MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION

FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION

"NOVOSIBIRSK NATIONAL RESEARCH UNIVERSITY

STATE UNIVERSITY"   
(NOVOSIBIRSK STATE UNIVERSITY, NSU)

15.03.06 - Mechatronics and Robotics

Focus (profile): Artificial Intelligence

|  |  |
| --- | --- |
| Job topic: | **EXPLANATORY NOTE**  **‘SPACE INVADERS’** |

Semenov Arseniy, 23931

Izhdeneva Vlada, 23931

Maksimov Egor, 23931

Novosibirsk

2024

TABLE OF CONTENTS[PAGEREF \_Toc158500730 \h3](#_Toc158500730)

[1 Introduction 3](#_Toc166503628)

[2 Purpose and area of application 3](#_Toc166503629)

[3 Functional characteristics 3](#_Toc166503630)

[4 Technical characteristics 3](#_Toc166503631)

[5 Conclusion 6](#_Toc166503632)

[6 Sources used in developing 6](#_Toc166503633)

# Introduction

Computer games in the form in which we have seen them over the past 20 years have come to the form we are familiar with after many years of evolution. Since the middle of the last century, people began to use computers to create various types of entertainment programs, and later, based on them, arcade machines began to appear. The arcade games of that time were the reason that piqued our interest, and we decided to recreate one of them as part of this group project.

Created in 1978, Space Invaders has become a true classic over the decades of its existence. Released in Japan, the arcade game managed to win fans all over the world and, having become one of the founders of the Shoot’em up genre, has not lost its popularity to this day, being reborn in various kinds of remakes. Our team was interested in repeating the experience of creating this game, as well as understanding its internal structure and principles of its operation.

The essence of the game is to prevent alien ships from reaching the earth, destroying them as they approach. To do this, the player controls of a turret, which, when a key is pressed, fires a projectile that destroys the alien spaceship when hit. However, the aliens also drop bombs in an attempt to destroy the player's turret. The goal of the game, as you might guess, is to destroy waves of enemy ships, and defeat is considered either when at least one of these ships reaches the surface of the earth, or when the player’s turret is destroyed.

# Purpose and area of application

The purpose of creating this program is to obtain a working copy of the above-mentioned game, while maintaining the main part of its functionality. The scope of application lies not only in the obvious opportunity to use a ready-made program for its intended purpose (i.e., to play), but also in the ability for an outsider to clearly understand what the circuitry is.

# Functional characteristics

Player can:

* Move to the right and to the left.
* Shoot enemies.
* Pause the game.
* Restart the game.
* Win or lose the game.

Enemies can:

* Shoot the player.
* Move to the bottom or to the left or to the right.

# Technical characteristics

Player can move to the left and to the right by pressing buttons A and D. When player press the button, the code of this button provides to circuit where this signal compares with codes of other buttons to find the necessary move. If code of pressed button doesn’t define in the list of codes, the circuit skip this signal and clear the keyboard. Else the signal of move provides to shift and shift can increment the counter or decrement it. If shift decrements the counter, player moves to the left, else to the right.

|  |
| --- |
|  |
| Picture 1 - Player movement |

Player also can shoot enemies by pressing the button. When player press the button, the signal provides to the circuit with the x-coordinate of player. The bullet appears at the screen above the player’s ship. Shift registers are using for bullet movement.

|  |
| --- |
|  |
| Picture 2 - Player shooting |

All the moves x-coordinates and y-coordinates provide into circuit. Into the circuit x-coordinates of bullet and enemy compare and y-coordinates too. If both of them are equal, the string with enemy redraws.

|  |
| --- |
|  |
| Picture 3 - Collision |

All of the clocks enemies should move. Into the circuit each register provides its value to the next register.

|  |
| --- |
|  |
| Picture 4 - Enemy movement |

Into this circuit processes the direction of enemies. When enemies moves to the left, the contact which defines it become 0, else 1.

|  |
| --- |
|  |
| Picture 5 - Shifting |

This project uses CdM-8 and a Harvard architecture around the CPU. Multiplexor to read data from registers and demultiplexor to write data. The program have a line of enemies and a random-generated number. It’s choses enemy, which’ll drop a bomb based on random-generated number from enemies that’re still alive on the bottom line of enemies. Output is a 8-bit number – coordinate of a bomb. It uses only first 5 bits of the number, because 5 bit is enough to encode any number between 0 and 31. By using a decoder we’re converting it into a 32-bit line, that has all zeros, except of 1 bit – coordinate where the bomb should be. Then it moves it down by demultiplexor.

|  |
| --- |
|  |
| Picture 6 - Using of Hardard architecture in the project |

# Conclusion

First of all the main part of original game functionality like movementrs or shooting was implemented in the project. Secondly objects like enemies or player’s ship have different colors.

During the development of project was skipped some important things like score or number of lives.

# Sources used in developing

1. [b] Computing platforms / A.Shafarenko, S.P.Hunt / 2015
2. [e] Logisim manual / [<http://cburch.com/logisim/index_ru.html>]
3. [e] CdM-8 instruction set / [http://ccfit.nsu.ru/~fat/Platforms/instruction-set.pdf]