Novosibirsk State University

**4COM1042 [Computing Platforms] Co-design Group Project B “The Game of TV-Tennis”**

Vasilii Krukovskii 21934

Mark Skvortsov 21934

Vladimir Lyoshin 21934

[**HARDWARE**](#_6xsw32vywotf) **3**

[Displays](#_bdbmh1jjlcn6) 3

[Main display](#_81mu52xq4h5) 3

[Information displays](#_4xzb7zkyhe19) 3

[Modules](#_bj8ubgwj5g4x) 3

[Main display implementation](#_27hi4iihgmqe) 4

[Eight-column control module](#_3ixxuqd49hh9) 4

[One-column control module](#_pc0rezp3hbad) 5

[Bit Shift module](#_2jtwjxqncinw) 5

[information displays](#_kh501om8odsp) 6

[Score counter](#_d6uz28i93img) 6

[Game menu](#_c805do7rwkqr) 6

[Game process](#_jhe3o54dolr6) 7

[**SOFTWARE**](#_sz3jzez011pz) **7**

[Introduction](#_4kxzfdyg8w3z) 7

[Code](#_8ej0zrlwhrum) 8

[Circuits in Logisim](#_ysx5omcak4mt) 11

[Store](#_i4u6jcjoj7q3) 11

[Bot](#_h240oquw4qb7) 12

[Ring and directory](#_6owq18p3zckp) 12

[Random and Win or Lose](#_u05t0qtnmk9r) 13

[The ball bounces](#_y0px3kj1c4xo) 13

[Ball bounce conditions](#_o2um9mcx7xm8) 13

[Changing the angle of inclination or momentum of the racket](#_licw6q38qhih) 14

[**Conclusion**](#_shym33ermd1) **17**

[**Bibliography**](#_i5r964nr6zae) **17**

# HARDWARE

## Displays

### Main display

We use two 16x32 pixels displays connected together to show a user the progress of a game. Information display shows the score rating of the current game session. Main central display has 2 rackets for making ball bouncing between players: the first racket belongs to the user and locates on the left side, and the second racket belongs to the bot and locates on the right side of the display.

### Information displays

There are also two 32x7 pixels information displays on the side of the main display. These displays are used to demonstrate each player's score. The score numbers appear in 7x5 pixels format.

## Modules

Let's examine all the main modules that are presented on the scheme. We have a control module responsible for 8 columns(1), the submodule responsible for one column(2), a score counter(3), keyboard control(4), the module responsible for the information display(5), racket expansion module(6).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [[1]](#footnote-0) | [[2]](#footnote-1) | [[3]](#footnote-2) | [[4]](#footnote-3) | [[5]](#footnote-4) | [[6]](#footnote-5) |

### Main display implementation

Let's look at the abstract work of the main display. To each 8 columns of the display, we attach control modules to which information is transmitted from the processor and from the keyboard. Then these modules output information to the columns, and the desired image is created from the pixels on the display.

#### Eight-column control module

Each module receives the input X and Y coordinates of the ball, the Y coordinates of both rackets. It has two output blocks, one block outputs information to the display, the other one transmits it to the next same block.

Having considered this block from the inside, we will notice that it simply consists of 8 identical sub-blocks that are linearly connected.input X and Y coordinates of the ball, the Y coordinates of both rackets. It has two output blocks, one block outputs information to the display, the other one transmits it to the next same block.

Having considered this block from the inside, we will notice that it simply consists of 8 identical sub-blocks that are linearly connected.

|  |  |
| --- | --- |
| [[7]](#footnote-6) | [[8]](#footnote-7) |

#### One-column control module

Each module receives the X and Y coordinates of the ball, the Y coordinates of both rackets and the data for which column is responsible. Foremost, we decode from the Y coordinate of the ball into a format for the display. We compare the coordinates of this module with the X coordinate, if the coordinates match, we transmit a signal to a multiplexer that allows us to transmit this data further. Another decoder is responsible for the output of the racket, the coordinates of the module are compared with the coordinates where the racket should be, if they match, then the coordinates are decoded into a view for the display using a separate module, making a little shift to the left and right. After that, all data is transferred to the OR element and connected in the output to the display.

|  |  |
| --- | --- |
| [[9]](#footnote-8) | [[10]](#footnote-9) |

#### Bit Shift module

The bit shift module receives a 32-bit number where there is only one unit and expands this unit to the left and right.

|  |  |
| --- | --- |
| [[11]](#footnote-10) | [[12]](#footnote-11) |

### information displays

An information display control module is attached to each information display.

The module receives the players' score and five decoders select the fragment which is required for output. After this, it transmits the fragment to the information display.

#### Score counter

The score counter was implemented quite trivially. We have created a module that receives the coordinates of the ball, and the coordinates that are responsible for the "Gate" getting into which is our goal. After that, it compares these coordinates. If the coordinates of the ball coincide with the coordinates of the "Gate", then we send a signal to the counter, then the output of this module is attached to an information display.

|  |  |
| --- | --- |
| [[13]](#footnote-12) | [[14]](#footnote-13) |

### Game menu

In order to start the game, you need to click on the Start button. The game can be restarted with the Restart button. Exiting the game - Exit button: pressing it for second time leads player to the menu. The console menu shows the console logo. The logo is transmitted to the display as a set of constants.

|  |  |
| --- | --- |
| [[15]](#footnote-14) | [[16]](#footnote-15) |

## Game process

At the beginning of the game, a ball appears on the field, the score counters are turned on, which are zeroed. The player can control the right racket using the keyboard. When the ball hits the "Gate" of one of the players, the other receives a score and the counter adds 1.

When one of the counters reaches the value of 9, the game ends, the system restarts.

# SOFTWARE

## Introduction

The software consists of two parts. The first part is a code written in assembly language, the second one is a circuit in Logisim.

The main purpose of the software is to create a circuit that takes the coordinates of the user's racket as input and generates the location of the ball.

To create such a circuit, we use Harvard architecture.

|  |  |
| --- | --- |
| [[17]](#footnote-16) | [[18]](#footnote-17) |
|  | |
| [[19]](#footnote-18) | |

## 

## Code

The code handles situations when the ball bounces off the walls and moves at any angle. Since the field size is 32\*32, it is necessary to check the X and Y coordinates for 0 or 31. If equality is observed, then it is necessary to change the trajectory of movement.

The movement at an angle is carried out in accordance with the Bresenham algorithm. This algorithm is used if the angle of inclination is from 0 to 45 degrees, but if you mirror the coordinates (change the X and Y in the algorithm), then you can get a movement from 45 to 90 degrees. Thus, having estimated the tangent of the angle, we will use either the usual algorithm or a mirror one.

The code also implements a condition, namely change\_dir == 1, if true, you can change the trajectory of the ball along the X axis (dx = -dx). This is necessary in order for the ball to bounce off the rackets.

|  |  |
| --- | --- |
| [[20]](#footnote-19) | [[21]](#footnote-20) |
|  | |
| [[22]](#footnote-21) | |

## 

## Circuits in Logisim

### Store

This circuit shows a storage with a memory of 11 registers. These registers store the values that the processor will use.

Loading and unloading are carried out using a multiplexer, a demultiplexer and conditions: read (command: st), write (command: ld). Each register can be accessed at its address: f0 - f10. In addition, the value in the registers is updated when the conditions are met: A1-A11.

Updater is responsible for updating registers. A high signal is sent to the update input to the register, at the same time the data must be loaded into the register via the data wire.

### 

|  | |
| --- | --- |
| [[23]](#footnote-22) | |
|  | |
| [[24]](#footnote-23) | |

### 

### Bot

#### Ring and directory

The bot is the second participant in the game of tennis. When the ball moves to the right

(dx == 1), and the ball coordinate reaches 6 (x\_ball == 6), the calculation of the ball trajectory begins.

The trajectory is a straight line defined by parameters. Let's use the formula of the bundle of straight lines at the point (y0, x0):

y - y0 = (dy/dx) \* (x - x0)

x0 и y0 -current coordinates of the ball,

x и y - the coordinates of the racket.the x-coordinate is known and is equal to 28. it is necessary to find the y-coordinate

dy and dx are the angle parameters, they are in the storage under the names y\_max, x\_max. These parameters are used by the Bresenham algorithm.

For large values of dy, the location of the ball may be outside the playing field, so a scheme called a Ring is used. In this circuit, the actual coordinate of the ball is calculated taking into account bounces from the walls.

|  |
| --- |
| [[25]](#footnote-24) |

#### Random and Win or Lose

There are two ways to win a bot.

The first way is by hitting the ball with a racket. You can change the angle at which the ball flies, so that the bot may not have time to move.

The second method: a random number generator produces numbers in such a way that with a possibility of 75% the bot hits the ball, and with a possibility of 25% it misses by ±3 pixels to the side. Also, using a random number generator, it is determined whether the bot will make a mistake by +3 pixels and the ball will fly over the racket, or make a mistake by -3 pixels and the ball will fly under the racket.

|  |  |
| --- | --- |
| [[26]](#footnote-25) | [[27]](#footnote-26) |

### The ball bounces

#### Ball bounce conditions

The ball changes direction along the X axis when it is in front of the racket.

Necessary conditions for a rebound:

1. The ball flying into the bot's racket bounces if its dx > 0, the ball coordinate (x\_ball) = one less than the X-coordinate of the racket (x\_bot - 1),

as well as the Y-coordinate of the ball (y\_ball) = y\_bot (±1)

1. The ball flying into the user's racket bounces, if its dx < 0, the ball coordinate

(x\_ball) = one more than the X-coordinate of the racket (x\_user - 1),

and the Y-coordinate of the ball (y\_ball) = y\_user (±1)

|  |  |
| --- | --- |
| [[28]](#footnote-27) | |

#### Changing the angle of inclination or momentum of the racket

For a more interesting game of tennis, a change in the angle of inclination is implemented when the ball bounces off the racket.

The values are stored in the storage: the new racket coordinate and the old racket coordinate.

The old coordinate is updated once every 300 bars, and the new racket coordinate is updated once per bar. Thus, when moving the racket quickly, you can calculate the difference between the new and old coordinates, and the difference will be non-zero.

Let's use a formula from physics: p1 + p2 = p1’ + p2’ (vector)

| x: M \* Vxr + m\* Vxb = M \* Vxr’ + m\* Vxb’  Vxr - Vxr’ - difference in the coordinates of the racket divided by the conditional time.  M - weight of the racket; m - weight of the ball;  Vxb - old ball speed\_x  Vxb’ - new ball speed\_x  result: |Vxr| = |Vxr’|  y: M \* Vyr + m\* Vyb = M \* Vyr’ + m\* Vyb’  M(Vyr - Vyr’) = m(Vyb’ - Vyb)  Vyb’ = (M/m)\*(dV) + Vyb |
| --- |

Thus, we get the formulas:

V\_y’ = k\*dV + V\_y

|Vxr| = |Vxr’|

V\_x’ and V\_y’ are error\_max\_X and error\_max\_Y for the Bresenham algorithm and are written to the store as y\_max and x\_max.

The new ball speed is calculated when the ball is in front of the racket. When the speed calculations are completed and the processor starts calculating the trajectory of the ball using error\_max: the speed components in the registers are updated.

The circuit shows the arithmetic modules that calculate the speed parameter: Vy, as well as the speed limits (v\_min = 1, v\_max = 2), using constants and multiplexers. Restrictions are necessary in order to better visualize the movement of the ball. The mass ratio coefficient is represented by a constant and it is equal to ¼. This value is also chosen for better visualization and it directly depends on the refresh rate of the old racket speed.

|  |
| --- |
| [[29]](#footnote-28) |

# Conclusion

We have successfully completed our project “TV-Tennis” according to the technical task. We used different sources of information - most of all - assembly language documentation и Logisim.

# Bibliography

* Donovan, Tristan (2010-04-20). Replay: The History of Video Games. Yellow Ant. pp. 1–9. ISBN 978-0-9565072-0-4.
* Bresenham, J. E. (1965). "Algorithm for computer control of a digital plotter" (PDF). IBM Systems Journal. 4 (1): 25–30. doi:10.1147/sj.41.0025.
* Abrash, Michael (1997). Michael Abrash's graphics programming black book. Albany, NY: Coriolis. pp. 654–678. ISBN 978-1-57610-174-2.

1. control module responsible for 8 columns [↑](#footnote-ref-0)
2. submodule responsible for one column [↑](#footnote-ref-1)
3. Module score counter [↑](#footnote-ref-2)
4. keyboard control [↑](#footnote-ref-3)
5. the module responsible for the information display [↑](#footnote-ref-4)
6. racket expansion module [↑](#footnote-ref-5)
7. Eight-column control module [↑](#footnote-ref-6)
8. Eight-column control module inside [↑](#footnote-ref-7)
9. One-column control module [↑](#footnote-ref-8)
10. One-column control module inside [↑](#footnote-ref-9)
11. bit shift module [↑](#footnote-ref-10)
12. bit shift module inside [↑](#footnote-ref-11)
13. the point counter module [↑](#footnote-ref-12)
14. The score counter is inside [↑](#footnote-ref-13)
15. In this diagram, there is a choice between the game and the logo. [↑](#footnote-ref-14)
16. logo [↑](#footnote-ref-15)
17. Circuit that takes the coordinates of the user's racket as input and generates the location of the ball [↑](#footnote-ref-16)
18. Harvard Architecture (1) [↑](#footnote-ref-17)
19. Harvard Architecture (2) [↑](#footnote-ref-18)
20. Bresenham algorithm [↑](#footnote-ref-19)
21. Condition: change\_dir [↑](#footnote-ref-20)
22. The code when executing which the ball bounces off the walls [↑](#footnote-ref-21)
23. Storage and methods of working with it [↑](#footnote-ref-22)
24. Updater of store [↑](#footnote-ref-23)
25. Circuit Ring - the actual coordinate of the ball is calculated taking into account bounces from the walls. [↑](#footnote-ref-24)
26. Will the bot hit the ball with a racket or not [↑](#footnote-ref-25)
27. Should the calculated coordinate for the bot be shifted [↑](#footnote-ref-26)
28. Conditions for ball and racket [↑](#footnote-ref-27)
29. This circuit calculates the angle of inclination of the trajectory of the ball [↑](#footnote-ref-28)