MINISTRY OF EDUCATION AND SCIENCE OF THE RUSSIAN FEDERATION

NOVOSIBIRSK STATE UNIVERSITY  
(NSU)

Higher College of Informatics

Department of intelligent systems of thermophysics

Field of study 15.03.06 Deep robotics

Specialty: Mechatronics and robotics

**TERM PAPER**

**Bespalov Sergey Vyacheslavovich**

**Grishchenko Alexander Mikhailovich  
Solopov Ilya Ruslanovich**

The theme of the paper:

**THE «RUSH Space» GAME**

Novosibirsk, 2022

CONTENTS

[INTRODUCTION 3](#_Toc103514378)

[1 PROBLEM DEFINITION 4](#_Toc103514379)

[2 ANALOGUES 5](#_Toc103514380)

[3 HARDWARE 7](#_Toc103514381)

[4 SOFTWARE 8](#_Toc103514382)

[CONCLUSION 9](#_Toc103514383)

[REFERENCES 10](#_Toc103514384)

[APPENDIX 11](#_Toc103514385)

INTRODUCTION

По заданию к курсовому проекту по дисциплине «Digital platforms» нужно было спроектировать и создать игру в жанре Shoot 'em up на электронных схемах, использовав процессор CdM-8 и его ассемблерный язык.

Shoot 'em up – это вид компьютерных игр, в котором управляемый игроком персонаж чаще всего представлен в виде космического корабля или другого транспортного средства, основная цель которого – победить множество врагов, используя стрельбу. Врагами в таких играх, как правило, выступают различные инопланетяне или монстры, атакующие игрока, стреляя в него или как-то иначе. Традиционно в таких шутерах используется вид сверху или сбоку, а для успешного прохождения важна хорошая реакция, чтобы уклоняться от вражеского огня

В соответствие с поставленной целью необходимо было решить следующие задачи:

* Изучить примеры игр жанра Shoot 'em up (аналоги);
* Изучить и проанализировать информацию о процессоре, его возможностях, командах и инструкциях;
* Определить функциональные требования.

1 PROBLEM DEFINITION

The purpose of this paper term is the design and creation of a Shoot 'em up game based on an electrical circuit with an 8-bit CdM-8 processor included in it.

Ниже представлены функциональные требования:

1. Управляемое движение игрока;
2. Неуправляемое движение противников;
3. Стрельба игрока;
4. Стрельба монстров;
5. Уничтожение монстров;
6. Управление с клавиатуры (движение игрока, стрельба игрока);
7. Возможность выигрыша;
8. Возможность проигрыша.

Методы решения поставленных задач…

2 ANALOGUES

During project creation, we explored Shoot 'em up games. Let's consider some of them.

1. Japanese arcade game "Space Invaders", released in 1978. It can be called one of the first in this genre. This game has all the functional requirements specified in the previous paragraph and additional ones, for example, a life counter and the number of points scored, sound effects, an infinite number of "waves" of enemies, protection in the form of "bunkers" from alien shots.

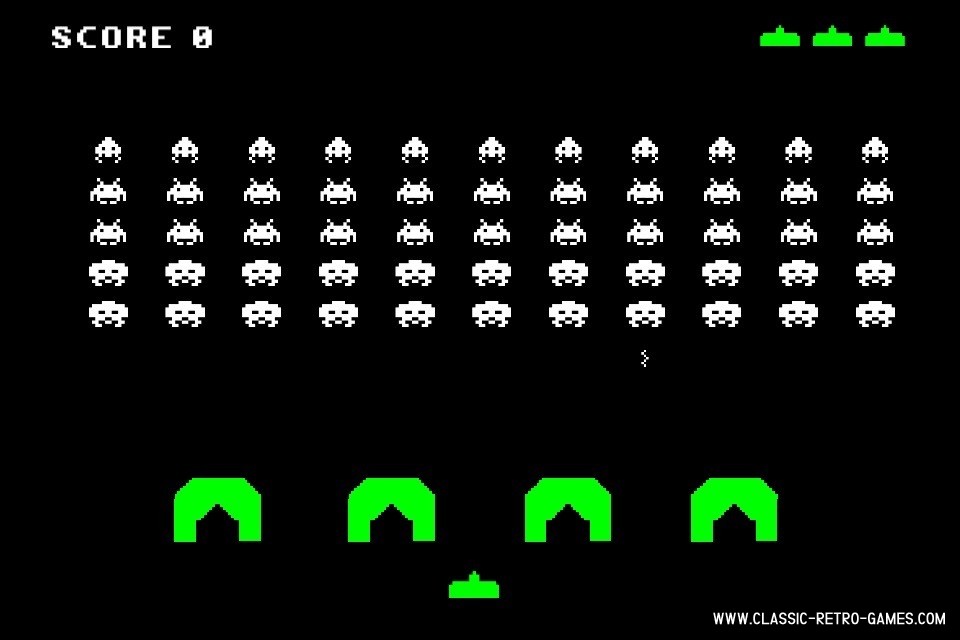


Figure 1 – the gameplay screenshot of "Space Invaders"

1. Another Japanese game released for arcade machines in 1981 is «Galaga». It can be called an improved and more modern version of the previous game. New game mechanics were created in «Galaga», such as an alien tractor beam that can take away the player's control for a short period of time and one life, monsters dive at the player in various trajectories. It is also one of the first games with color RGB graphics.

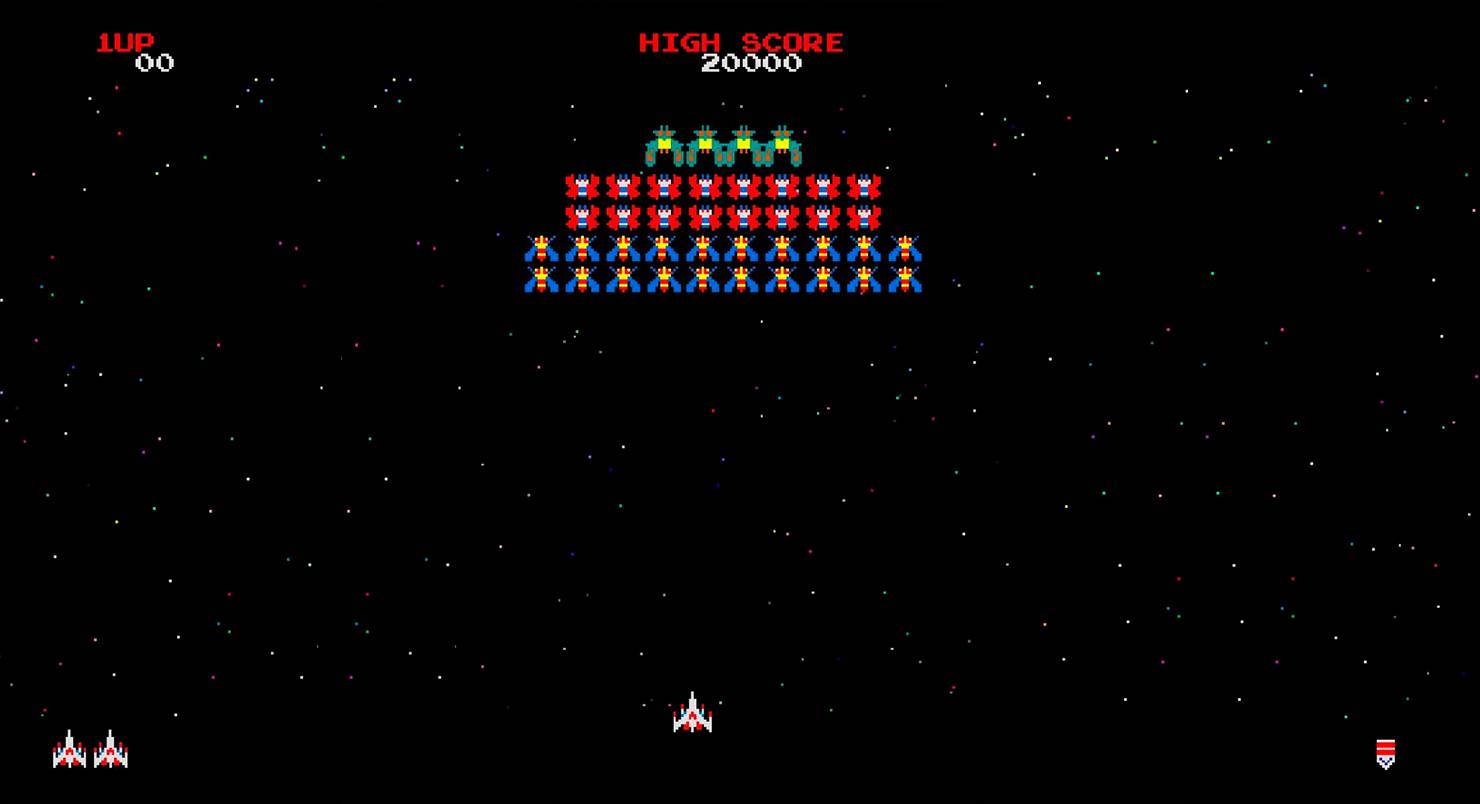


Figure 2 – the gameplay screenshot of "Galaga"

Изучив и проанализировав данные примеры, нашей командой было принято решение создавать проект, основываясь на дизайне и наполнении игры «Space Invaders». Из-за ограниченного количества времени, выделенного на разработку нам нужен был не очень сложный и ресурсозатратный вариант, который было бы возможно реализовать, используя CdM-8 и наши знания в области схемотехники.

3 HARDWARE

The hardware part of our project consists of logical electrical circuits created in the Logisim program, which allows you to model and edit them using a convenient graphical interface. Let’s consider our developments.

4 SOFTWARE

The software part of our project was made using the assembly language of the CdM-8 processor in the CocoIDE integrated development environment, created specifically for developing the code executed by this processor. Let’s consider the algorithms.

**Wall destruction**

This is an algorithm that implements the initialization and destruction of protective walls when monster bullets hit them. The input data is two four-byte arrays: a string with the current state of the walls and a null string with one hit bit equal to 1. The output data is a string with the state of the walls after a bullet hit or miss.

The first step initializes the registers located at addresses F0-F3 with the value 3C (00111100 in binary). This constant is the four-pixel wall displayed on the screen. At the second step, the input data is loaded into the general-purpose registers, the counter is initialized. Then, in a loop, while the counter is greater than zero, an exclusive "or" operation is performed between the corresponding bytes of the input arrays, and the result is stored in the register into which the walls were loaded. The result is a bit string with states after hitting or missing bullets. At the end of the loop, there is a transition to the second step of the algorithm. Figure \_ shows a block diagram of the algorithm for initializing and destroying walls.

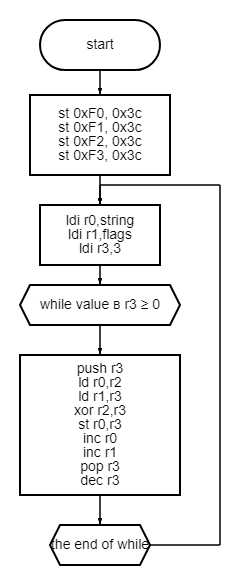


Figure \_ - a block diagram of the algorithm for initializing and destroying walls

**Monsters shooting**

This is an algorithm that implements the calculation of the coordinate (column number) in which the monster will shoot. The input data is two one-byte numbers: the number of the shooting alien, (from 0 to 3) and the offset value of the monster (-1 or 1). The output data is a one-byte column number.

The first step is to load the constants 8 and 0 and the alien number to be used as a counter into the general-purpose registers. Then the loop will calculate a value equal to the number of the shooting monster, multiplied by 8. After that, the offset value and the constant 4 are added to it, and the result (the resulting column number) is stored in the register. At the end of the loop, there is a transition to the first step of the algorithm. Figure \_ shows a block diagram of the algorithm for calculating the number of the column in which the shot will be fired.

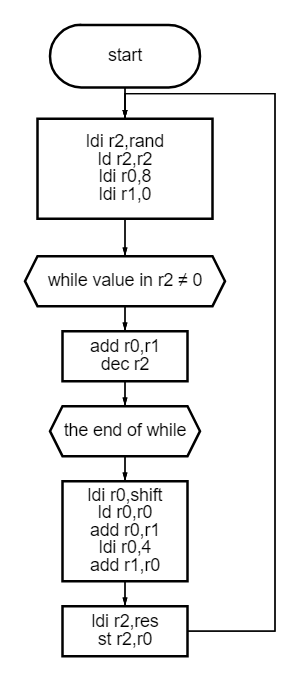


Figure \_ - a block diagram of the algorithm for calculating the number of the column in which the shot will be fired

**Hit**

this is an algorithm that implements the initialization of the state of the aliens and the processing of the player's bullet hitting them. The input data is a byte that stores the state of the aliens (if the bit is 1, then the monster is alive, if 0, then not) and a null string with one hit bit equal to 1. The output data is a byte that stores the new states of the enemies.

At the first step, the monster status byte is initialized with the value FF (11111111 in binary). The second step loads the zero string with one hit bit equal to 1 and the constant 0 into the general-purpose registers, the counter is initialized. Then the loop looks for a non-zero byte in the string and calculates the number of the alien that was hit. After that, by bitwise shifts to the left, a one-byte mask is calculated, where the only bit equal to 1 has an index equal to the number of the alien that is in the bottom row. The next step is to compare this mask and the byte that stores the states of the enemies by performing a bitwise "and" operation. If the result is zero, then there is no monster in the bottom row, so 4 more bitwise shifts to the left are performed to check the top row. Next, an exclusive “or” operation is performed between the new mask and the status byte, and the result is stored in a register. There is a transition to the second step of the algorithm. Figures \_ and \_ show parts of the block diagram of the algorithm for initializing the state of aliens and processing hitting them.

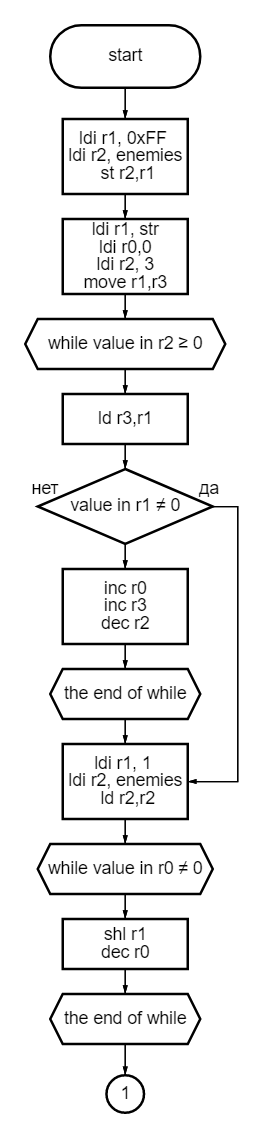


Figure \_ - the first part of the block diagram of the algorithm for initializing the state of aliens and processing hitting them

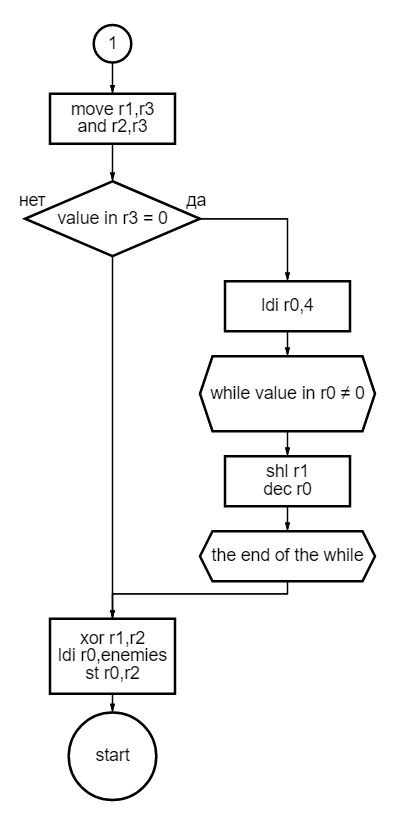


Figure \_ - the second part of the block diagram of the algorithm for initializing the state of aliens and processing hitting them

Thus, the developed game is the result of the interconnected work of the hardware and software parts of the project described earlier.

CONCLUSION

As a result of the work done, we managed to create the game "Rush space" in the Shoot 'em up genre (see Appendix A for the user manual). It is implemented using electronic circuits in which we have used a CdM-8 processor that executes the codes we have written. The problems set before us at the beginning of the project were successfully completed. We were able to implement all the indicated functional requirements: movement and shooting of the player and opponents, control from the keyboard buttons, the possibility of winning and losing in the game. Creating the project, we got knowledge in the field of creating electrical circuits, working with a processor built on the Harvard architecture and its programming, writing project documentation and experience in working in a team.

REFERENCES

1. Computing platforms / A.Shafarenko, S.P.Hunt. – 2015.

APPENDIX

APPENDIX A

**User manual**

When the game is opened, the user sees an electrical circuit stylized as an arcade machine. It has two buttons and a keyboard with which the player can control the game. Figure A.1 shows a screenshot of the initial screen of the game.

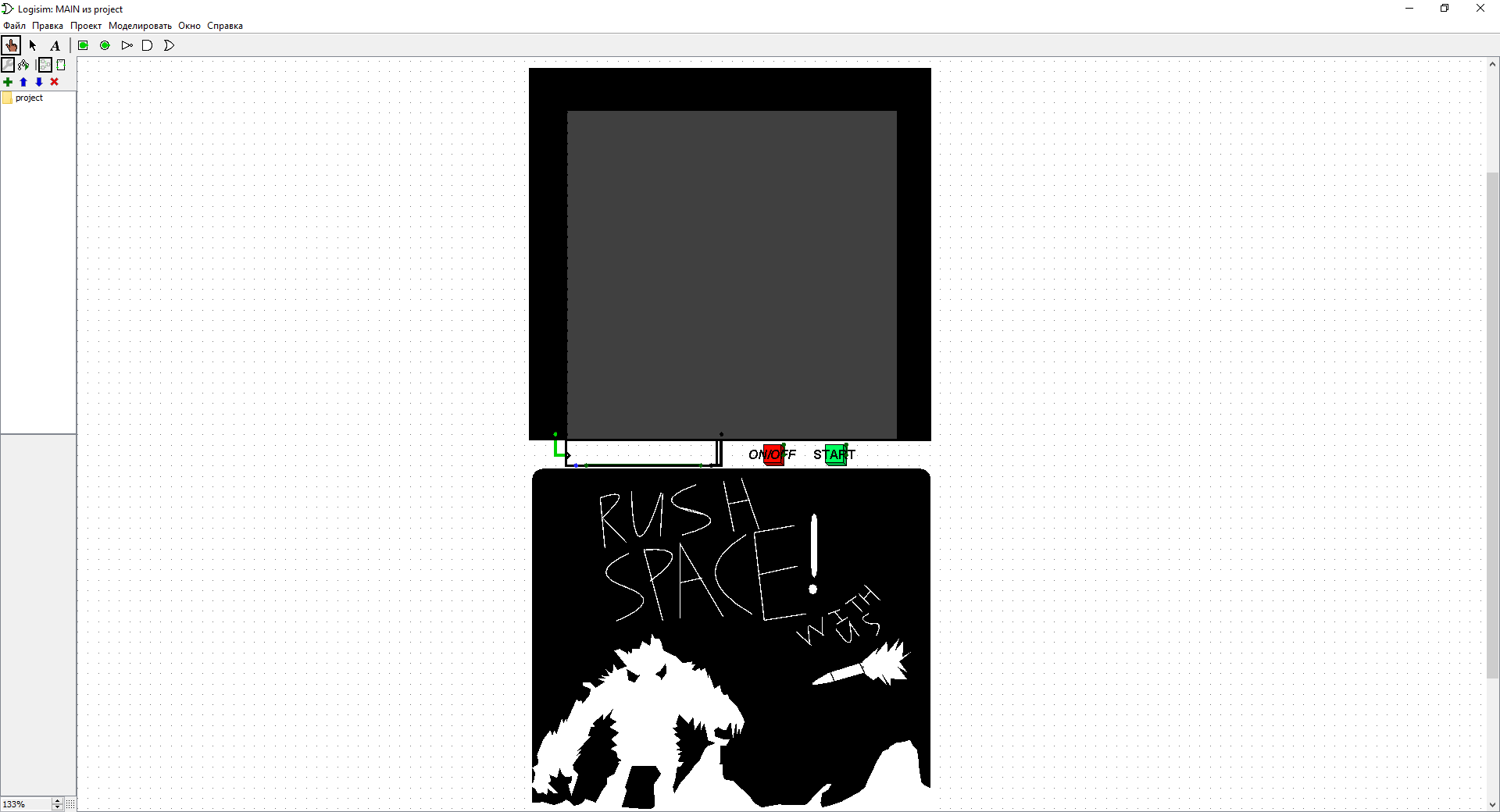


Figure A.1 – screenshot of the game's initial screen

To turn on the game, the user needs to press the "ON / OFF" button. After that, the player will see a message on the screen that he needs to press the “START” button to start the gameplay. Figure A.2 is a screenshot of the screen with this message.

Изображение выглядит как текст, часы

Автоматически созданное описание

Figure A.2 – screenshot of the screen asking user to press the "START" button

When user click on the button that starts the game, the field is displayed on the screen and the game process begins. The player controls the cannon, located at the bottom of the screen, in a horizontal plane using the keyboard, namely the keys “w”, “a”, “s”, “d”. Also on the field, the user sees 4 static walls, and 8 monsters moving uncontrollably horizontally and vertically. Figure A.3 shows a screenshot of the playing field at the start of the game.

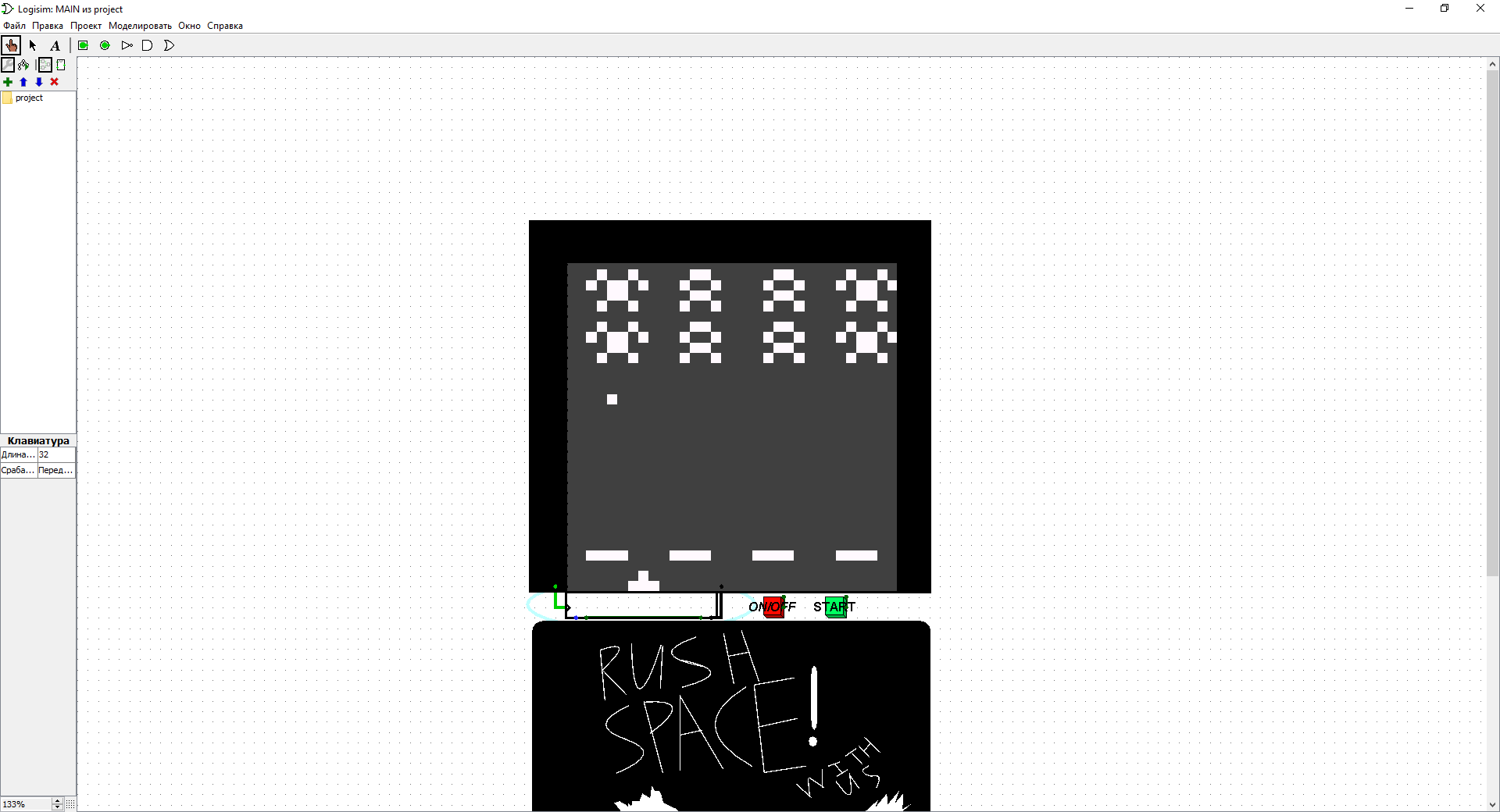


Figure A.3 – screenshot of the playing field at the start of the game

The user can also shoot using the "SPACE" key on the keyboard. When pressed, a bullet flies out of the cannon, flying vertically upwards until it hits an alien, or before it goes out of the playing field. Figure A.4 shows a screenshot of the playing field at the time of the shot.

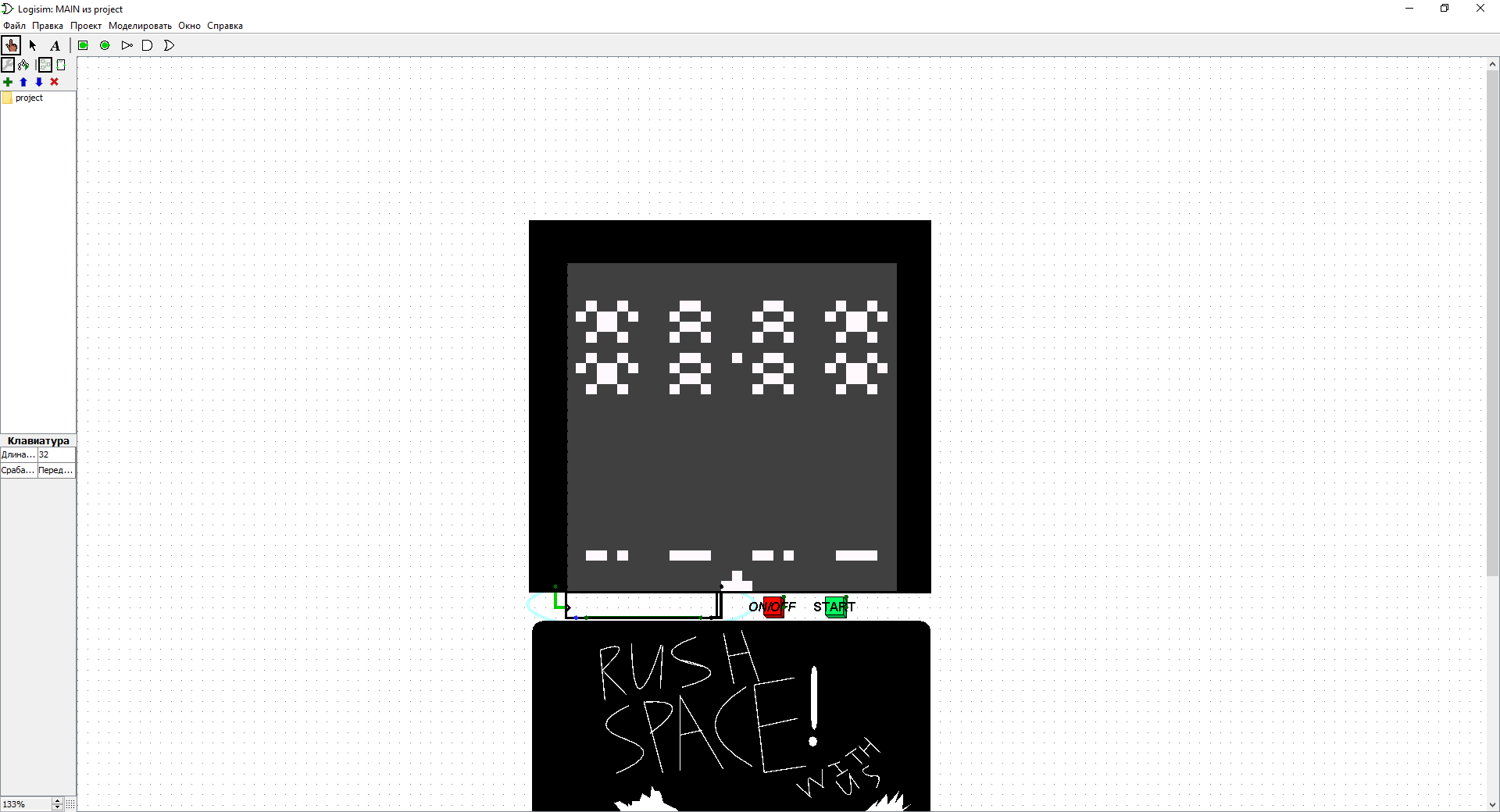


Figure A.4 – screenshot of the playing field at the time of the player’s shot

Monsters also shoot. Every few seconds, a random alien fires a bullet that travels vertically downwards until it hits the player, either against the wall or before it goes out of bounds of the playing field. The wall protecting the player is partially destroyed when a bullet hits it. Each "protection" can withstand 4 hits before it completely breaks down. It is important to note that the wall can be attacked not only by aliens, but also by the player himself. Figure A.5 is a screenshot of the playing field with partially destroyed walls.

Изображение выглядит как текст, часы, снимок экрана

Автоматически созданное описание

Figure A.5 – screenshot of the playing field with partially destroyed walls

When an alien bullet hits the cannon, the user loses and a game over message shows on the screen. Figure A.6 is a screenshot of the game over message.

Изображение выглядит как текст, часы, снимок экрана

Автоматически созданное описание

Figure A.6– screenshot of the screen with a message about the game over

The player can lose under another condition. Every few seconds, all the monsters move closer to the cannon. If at least one alien can survive and get the walls, then the game will be end and an endgame message will show on the screen. Figure A.7 is a screenshot of the playing field in a state close to losing.

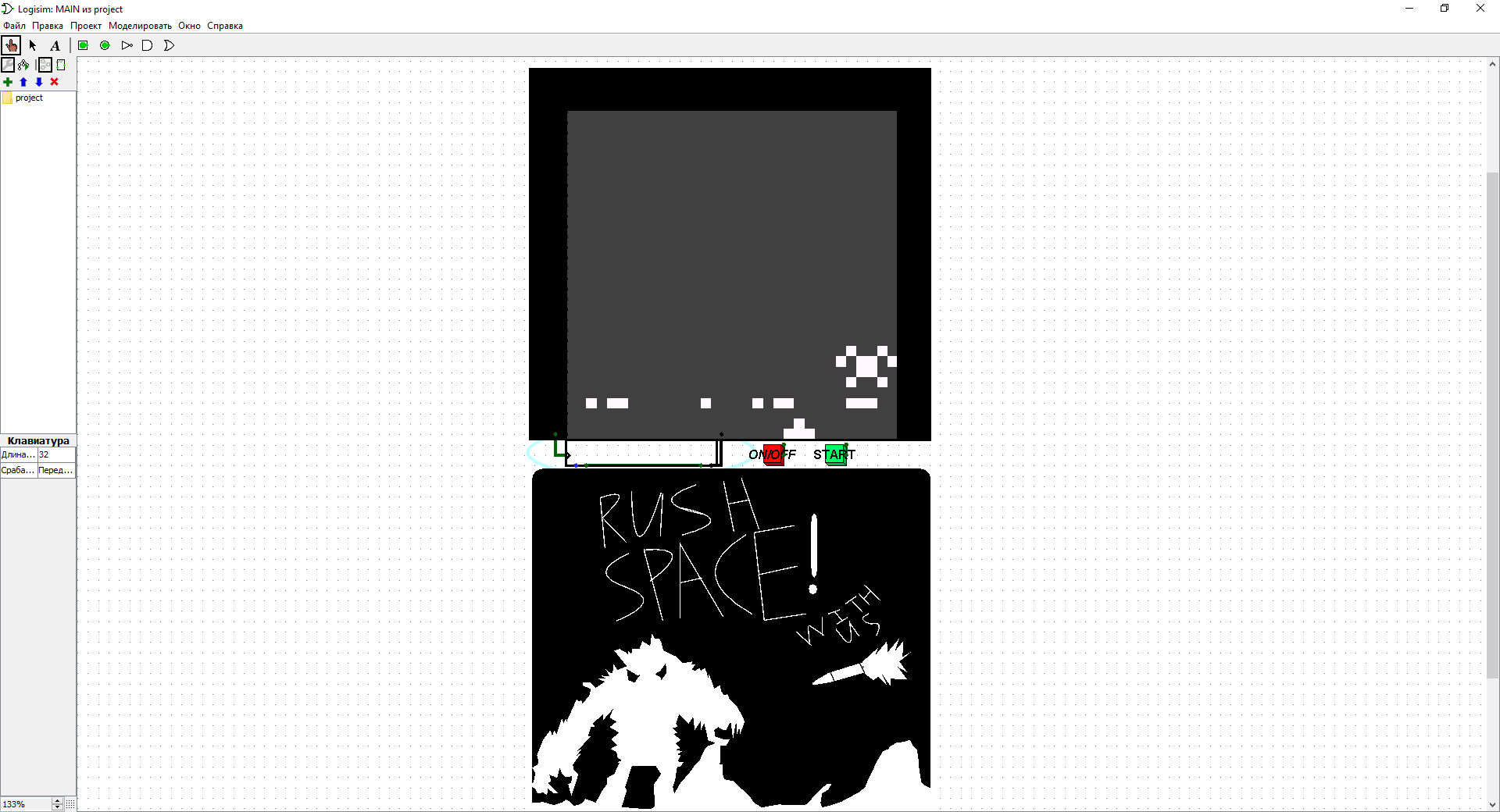


Figure A.7 – screenshot of the playing field in a state close to losing.

The player must destroy all the monsters to win. Figure A.8 shows a screenshot of the playing field after hitting most aliens.

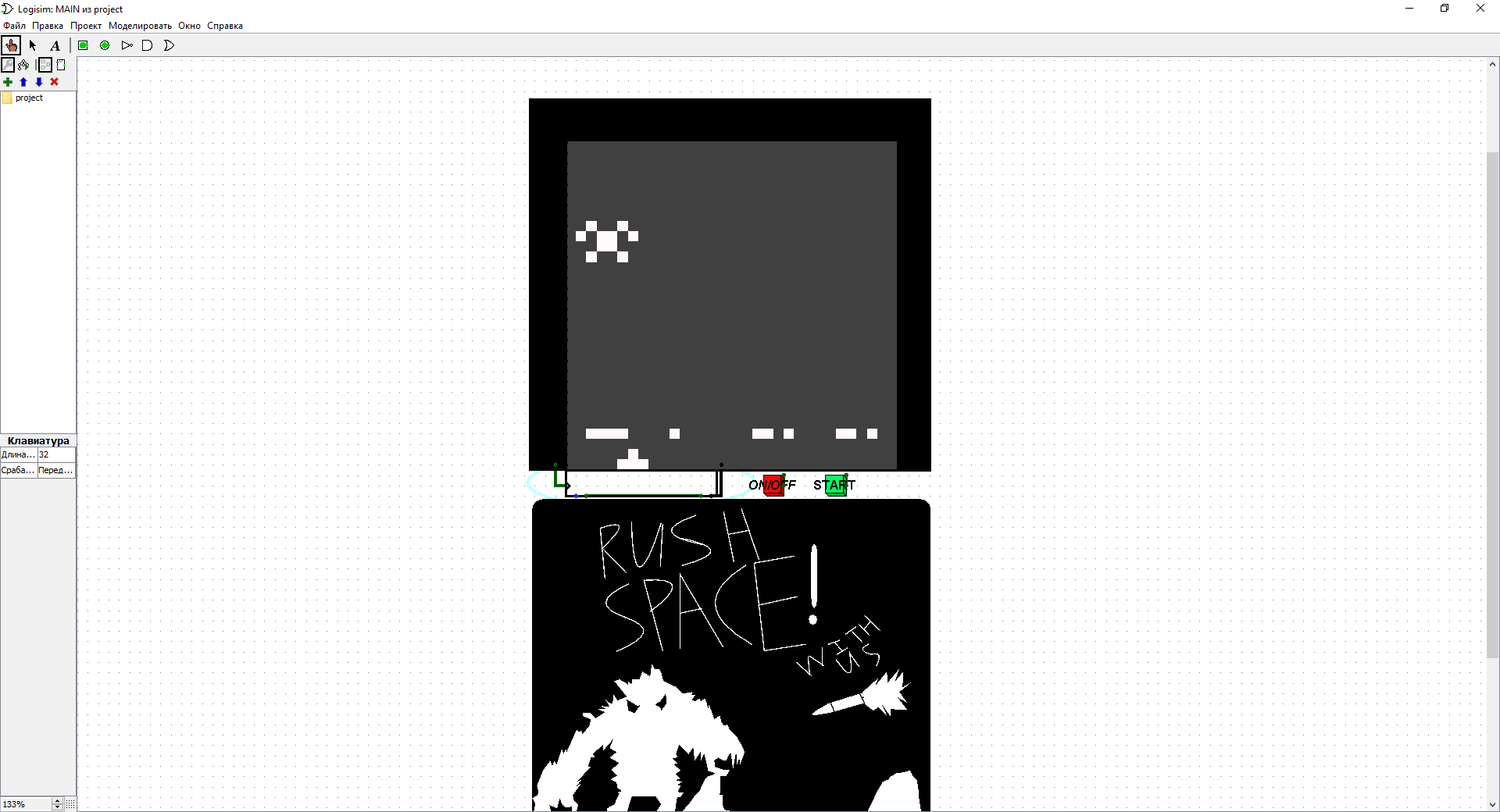


Figure A.8 – screenshot of the playing field after hitting most aliens

After the destruction of all monsters, a message about the victory of the player will show on the screen. Figure A.9 is a screenshot of the win message.

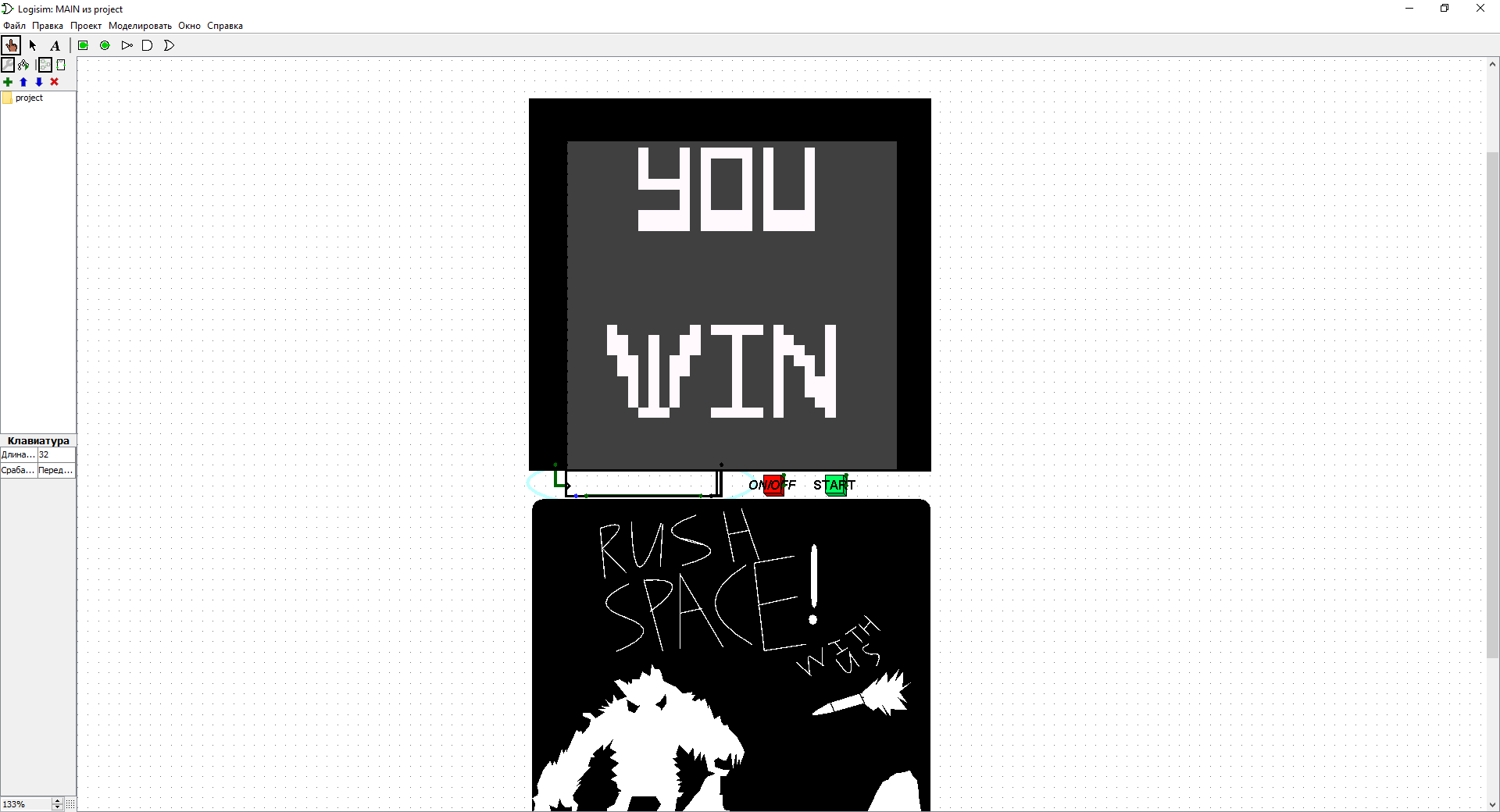


Figure A.9 – screenshot of the win message