Arduino Game

Arduino The Worms with 2 Nokia 5110/3310 LCD Displays

Ivan Vesel, Miguel Lucas, Nuno Pinto
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal

Abstract— In this article, it is described the implementation of a video-game, based on an Arduino board. Arduino The Worms is a Player vs Player (PvP) game, rules based on the real The Worms game, where each display belongs to each player - players can't go from one display to another. The player controls the worm - movement, aiming and shot - via Bluefruit LE smartphone application, that connects to the bluetooth Bluefruit LE shield attached to the Arduino. A more detailed explanation of the problem, as well as the hardware used in the project will be given. It will also be presented the approach followed to schedule the control tasks to achieve the desired system behaviour. Furthermore, a discussion on the measurements made to verify the execution and response times will be presented.

Index Terms—*Arduino, Bluetooth, Nokia 3310 LCD, The Worms, Bluefruit, Adafruit, Breadboard.*

- *Ivan Vesel is with the Informatics Engineering Department, University of Porto, Porto. E-mail: up201711398@fe.up.pt.*
- *Miguel Lucas is with the Informatics Engineering Department, University of Porto, Porto. E-mail: ei11140@fe.up.pt.*
- *Nuno Pinto is with the Informatics Engineering Department, University of Porto, Porto. E-mail: up201307878@fe.up.pt.*

I. INTRODUCTION

AN embedded system is an electronic device, which conducts a specific function, normally associated with a real-time constraint. This means that the system must guarantee a specific response within a specified time, in order to accomplish the desired deadline of the control tasks involved.

In the Embedded Systems project context was required a creation of a real-life system, where previous characteristics must be fulfilled. With this, the famous *The Worms* game was developed, an artillery tactical computer game first released in 1995. The main objective of the game is to shoot your opponent, using a smartphone application to control your character. A player's turn ends when he shoots and the opponent turn starts immediately. Both characters can't move on each others turn and until the bullet hits an object (a wall or the opponent), or disappears from the screen (out of boundaries). The bullet's trajectory is a parable, calculated according to the aiming degree.

The real-life requirements are the representation and constant changes on the displays, in a very short-time, and the constant bluetooth state command's checking.

II. HARDWARE

The Arduino chosen was the Arduino Uno featuring an 8-bit ATMega328P as a processor and 32kb Flash Memory. This board was selected due to the number of digital I/O ports available which were required for both bluetooth (6 ports, plus SPI port) and for both displays (7 ports total).

In order to assembly the bluetooth shield - Adafruit Bluefruit LE Shield [2] - several crucial steps were required: attaching the female headers to all Arduino's pins, place the shield on top of the Arduino (all the pins should stick out through the matching holes in the shield), solder in all used headers and attach new female wires to the other headers. After the whole assembly, the bluetooth shield uses hardware SPI by default: Serial Clock (SCK) on pin 13, Master In Serial Out (MISO) on pin 12 and Master Out Slave In (MOSI) on pin 11. It also uses pin 8 for Chip Select (CS), pin 7 for Interrupt Request (IRQ) and pin 4 for Reset (RST). Bluefruit bluetooth's configuration is in BluefruitConfig.h.

On the other hand, the Nokia 3310 LCD displays were pretty simple to wire theoretically, although big problems came when trying to wire them with the Arduino. These displays have 8 pins: Power-supply (VCC), Ground (GND), Chip Select (SCE), Reset, Data Command (DC), MOSI, SCK and LED. For a perfect use of both displays, 13 pins were required - VCC, GND and LED can be shared. Unfortunately, with arduino having 14 digital I/O pins available and bluetooth already using 6, and pin number 0 and number 1 can't be used because they use TTL logic levels, there were only 6 pins remaining. After an intense search and several failed experiences we come up with a solution: share all pins from both displays with the arduino but two pins - SCE and RST. This way, both LCD could display different content, without any bugs or mistress. A breadboard was used to share the pins.

III. SOFTWARE

The software that controls the game was built following

a hybrid approach - cyclic plus interrupts. The Bluetooth module is based on interrupts, and commands from the smartphone application are read with a timeout of 50 ms. Since the user interface is button-based, this timeout is large enough to receive all kind of inputs, without disruption. All the other tasks (to enumerate later) follow a cyclic approach.

A. State Machine

A state machine was also used to easily switch between the different game states and user states to identify the current player. The following game states were defined:

1. Main menu - the first part of this state is the Bluetooth connection. The application asks the user to connect with the Arduino, and a confirmation message is shown once connected. After the connection is established, the user can choose if he wants to start the game or exit the application. If the connection is lost throughout the game, it becomes "paused" and waits for a reconnection to resume the game.

2. Moving phase - this is where the players, each on his turn, can move throughout the field, using the left and right arrow buttons, or jump forward or backwards using the up and down arrows. When the user is satisfied with his placement, he can press the button "1" to enter the aiming phase.

3. Aiming phase - in this phase, a small reticle appears in front of the player, which represents in which angle he wants to shoot. By clicking the up and down arrows, the player can move the aim up and down. The aim movement is restricted to various points of a quarter of circle, calculated with a radius of 10 pixels from the center of the player, and is limited by the x-axis (x=0) and the y-axis (y=0). Then, the user can press the button "1" to shoot in the angle chosen, or press "2" to cancel the shot, and return to the previous state (moving phase).

4. Shot phase - the initial position of the shot will be the position of aiming reticle, and is given an initial pre-defined velocity, vertical and horizontal resistances are also pre-defined to simulate gravity and air resistance. The shot motion follows a parable, calculated according to physic rules. After the shot collides with an object, the player's turn ends and the opponent turn starts, going back to phase number 2.

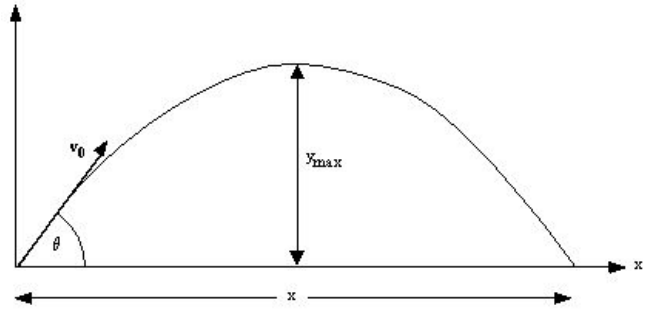

Fig. 1. Graphic showing the standard bullet's trajectory and two of the main points - Maximum Height (ymax) and Maximum Distance (x).

4.1. Trajectory time - Time that the bullet takes to do the whole trajectory, and to reach the maximum height.

$$t = \frac{2 * v_0 * \sin(\theta)}{g} \quad ; \quad t_{y_{max}} = \frac{t}{2} = \frac{v_0 * \sin(\theta)}{g}$$

4.2. Maximum Height - This way we know the y position of the bullet on top.

$$y_{max} = y_0 + v_{0y} * t - \frac{g * t^2}{2} \quad ; \quad v_{0y} = v_0 * \sin(\theta)$$

4.3. Maximum Distance - This way we know the x position of the bullet at the end of its movement, and the x position of the bullet on the maximum height.

$$x = \frac{v_0^2 * \sin(2\theta)}{g} \quad ; \quad x_{y_{max}} = \frac{x}{2}$$

4.4. Position at each time - This way we know the position of the bullet in the desired time.

$$\begin{cases} x_{(t)} = v_0 * \cos(\theta) * t \\ y_{(t)} = v_0 * \sin(\theta) * t - \frac{g * t^2}{2} \end{cases}$$

5. Game over - After one of the players reaches 0 Hp, the game ends with a message on both screens displaying a victory message.

B. Structs

Different structs were used to easily represent each of the different objects of the application: Game, Bluetooth, Worm, Aim, Shot and Block. The Block struct is a way to represent the obstacles present in the map, and contain the initial X and Y position, and its width and height. This makes it very easy to draw a map, coupling some different blocks.

IV. MAIN TASKS

A. Tasks

1. Draw display 1 - this task consists in drawing every content to show on display 1: the player 1 worm, the blocks, the aim and the shot, if existent.

2. Draw display 2 - similar as the previous task, but related to the second display. To draw a transition between the two displays, and although each display has the same size (84x48 pixels), a larger combined width was considered for the object position (84*2=168 pixels). When the position of an object plus his width was superior the width of the first display, the rest of the object was drawn of the second display, and vice versa.

3. Ground detection - every 50ms, the program verifies if each of the players is touching ground. If they are, nothing is done. If they aren't, a downwards force is applied, and they fall down. If a player reaches the end of the display, they lose the game.

4. Jumping - when the user orders a jump, the worm jumps to a predefined height. The length of the jump will be two times the height of the jump.

5. Shooting - this is the task responsible for calculating the shot trajectory and executing its movement. The shot is destroyed when it collides with an object. There are three types of collisions:

A. Display Boundaries: if a shot hits one of the boundaries of the display

B. Blocks : if a shot hits one of the blocks, the block is also destroyed.
C. Player: if a shot hits a player, one Hp is reduced

6. Bluetooth - the Bluetooth module tries to read a packet every 50 ms, and if none is received, ends the game loop. If one is received, it parses the received packet into a readable command and directs the code flow accordingly to the command. It will only call the functions or the beforementioned tasks if the respective button or flow is followed.

B. Execution times measurement

In order to obtain the execution times for each task, a total of 100 repetitions of each were executed, and the maximum execution times were saved.

Task	**Max. Execution Time (ms)**
Draw display 1	33
Draw display 2	28
Ground detection	58 (with 50 ms delay)
Jumping	54 per height pixel
Shooting	193 (with 100 ms delay)
Bluetooth	50

Fig. 2. Table showing the maximum execution time for each task

C. Memory

The memory usage of the Arduino totals to 1501 bytes of the 2048 available, which represent 73% of the available memory.

V. CONCLUSION

The aim of this project was to develop a The Worms based video-game, where the system must have the capability of responding to multiple situations in a certain time. It was shown that all the objectives were contemplated and successfully completed, within each function time constraints. Hence, the game has a high playability, the characters smoothly move on the display, bullets can successfully detect objects. Although it was developed with success, improvements could be performed, especially in what concerns the complexity of the game. Adding one more bluetooth module, would allow both players to play simultaneous, avoiding turns. Besides appearing a very simple game, it was required a lot of coding hours and mathematical knowledge in order to be successful.

VI. GROUP ORGANIZATION

We can distinct 3 distinctive phases over the last month. The first two weeks and a half were focused on assembly the bluetooth shield and connecting the arduino and the displays. This was done only on class schedule. This process was very challenging because we had never been in touch with any kind of electronics physical hardware (displays, sensors, etc.), not even the Arduino, and we had a lot of display replication problems. The second phase was starting testing, getting used to the Adafruit library, for both bluetooth and displays. This took us one week and a half and we met three times - once in the class schedule and twice out of the class schedule. The third phase was dedicated exclusively to project development, and took us 3 weeks. We met 7 times - three times on class and four times extra class. The hardware was taken home by one of our members (selected each week randomly) that was also responsible for improving the project if possible.

Ivan Vesel - 25%.
Miguel Lucas - 37.5%.
Nuno Pinto - 37.5%.

VII. ACKNOWLEDGEMENT

The authors wish to thank Prof. Luís Almeida and Prof. Mário Sousa for supporting and advising throughout the whole project.

VIII. REFERENCES

[1] Adafruit, Bluefruit LE Shield. [Online]. Available: https://learn.adafruit.com/adafruit-bluefruit-le-shield. [Access: May 2018].
[2] FEUP, Conteúdos de Sistemas Embarcados. [Online]. Available: https://sigarra.up.pt/feup/pt/conteudos_geral.ver?pct_pag_id=249640&pct_parametros=pv_ocorrencia_id=399930. [Access: May 2018]
[3] Charismatic Arduino, Say 'Hello World' with Nokia 3110/5110. [Online].Available: http://lunchwitharduino.blogspot.com/2015/02/say-hello-world-with-nokia-31105110.html. [Access: May 2018]
[4] Github, Running Multiple Displays on a Single Processor. [Online].Available: https://github.com/olikraus/u8g2/issues/228. [Access: May 2018]
[5] Adafruit, Bluefruit LE Shield Guide Contents. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bluefruit-le-shield.pdf. [Access: May 2018].
[6] Universidade do Minho, Lançamento de Projéteis. [Online]. Available: https://repositorium.sdum.uminho.pt/bitstream/1822/23542/1/Lancamento%20de%20Projeteis.pdf. [Access: May 2018]

IX. ANNEX

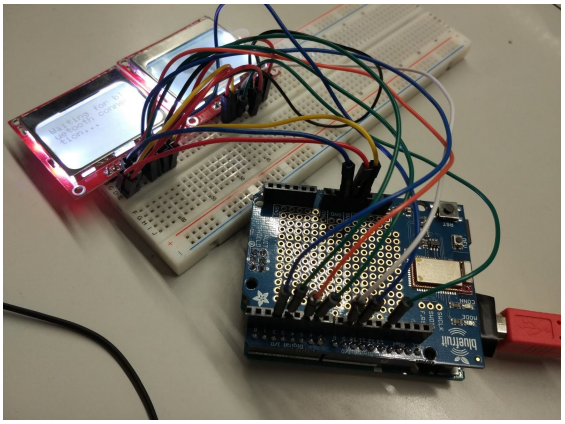

Fig. 3. Circuit assembly.

Simple video of the project demonstration:
https://youtu.be/lLk39jCllTs