game section contains the main game code where the main features are used such as angle and velocity control as well as accessing the game-over and WIN screen.

**INITIALISING:**

The structure of this section is as follows:

|  |  |  |
| --- | --- | --- |
| **SUBSECTION** | **DESCRPTION** | **EXAMPLE** |
| LOAD FUNCTION  (GFX\_InitX) | Functions are called to produce a window and initialising usage of different features such typing arrays/strings on the screen and using the keyboard. | GFX\_INIT:  WINDOW  BITMAPS  KEYBOARD FONT |
| DEFINE VALUES  (INT, CHAR, FLOAT)  (STORE DATA) | By defining numerous components, calculations and different features are enabled such as storing numbers and strings into array for typing/scripting on the screen. | int n1  float g = 9.81(ms^-2)  float n2  char ar1[n]  char ar2[n]  storeDATA(ar1,”text”,n1) |
| MAIN MENU  (GFX\_DRAW/KEYBOARD/PICTURE) | Created by drawing over the screen and using the buttons to direct the player either to the game or exiting the window. | GFX\_:  Draw background  Draw lines  Draw Text  Press Enter = Start Game  Update Display |
| NESTED FUNCTIONS  (e.g. SCREEN REFRESH) | By having functions in functions space is saved and increases efficiency. | Void ScreenRef():  {desired- functions}  ScreenRef() |

Table 1:

FIGURE 2: NESTED FUNCTION:

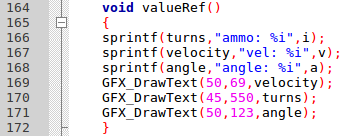
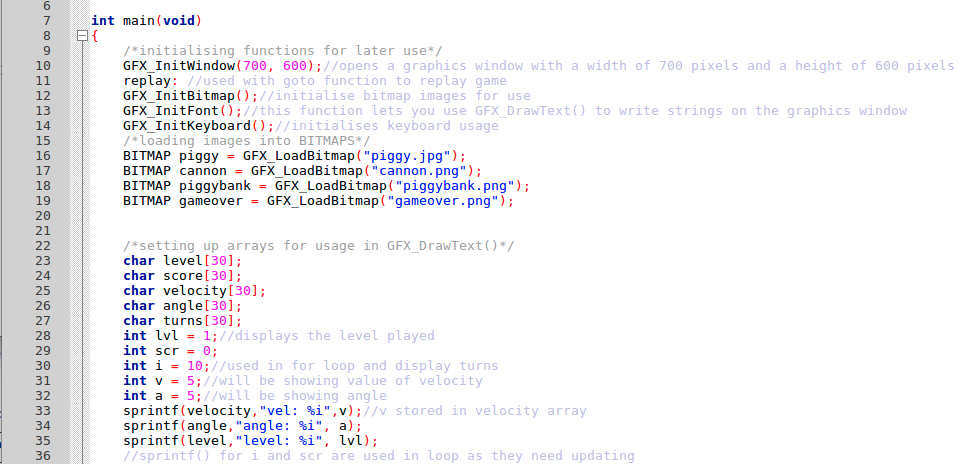


FIGURE 1: INITIALISING FUNCTIONS AND DEFINING VALUES:



**GAME:**

The structure of this section is as follows:

Table 2:

|  |  |  |
| --- | --- | --- |
| **SUBSECTION** | **DESCRPTION** | **EXAMPLE** |
| KEYBOARD EVENTS | By pressing different buttons, values of angles or velocity change or instead fire the projectile. | GFX\_:  Wait for event  Record Keyboard Press  If int press = wanted Key:  (Then do something) |
| PROJECTILE PATH | Depending on the angle and velocity a projectile path is drawn. | Initial V = B and angle U = A  So:  Y Velocity = Initial V x sin(u)  X Velocity = Initial X x cos(u)  While (Y-POS<ground)  {calculate time  Calculate Y-POS  X-POS add 2.5  GFX\_DrawLine  Update Display} |
| LEVEL UP | If the volley coordinates are within the range of that of the target then the player advances to level two. | If X-POS is between X1 and X2 and between Y1 and Y2:  {level add 1  GFX\_Draw[OBSTACLE]()  Update display} |
| AMMUNITION | One volley decreases the ammunition by a value of 1 until it reaches 0. | Ammo = 10  If int press = enter:  {ammo substract 1}  If int press = key U:  {ammo add 1} |
| GAME-OVER/WIN | This is dependent upon whether all levels have been cleared or whether ammunition value reaches 0. | If level = 3:  {call WIN()  If int press = key ESC  Then exit  Else if int press = key R  Then replay}  if ammo = 0 :  {call gameover()  ….similar code} |

Figrue 4: PROJECTILE PATH:

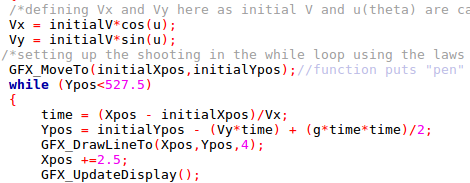
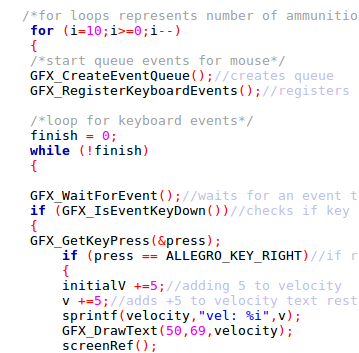


Figure 3: KEYBOARD EVENTS: 

2.Libraries/ User input and output:

**LIBRARIES:**

Table 3:

|  |  |  |
| --- | --- | --- |
| **LIBRARIES** | **SOURCE** | **DESCRIPTION** |
| graphics\_lib.h | Library provided to create game. | Provides GFX. functions. |
| math.h | Standard mathematics library. | Provides possibility of using sin(u) and cos(u) as well as giving values power. As well as this makes use of double variables, which are more precise. |
| stdlib.h | Standard Library | Standard Library |

**User input/output:**

Each input put in by the user gives an output. These inputs and outputs are shown in the table below:

Table 4:

|  |  |  |
| --- | --- | --- |
| **Input** | **Description** | **Output** |
| Left/Right arrow keys (-/+) | Subtracts or adds 5(ms^-1) to the velocity | Visually, velocity value between cyan arrows changes as well as distance of the projectile travelled. |
| Down/Up arrow keys (-/+) | Subtracts or adds 5(radians)  to angle. | Changes the angle value between the yellow arrows and adjust the orientation of the volley when fired. |
| Main Menu Enter | If button is pressed it is used to start game. | Clears main menu screen and refreshes with the main gameplay. |
| Game Enter | Enter in the main game is used to fire. | Enters the while loop with selected values to draw volley path. |
| ESC key | All ESC keys are used to exit the game when shown on the screen. | When pressed closes the game window, leaving you with an open terminal. |
| R key | Used in game-over and WIN screen to play game again. | Used with “goto” function to jump to line 12 of the program as a replay. |

|  |
| --- |
| ***EXTENDED FEATURE*** |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| U key | When the user has acquired coins, they have the chance to buy more ammunition (15 coins gained for shooting the target and one volley costs 5 coins). | User gains ammunition according to how many times they pressed the U hotkey until the amount of coins reaches 0. This can be seen at the bottom left of the window. |

3.Testing:

The plan to testing the game was to have several individuals run and play the program and report any issues. As well as this the program was run itself a couple of times to identify any bugs or unintended responses.

Bugs found:

* Projectile distance decreases after several turns with the same values of angle and velocity used. Bug [A]
* Occasional crashing, after pressing enter once whilst the main game is running (interface freezes), must restart. Bug [B]
* First volley fired makes a vertical line upwards or if the highest velocities are used sometimes, afterwards game continues as normal. Bug [C]

Identified bugs will be reviewed in the Problem Hypothesis section.

**User Testing:**

Table 5:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User | User friendly? Y/N | Times Played | Bug Identified | Fixed? Y/N |
| User 1 | Y | 2 | B | Y |
| User 2 | Y | 1 | Null | Null |
| User 3 | Y | 1 | A | Y |
| User 4 | Y | 1 | C | N |

**EVALUATION:**

1.Problem Hypothesis:

**Fixing Bugs:**

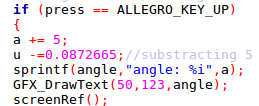
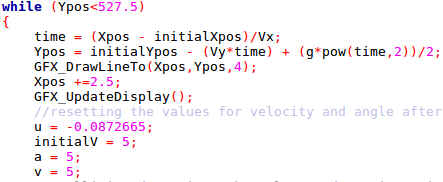
Bug A was identified in the for loop of the program in the main game. It was triggered by the fact that every time the for loop ran instead of resetting the variables of angle and velocity they would be added on breaking the if statement that prohibits the values of exceeding certain limits. As well as this by using the printf() function on the angle as it changed it showed that it became more and more negative even through addition. The way that this was dealt with was to reset the velocity and angle to units of 5 after every time enter is pressed, this was done so by resetting the values in the while loop as well as changing the polarity (e.g. negative) of the angles in the code.

Figure 2:

Figure 1:



Additionally, when Bug B affects the program all the inputs freeze from the main game apart from the ESC keys and R buttons, this indicates that occasionally the code displays the WIN screen without updating the display as the only the ESC and R buttons work. This does not cause an extremely significant change to the game as it can be reset through the R button or rerunning the code by exiting and running it again. This bug was fixed as the if statement of the WIN screen was missing its brackets unlike the if statement for the game-over screen.

Figure 4:

Figure 3:

C:\Users\Seb\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot from 2018-04-12 13-40-41.png

C:\Users\Seb\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot from 2018-04-12 13-41-05 (1).png

Bug C was caused because the variables for velocity (initialV) and the angle (u) have been defined however they have not been given a value therefore the functions for adding or subtracting, add and subtract Null (not zero) therefore when enter is pressed the line drawn is straight horizontal. However, the part of the bug when a high velocity is selected has not been fixed, however a good hypothesis is that it is related to calling the initialV and u as their values may change to an unwanted one.

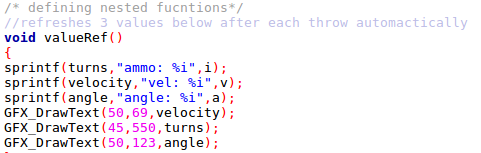
2.Conclusion:

Overall the program has reached the required criteria. The code itself is made to be clear. Having well named variables for others to get a key concept of their roles in the code, as well as placing comments that dictate the roles of the functions used. Besides this, comments have also been added by the ending brackets of loops and if statements to make them more accessible to future modification. There are two types of commenting that can be done in the C Programming language, this is the forward slash (/) and the slash asterisk (/\*) they have been used differently so that the code be split into smaller different subsections for better understanding.

C:\Users\Seb\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot from 2018-04-12 15-58-07.pngThe design of the program is split as said into two segments, the Initialising and the Game sections. This was done so that any person inspecting the code can understand where different parts are called/calculated as well as anticipating in advance the overall design of the code. Therefore, the code has been split into two large sections with smaller subsections identified by the slash asterisk commenting.

Figure 5:

Figure 6:



The program is limited only to the fact that the graphics library provided is to be specifically used which has hindered greater development as there was no variety to choose from. There is also one unsolved bug in the code though it makes an insignificant change as very large values for velocity are not needed to hit the target but only provide a more diverse combination of values to hit the target.

Further improvements can be done to the game, some of these are making the target spawn randomly, adding more levels as well as changing the way that the obstacle spawns to create a more challenging environment.

To conclude the code has been given clarity for others to understand as well as an ordered structure to ensure the program runs smoothly by keeping a good frame rate. The testing proved useful as it showed the rate at which the bugs appeared at, by also running it several times individually the lines of code causing the bugs were identified and fixed.

|  |  |  |
| --- | --- | --- |
| **Title** | **TYPE** | **SOURCE** |
| BITMAP piggy | IMAGE | <https://pixabay.com/en/piggy-bank-piggy-money-coins-1246949/>  created by: simon\_ramone |
| BITMAP cannon | IMAGE | <https://pixabay.com/en/cannon-weapon-medieval-metal-161481/>  created by: OpenClipart-Vectors |
| BITMAP piggybank | IMAGE | <http://www.publicdomainfiles.com/show_file.php?id=13488690016776>  created by: Maw |
| BITMAP gameover | IMAGE | <https://pixabay.com/en/game-over-reminder-post-note-sticker-1432947/>  created by: Marklay62 |
| Ref A | Data | This knowledge was provided in Lab 11 of the internal web:  <https://www.elec.york.ac.uk/internal_web/meng/yr1/modules/Introduction_to_Programming/Labs/Lab11.pdf> |
| Ref B | Data | This knowledge was provided in Lab 10 of the internal web:  <https://www.elec.york.ac.uk/internal_web/meng/yr1/modules/Introduction_to_Programming/Labs/Lab10.pdf> |
| Image Resize | Website | This website was used to resize the BITMAPS:  <http://www.picresize.com/> |

**REFERENCES:**

**APPENDIX A:**

Code:

#include <graphics\_lib.h>

#include <math.h>

#include <stdlib.h>

int main(void)

{

/\*initialising functions for later use\*/

GFX\_InitWindow(700, 600);//opens a graphics window with a width of 700 pixels and a height of 600 pixels

replay: //used with goto function to replay game

GFX\_InitBitmap();//initialise bitmap images for use

GFX\_InitFont();//this function lets you use GFX\_DrawText() to write strings on the graphics window

GFX\_InitKeyboard();//initialises keyboard usage

/\*loading images into BITMAPS\*/

BITMAP piggy = GFX\_LoadBitmap("piggy.jpg");

BITMAP cannon = GFX\_LoadBitmap("cannon.png");

BITMAP piggybank = GFX\_LoadBitmap("piggybank.png");

BITMAP gameover = GFX\_LoadBitmap("gameover.png");

/\*setting up arrays for usage in GFX\_DrawText()\*/

char level[30];

char score[30];

char velocity[30];

char angle[30];

char turns[30];

int lvl = 1;//displays the level played

int scr = 0;

int i = 10;//used in for loop and display turns

int v = 5;//will be showing value of velocity

int a = 5;//will be showing angle

sprintf(velocity,"vel: %i",v);//v stored in velocity array

sprintf(angle,"angle: %i", a);

sprintf(level,"level: %i", lvl);

//sprintf() for i and scr are used in loop as they need updating

/\*creating variables for keyboard\*/

int press;//for button press

int finish;//to initialise rest of loop after selection of values

int over;//for game-over screen

int menu;//for menu screen

int win;//for winning screen

/\*menu here\*/

//draws projectile word on main menu

GFX\_DrawFilledRectangle(0,0,700,600,BLUE);

GFX\_SetColour(RED);

GFX\_DrawLine(145,100,145,200,15);

GFX\_DrawRectangle(145,100,195,140,15);

GFX\_DrawLine(215,140,215,200,15);

GFX\_DrawLine(208,140,250,140,15);

GFX\_DrawLine(243,140,243,158,15);

GFX\_DrawRectangle(265,135,300,190,15);

GFX\_DrawLine(308,190,340,190,15);

GFX\_DrawLine(340,197,340,140,15);

GFX\_DrawLine(340,120,340,130,15);

GFX\_DrawLine(355,150,396,150,12);

GFX\_DrawLine(355,170,396,170,10);

GFX\_DrawLine(355,190,396,190,15);

GFX\_DrawLine(362,150,362,190,13);

GFX\_DrawLine(390,150,390,170,12);

GFX\_DrawLine(410,151,445,151,13);

GFX\_DrawLine(410,190,445,190,13);

GFX\_DrawLine(410,196,410,144,13);

GFX\_DrawLine(455,150,495,150,13);

GFX\_DrawLine(475,135,475,197,13);

GFX\_DrawLine(510,140,510,197,13);

GFX\_DrawLine(510,120,510,130,15);

GFX\_DrawLine(535,125,535,197,15);

GFX\_DrawLine(564,150,605,150,12);

GFX\_DrawLine(564,170,605,170,10);

GFX\_DrawLine(564,190,611,190,15);

GFX\_DrawLine(564,197,564,144,15);

GFX\_DrawLine(605,144,605,175,13);

//how to play

GFX\_DrawText(10,401,"How to play:");

GFX\_DrawText(10,416,"\*Use left and right arrow keys for changing the velocity.");

GFX\_DrawText(10,431,"\*Use up and down arrow keys for changing the angle of the projectile.");

GFX\_DrawText(10,446,"\*Every time piggy bank is broken you receive 15 coins, use them to buy more ammo!");

GFX\_DrawText(10,461,"\*Be careful at level 2 as there will be a wall between you and the piggy bank.");

GFX\_DrawText(10,476,"\*One shooting ray costs 5 coins.");

GFX\_DrawText(10,491,"\*Don't forget to have fun!");

GFX\_DrawText(275,300,"Press ENTER to start");

GFX\_DrawText(275,315,"Press ESC to exit game");

GFX\_UpdateDisplay();

//menu keys

GFX\_CreateEventQueue();

GFX\_RegisterKeyboardEvents();//registers all keyboard presses

menu = 0;

while (!menu)

{

GFX\_WaitForEvent();

if (GFX\_IsEventKeyDown())

{

GFX\_GetKeyPress(&press);

if (press == ALLEGRO\_KEY\_ESCAPE)//escape button(exit game)

{

GFX\_CloseWindow();

}

else if (press == ALLEGRO\_KEY\_ENTER)//enter button (play game)

{

break;

}

}

}

/\*forming ground for game and loading bitmpas\*/

GFX\_DrawFilledRectangle(0,500,700,600,GREEN);

GFX\_DrawBitmap(piggy,300,250);//this function draws bitmaps

GFX\_DrawBitmap(cannon,80,500);

GFX\_DrawBitmap(piggybank,530,515);

GFX\_UpdateDisplay(); //updates the display in order for drawings to appear

/\*calling variables for projectile shooting, based on laws of physics\*/

//using mostly doubles as for cos and sin the calculations are only done in doubles

double initialYpos;

double initialXpos;

double Xpos;

double Ypos;

double initialV;

double Vx;

double Vy;

double time;

double u; //u is the theta I will use to put in cos(u) and sin(u)

float g; //g is gravity

//////IMPORTANT AS MATHS LIB WORKS IN RADIANS: 5 degrees is 0.0872665 radians//////

/\*defining locations for projectiles and giving gravity its value (9.81 m/s^2)\*/

initialXpos = 110;//for the initial positions of x and y setting 110 and 469 as they are the location of where the projectile will come from

initialYpos = 469;

Xpos = initialXpos;//the changing x and y values are set equal to the initial as that is where the projectile starts form

Ypos = initialYpos;

g = 9.81;

/\*drawing triangles as arrows of velocity and angles\*/

GFX\_DrawFilledTriangle(125,50,150,75,125,100,CYAN);//cyan is for velocity

GFX\_DrawFilledTriangle(50,50,25,75,50,100,CYAN);

GFX\_DrawFilledTriangle(125,104,150,129,125,154,YELLOW);//yellow is for angel manipulation

GFX\_DrawFilledTriangle(50,104,25,129,50,154,YELLOW);

GFX\_SetColour(RED);//setting "pen" colour to red

GFX\_DrawText(50,69,velocity);

GFX\_DrawText(50,123,angle);

GFX\_DrawText(63,170,"Enter!");

GFX\_DrawText(45,550,"ammo: 10");

sprintf(score,"coins: %i",scr);

GFX\_DrawText(45,563,score);

GFX\_DrawText(45,576,level);

GFX\_DrawRectangle(52.5,165,122.5,190,3);

GFX\_DrawText(45,537,"Press U to buy ammo!");

GFX\_UpdateDisplay();

/\* defining nested functions\*/

//refreshes 3 values below after each throw automatically

void valueRef()

{

sprintf(turns,"ammo: %i",i);

sprintf(velocity,"vel: %i",v);

sprintf(angle,"angle: %i",a);

GFX\_DrawText(50,69,velocity);

GFX\_DrawText(45,550,turns);

GFX\_DrawText(50,123,angle);

}

//refreshes screen

void screenRef()

{

GFX\_ClearWindow();

GFX\_DrawFilledRectangle(0,500,700,600,GREEN);

GFX\_DrawBitmap(piggy,300,250);

GFX\_DrawBitmap(cannon,80,500);

GFX\_DrawFilledTriangle(125,50,150,75,125,100,CYAN);

GFX\_DrawFilledTriangle(50,50,25,75,50,100,CYAN);

GFX\_DrawFilledTriangle(125,104,150,129,125,154,YELLOW);

GFX\_DrawFilledTriangle(50,104,25,129,50,154,YELLOW);

GFX\_DrawText(50,69,velocity);

GFX\_DrawText(50,123,angle);

GFX\_DrawText(63,170,"Enter!");

sprintf(score,"coins: %i",scr);

GFX\_DrawText(45,563,score);

sprintf(turns,"ammo: %i",i);

GFX\_DrawRectangle(52.5,165,122.5,190,3);

GFX\_DrawText(45,550,turns);

GFX\_DrawText(45,576,level);

GFX\_DrawBitmap(piggybank,530,515);

GFX\_DrawText(45,537,"Press U to buy ammo!");

}

//screen refreshes and level up with collision detection of piggy bank

void levelup()

{

lvl +=1;

scr +=15;

GFX\_ClearWindow();

GFX\_MoveTo(initialXpos,initialYpos);

GFX\_DrawFilledRectangle(0,500,700,600,GREEN);

GFX\_DrawBitmap(piggy,300,250);

GFX\_DrawBitmap(cannon,80,500);

GFX\_DrawFilledTriangle(125,50,150,75,125,100,CYAN);

GFX\_DrawFilledTriangle(50,50,25,75,50,100,CYAN);

GFX\_DrawFilledTriangle(125,104,150,129,125,154,YELLOW);

GFX\_DrawFilledTriangle(50,104,25,129,50,154,YELLOW);

GFX\_DrawText(50,69,velocity);

GFX\_DrawText(50,123,angle);

GFX\_DrawText(63,170,"Enter!");

sprintf(turns,"ammo: %i",i);

sprintf(score,"coins: %i",scr);

GFX\_DrawText(45,563,score);

GFX\_DrawRectangle(52.5,165,122.5,190,3);

sprintf(level,"level: %i",lvl);

GFX\_DrawText(45,550,turns);

GFX\_DrawText(45,576,level);

GFX\_DrawText(45,537,"Press U to buy ammo!");

GFX\_UpdateDisplay();

GFX\_PauseFor(325);//so pig disappears for 0.325s

}

//collision with wall

void hitwall()

{

GFX\_ClearWindow();

GFX\_DrawFilledRectangle(0,500,700,600,GREEN);

GFX\_DrawBitmap(piggy,300,250);

GFX\_DrawBitmap(cannon,80,500);

GFX\_DrawFilledTriangle(125,50,150,75,125,100,CYAN);

GFX\_DrawFilledTriangle(50,50,25,75,50,100,CYAN);

GFX\_DrawFilledTriangle(125,104,150,129,125,154,YELLOW);

GFX\_DrawFilledTriangle(50,104,25,129,50,154,YELLOW);

GFX\_DrawText(50,69,velocity);

GFX\_DrawText(50,123,angle);

GFX\_DrawText(63,170,"Enter!");

sprintf(turns,"ammo: %i",i);

sprintf(score,"coins: %i",scr);

GFX\_DrawText(45,563,score);

GFX\_DrawRectangle(52.5,165,122.5,190,3);

sprintf(level,"level: %i",lvl);

GFX\_DrawText(45,550,turns);

GFX\_DrawText(45,537,"Press U to buy ammo!");

GFX\_DrawText(45,576,level);

GFX\_UpdateDisplay();

}

//draws screen with no projectile path after collisions

void noPROJECTILE()

{

GFX\_DrawFilledRectangle(0,500,700,600,GREEN);

GFX\_DrawBitmap(piggy,300,250);

GFX\_DrawBitmap(cannon,80,500);

GFX\_DrawFilledTriangle(125,50,150,75,125,100,CYAN);

GFX\_DrawFilledTriangle(50,50,25,75,50,100,CYAN);

GFX\_DrawFilledTriangle(125,104,150,129,125,154,YELLOW);

GFX\_DrawFilledTriangle(50,104,25,129,50,154,YELLOW);

GFX\_DrawText(50,69,velocity);

GFX\_DrawText(50,123,angle);

GFX\_DrawText(63,170,"Enter!");

sprintf(turns,"ammo: %i",i);

sprintf(score,"coins: %i",scr);

GFX\_DrawText(45,563,score);

GFX\_DrawRectangle(52.5,165,122.5,190,3);

GFX\_DrawText(45,550,turns);

GFX\_DrawText(45,576,level);

GFX\_DrawBitmap(piggybank,530,515);

GFX\_DrawText(45,537,"Press U to buy ammo!");

}

//winning screen however button functions were not added as adding the goto function for replay will cause an error the same as the game over screen

void WIN()

{

GFX\_DrawFilledRectangle(0,0,700,600,YELLOW);

GFX\_DrawText(317,250,"YOU WON!!!");

GFX\_DrawText(260,265,"Press R for to play again");

GFX\_DrawText(260,280,"Press ESC to exit game");

GFX\_UpdateDisplay();

GFX\_CreateEventQueue();

GFX\_RegisterKeyboardEvents();

win = 0;

}

//draws game-over screen

void Gameover()

{

GFX\_DrawFilledRectangle(0,0,700,600,BLUE);

GFX\_DrawBitmap(gameover, 350,300);

GFX\_DrawText(425,500,"R = Replay");

GFX\_DrawText(425,520,"ESC = Exit");

GFX\_UpdateDisplay();

over = 0;

}

/////////////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////////////////

////////////////////////\*main game starts here\*//////////////////////////////////////

/\*giving initialV and u(theta) values in order to add or subtract through key press when they are called later on\*/

initialV = 5;

u = -0.0872665;

/\*for loops represents number of ammunition available at the start\*/

for (i=10; i>=0; i--)

{

/\*start queue events for mouse\*/

GFX\_CreateEventQueue();//creates queue

GFX\_RegisterKeyboardEvents();//registers keyboard events

/\*loop for keyboard events\*/

finish = 0;

while (!finish)

{

GFX\_WaitForEvent();//waits for an event to happen

if (GFX\_IsEventKeyDown())//checks if key is pressed

{

GFX\_GetKeyPress(&press);

if (press == ALLEGRO\_KEY\_RIGHT)//if right arrow is pressed

{

initialV +=5;//adding 5 to velocity

v +=5;//adds +5 to velocity text rest of code is to redraw on window

sprintf(velocity,"vel: %i",v);

GFX\_DrawText(50,69,velocity);

screenRef();

if (lvl >= 2)//represents level 2 by adding obstacle, if statement across programme checks which level the game is on

{

GFX\_DrawFilledRectangle(300,430,310,522,BLACK);//creates obstacle if level 2 is detected

}

GFX\_UpdateDisplay();//keeping update after screen ref in order for the obstacle to stay on screen

if (v>=100 && initialV>=95)//if velocity reaches 100 it does not let it increase

{

initialV -=5;

v -=5;

}

}

else if (press == ALLEGRO\_KEY\_LEFT)

{

initialV -=5;

v -=5;

sprintf(velocity,"vel: %i",v);

GFX\_DrawText(50,69,velocity);

screenRef();

if (lvl >= 2)

{

GFX\_DrawFilledRectangle(300,430,310,522,BLACK);

}

GFX\_UpdateDisplay();

if (v<=0 && initialV<=0)//does not let velocity go under 0

{

initialV +=5;

v +=5;

}

}

//doing the same for angles

if (press == ALLEGRO\_KEY\_UP)

{

a += 5;

u -=0.0872665;//subtracting 5 radians instead of adding as this is dependent on the plane of the screen so it shoots in the correct direction

sprintf(angle,"angle: %i",a);

GFX\_DrawText(50,123,angle);

screenRef();

if (lvl >= 2)

{

GFX\_DrawFilledRectangle(300,430,310,522,BLACK);

}

GFX\_UpdateDisplay();

if (a>90 && u>-1.5708)//1.5708 is 90 degrees

{

u +=0.0872665;

a -=5;

}

}

else if (press == ALLEGRO\_KEY\_DOWN)

a -=5;

u +=0.0872665;

sprintf(angle,"angle: %i",a);

GFX\_DrawText(50,123,angle);

screenRef();

if (lvl >= 2)

{

GFX\_DrawFilledRectangle(300,430,310,522,BLACK);

}

GFX\_UpdateDisplay();

if (a<=0 && u>=0)

{

u -=0.0872665;

a +=5;

}

else if (press == ALLEGRO\_KEY\_U)//buying ammo

{

if (scr>0)

{

i +=1;

scr -=5;

}

screenRef();

if (lvl >= 2)

{

GFX\_DrawFilledRectangle(300,430,310,522,BLACK);

}

GFX\_UpdateDisplay();

}

else if (press == ALLEGRO\_KEY\_ENTER)

{

/\*defining Vx and Vy here as initial V and u(theta) are called in the function below, as they will be called below\*/

Vx = initialV\*cos(u);

Vy = initialV\*sin(u);

/\*setting up the shooting in the while loop using the laws of physics based on the named variables above\*/

GFX\_MoveTo(initialXpos,initialYpos);//function puts "pen" to the starting location from where the projectile path will be drawn from

while (Ypos<527.5)

{

time = (Xpos - initialXpos)/Vx;

Ypos = initialYpos - (Vy\*time) + (g\*pow(time,2))/2;

GFX\_DrawLineTo(Xpos,Ypos,4);

Xpos +=2.5;

GFX\_UpdateDisplay();

//resetting the values for velocity and angle after shooting

u = -0.0872665;

initialV = 5;

a = 5;

v = 5;

/\*collision detection takes place and getting coins\*/

//hitting piggy bank

if (500<=Xpos && 560>=Xpos && 510<=Ypos && 535>=Ypos)//collisions in the game work through checking where the x and y pos of the head of ray are

{

levelup();

break;

}

//hitting wall

else if (lvl == 2 && Xpos>=300 && Xpos<=310 && Ypos>=430)

{

break;

hitwall();

}

}

i -=1;//decreases number of ammo as while loop cannot finish so internally decreasing the ammunition, if this is moved under valueref() then ammo wont update automatically and will do so at next button press

valueRef();

Xpos = 110;//resetting X and Y positions in order to shoot again

Ypos = 469;

GFX\_ClearWindow();//clears all window

/\*re-adding functions so only projectile path is deleted after ray hit obstacle or ground or piggy bank\*/

noPROJECTILE();

if (lvl >= 2)

{

GFX\_DrawFilledRectangle(300,430,310,522,BLACK);

}

GFX\_UpdateDisplay();

//game-over screen//

if (i==0)

{

Gameover();

while (!over)

{

GFX\_WaitForEvent();

if (GFX\_IsEventKeyDown())

{

GFX\_GetKeyPress(&press);

if (press == ALLEGRO\_KEY\_ESCAPE)

{

GFX\_CloseWindow();

}

else if (press == ALLEGRO\_KEY\_R)

{

goto replay;

}

}//gfx is event down here, adding comments for certain brackets for easy identification if modification is needed

}//while loop here

break;

}//if i == 0 statement ends here

//winning screen

else if (lvl == 3)//when level 2 piggy bank is hit winning screen is shown

{

WIN();

while (!win)

{

GFX\_WaitForEvent();

if (GFX\_IsEventKeyDown())

{

GFX\_GetKeyPress(&press);

if (press == ALLEGRO\_KEY\_ESCAPE)

{

GFX\_CloseWindow();

}

else if (press == ALLEGRO\_KEY\_R)

{

goto replay;

}

}

}

}

}

}

}

}

return 0;

}

**Bitmaps:**

|  |  |
| --- | --- |
| **Bitmap** | Description |
| Piggy Bank, Piggy, Money, Coins, Savings, Save, Bank | This BITMAP image was used as the background. |
| Cannon, Weapon, Medieval, Metal | This BITMAP image was used as the cannon. |
| Illustration of a cartoon pig | BITMAP image used for piggy bank. |
| Game Over Reminder, Post Note, Sticker, Sticky Paper | Used to cover whole screen for game-over screen. |