IOWA STATE UNIVERSITY

AEROSPACE ENGINEERING DEPARTMENT

COMPUTATIONAL TECHNIQUES FOR AEROSPACE DESIGN

AERE 361

PROJECT PROPOSAL

THE B-TEAM

Team Member Names :
White, Tristen
McCarrick, Gerald
Blochowitz, John
Strobbe, Mitchell
Case, Brandon

Contents

I ABSTRACT

This project includes the following members: Tristen White, Gerald McCarrick, John Blochowitz, Mitchell Strobbe, and Brandon Case. Side-scrolling games have been around for a very long time. One of the most popular, "Doodle Jumper" came out in 2009. Vertical scrolling games are even older. One example of the most famous would be "Flappy Bird" released in 2013. Our game will take the gameplay component of Flappy bird (ex. moving an object through obstacles) and integrate it with the tilt controls and vertical scrolling layout of "Doodle Jumper." Our main objective for this project is to create a game where the user has to stay active from start to end so they are always engaged. We will use the C language to create this game and display the visuals through the Adafruit CLUE board. Throughout the development process, we will continuously test our code and make critiques to ensure the game runs smoothly with little to no bugs. We not only want our game to be fully functional but efficient and compact as well. With our beginner level of knowledge, we hope to create a functional and nostalgic game for those who wish to play.

II INTRODUCTION

Vertical-scrolling games have been around since the 1970s. These games are typically coded using subroutine blocks. These blocks of code control what is added to the game, such as obstacles, a user menu, and the player's character ([2]). In the case of our game, the purpose is to use tilt controls to roll a ball left and right to avoid random obstacles moving toward the player. A timer will keep track of the amount of time a player survives. When the ball collides with an object, the player loses and is returned to the start of the game. We, Tristen White, Gerald McCarrick, John Blochowitz, Mitchell Strobbe, and Brandon Case, intend to recreate this well-known game to give the user a glimpse into the nostalgia of a retro-feeling game.

III FEATURES

Our game will be controlled with a simple interface of 2 buttons. We feel two buttons should be the bare minimum features of our project since a minimum of 2 buttons would be needed to control the menu and to exit the game. If pressed again, one button will be used to start playing along with exiting the game. The other will be used to view the high score; if held, it will turn off the handheld.

The main method of controlling the ball to go between the gaps in the pillars will be with the built-in IMU (ST Micro series 9-DoF motion sensors). This will allow us to use tilt controls to move the ball from side to side. Many similar smartphone games use a similar control method, such as "Doodle Jump," where tilting the phone causes the player to move from side to side. In our case, this is necessary to make the game more difficult.

Ideally, it will be able to fit into a compact handheld format. This would mean it should be free of cables and able to run solely on 3 AAA or AA batteries. The device's enclosure will be 3d printed; however, we aim to make it aesthetically pleasing and easy to hold. We think this is necessary because no one would want to play a game while having to hold onto a circuit board. The device's casing should also withstand slight abuse but nothing that would involve throwing or dropping.

Another feature of the handheld would be that it should have an electronic buzzer. This would give tactile feedback for when the ball collides with the obstacles. If the player loses, it should play a losing sound. While this feature isn't technically necessary to the function of the game, many other retro handhelds have a simple buzzer implemented to play sound effects and music.

Integrating a high score counter is also another part that would be important. This would allow players to see the highest score on the device. The high score could be based on how quickly levels are beaten, the highest level achieved, or the longest survival time. We have not decided what score metric we will use or if we will use multiple metrics. This detail will be decided once we have finalized the characteristics of the game's different levels.

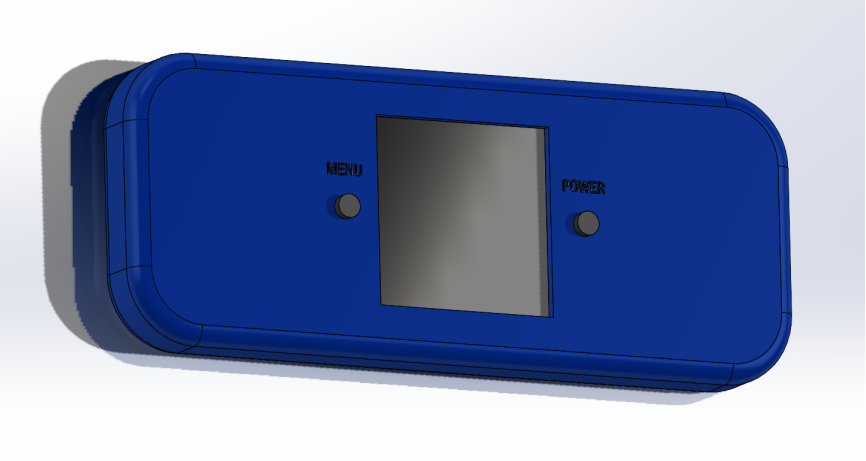

Figure 1: Handheld Concept Art

IV PROBLEM STATEMENT

One of the most difficult issues in creating the code will be ensuring collisions between the ball and the obstacles are recognized. Recording and assessing a collision will require the scrolling component to respond to a rather small time step. Otherwise, on-screen collisions during gameplay will make the game slow and sluggish, and completely dysfunctional at worst. We must also constrain the ball's movement within the bounds of the screen, as it would not be good for the ball to vanish from the user's vision and "fall off" the screen while playing.

Another challenge will be making the game progressively more difficult the longer the player plays. This would involve a separate time step only linked to the movement of the pillars. As a substitute for that, the obstacles may also be able to get more difficult. Either way, making randomly generated obstacles appear frequently will be a challenge.

Figuring out how to convert IMU data to tilt controls will be difficult. It shouldn't be the most difficult part of the game; however, raw data has outliers, so the sensor will most likely need some filter to ensure the controls feel as intuitive as possible when tilting the device.

While nowhere near the challenge compared to any previously mentioned problems, storing the high score may also be an issue. It has to be able to save so that you can view the furthest that someone else has made it on the device. This will involve our team having to devise a counting system for the number of obstacles passed or the total time for a run.

V PROBLEM SOLUTION

All of the problems outlined in this report are primarily software related. As such, we must fully use C and its features and capabilities to create a fully functional game.

The ball and incoming obstacles will move in one dimension in our game. The ball will move left and right, whereas the incoming obstacles (which we will refer to as "bricks" throughout this report) will move vertically from the top of the screen toward the ball. Moving the ball left and right is relatively straightforward with button inputs- we define a variable for the ball's position, width, and height and then write a conditional if-else statement telling the ball to move left or right depending on if the left or right button is pressed. A function could be written in C to do this repeatedly, and said function would take the arguments of initial ball position, ball width, and height and update the ball's new position will a call of the function based on the left or right input of the user. However, our game will not use manual buttons but rather tilt controls. Still, the logic behind controlling the ball is the same- use conditional statements based on the tilt sensor's input to determine if the ball will move left or right. As for the bricks, C comes equipped with structures. A structure contains variables called members that can be initialized with "*struct.*" Variables can have the same name without conflict since they can always be distinguished by context ([1]), which we would use to declare an array of bricks to serve as obstacles for the ball to maneuver around. Said structure could contain important information about the bricks, mainly brick position (as x and y coordinates), brick width, and brick height. This information could then be passed into a function as inputs, and then the function will output the new position of the bricks (a similar strategy to moving the ball). We would need to initialize the speeds of the bricks beforehand with a separate variable.

It is also worth noting that the screen on the Clue board will need to hold the data within the structures. This is plausible, as structures can be used to organize information in a way compatible with small screens, such as mobile device screens ([3]).

As stated in the Problem Statement, we would like our game to feature increasing difficulty as the player gets farther along. To increase the difficulty, there are three options we see as good solutions:

- Increased scroll rate
- Increased obstacle density
- Harder optimal path

The first is probably the easiest to implement, assuming the microcontroller and screen can process the graphics quickly enough. We can increase the scroll rate by running the game clock is a higher frequency.

The second option will also not be so hard to implement; as the game progresses, place more obstacles on the screen. This will require the user to process the information faster as more obstacles are coming. If all goes well with the structure and function method of generating bricks, we will already have a mechanism to do this. We can either update the speed of the

moving bricks with each new level by gradually increasing the speed variable, which could be done with a "for" or "while" loop that updates every level. Alternatively, when randomly assigning the bricks to different locations, the program could create more bricks to avoid with each level. Both of these techniques would add difficulty to the game, and we may even be able to incorporate both into the game for higher levels.

The third option is probably the hardest to implement. The player will have to move to avoid the obstacles. With increased density or scroll speed, this becomes harder, but we can also make the player move further to avoid the obstacle as they progress through the game. This option is the hardest but also the most challenging. So this difficult enhancement will only come late game, if at all.

Grabbing and filtering the data from the IMU to get an accurate angle reading may be a little challenging, but as it is not a new idea, there is probably a lot of good information on achieving this.

We could implement the tilt as a position control or velocity control. For example, the board's angle determines the user's position, or it could determine how fast the user's position changes. While the second one would be much more finicky, we think it would make the game more difficult. Because of this, we will start with the angle-controlling position and switch it over to accelerating control if the game proves too easy.

As previously said, the high score component will likely not be difficult to implement. The hardest part will be implementing a system that allows the user to decide which person to store the high score to. This feature will likely require a new menu, but it should be doable with only a two-button and angle tilts control system.

As a final note, whichever route we choose to solve these problems, we will work to ensure that our code is as lean as possible. This is necessary to improve run-time performance and ensure the hardware controls adjust quickly enough to user inputs when playing the game. Nobody likes a "laggy" game!

Table 1: Parts Requested

Part Description	Qty
Adafruit Clue Board	1
AAA Battery Holder	1
USB Cable	1
3D Printed Enclosure	1
Arcade Style Selection Buttons	2
*Buzzer (if not included on the Clue Board)	1
*On-Board Speaker (if we play a losing sound)	1
"*" denotes potentially need	

VI CONCLUSION

Our team plans to begin by ensuring the code is developed first and as bug-free as possible. We hope to be able to spend a lot of time debugging and optimizing the code. One of the last steps of our project will be dedicating time to ensure the form and function of the 3D-printed enclosure. We will do our best to stick to the parts list outlined in this report, but we have a few backup plans if things don't go according to plan. With this project, we hope to deliver a functional and fun experience for anyone interested in retro-style games.

References

[1] Dennis M. Ritchie Brian W. Kernighan. *The C Programming Language*. Prentice Hall, 1978. ISBN: 9780137460847.

[2] Minhee Chae and Jinwoo Kim. "Do size and structure matter to mobile users? An empirical study of the effects of screen size, information structure, and task complexity on user activities with standard web phones". In: *Behaviour & Information Technology* 23.3 (2004), pp. 165–181. DOI: https://doi.org/10.1080/01449290410001669923.

[3] Nonki Takahashi. *Small Basic Game Programming- Vertical Scrolling Game*. URL: https://techcommunity.microsoft.com/t5/small-basic-blog/small-basic-game-programming-vertical-scrolling-game/ba-p/336809. (accessed: 02.24.2023).