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

SPECIFICATION

Job topic: **‘TETRIS’**

Egor Kravchenko, 23930
Valeriia Zaichikova, 23930
Dmitry Zlobin, 23930

Novosibirsk
2024

TABLE OF CONTENTS**ОШИБКА! ЗАКЛАДКА НЕ ОПРЕДЕЛЕНА.**

1 TERMS AND ABBREVIATIONS

CdM8 Mark5	Coco de Mer 8 Mark5 – the processor which was made in Logisim
LED	Light-emitting diode
Tetromino (tetrominoes)	Figure for the game of Tetris, geometric shape composed of four squares, connected orthogonally
Randomizer	A device to generate a value from a range randomly
UI	User interface

2 DEVELOPMENT REQUIREMENTS

- The project of the game "Tetris" should work according to the standard canons of this video game:

  The game Tetris should have the ***tetrominoes*** (the Tetris figures) which shall fall from the top of the screen. For each full line made of these figures, the player gets 100 points, 300 for two, 700 for three and 1500 for four. If the next piece doesn't fit, the game is over.
- For the better result, the game also should have:
  1) The Randomizer: we need to create a random number generator using assembly language;
  2) The collision: we also need to implement the "***physics of contact***" between objects (i.e. the figures themselves) and the walls and each other. We need to calculate the collision for a piece's potential rotation so that it (or parts of it) does not extend beyond the boundaries of the screen or other pieces. The work of the collision shall help the games' figures behave more naturally as in the real Tetris;
- Also, most of our project will be on the Logisim platform.

2.1 Functional requirements

- The main objective:

We need to write assembler code for our project's **randomiser**, which will be used to generate random shapes for the game.

- The possible problems with it:

The main problem in creating a randomiser is to ensure an even distribution of tiles during the game. This is to avoid periods of "*drought*" or "*flood*". A "drought" refers to a long period when a particular piece does not fall, and a "flood" refers to moments when the same piece falls several times in a row.

Our tasks therefore include:

- Developing a method or an algorithm that will prevent a character from being absent for a long period of time or from appearing many times in a row.
- Improving the random number generator so that the distribution of numbers is predictable and fair, while retaining an element of randomness.

- The possible ways to solve these problems:

1) Explore the possibility of implementing a "bag" system, where a list of all the pieces is placed in a "bag" and then the pieces are removed at random, one by one, until the "bag" is empty. When the bag is empty, it is filled again and the process is repeated;

2) Determine the optimum size of "bag" for our system; for example, a 7-bag containing each of the seven tetrominoes;

2.2 Technical requirements

- The objectives:

We need to implement a UI circuit and controlling subcircuits.

List of mechanics:

- Sprite rendering.

- Movement: to the right, to the left, rotate with the buttons, and move down automatically.
- Rotation itself based on super rotation system, current standard of Tetris games.
- Collision: border, bottom collision, collision with the background figures, rotation collision.
- Score counter with a proper display of the current result.
- Pause, restart, game over.

➢ The possible problems with it:
1. Shift of the multiple rows on the main matrix (when 1 to 4 rows are full) is a complex game mechanic, hard to implement correctly.
2. Rotation collision based on super rotation system is pretty complex and can cause errors and slowdowns.

➢ The possible ways to solve these problems:
1. Efficient field rendering with convenient access to any of the rows can help with this problem.
2. We shall implement a simpler version of it. Basically, if the rotation of figure will cause merging with background or field edges, it has to be disabled. Wall kicks need a much more computing power, than we have.

3 DOCUMENTATION REQUIREMENTS

For the documentation of our project, we shall use only an Explanatory Note, according to the requirements of the Review Board of our Digital Platforms 2024 lecturers. It will be required to have the following paragraphs explaining the components of our project and the things that helped us to create it:

1) TERMS AND ABBREVIATIONS
2) INTRODUCTION
3) PURPOSE AND SCOPE
4) FUNCTIONAL CHARACTERISTICS
5) TECHNICAL CHARACTERISTICS
6) CONCLUSION
7) SOURCES USED IN DEVELOPMENT

4 STAGES OF DEVELOPMENT

The creation of the finished project will take place in several stages, namely:

1) ***Creation of the basic Tetris playing field:*** a circuit will be assembled, consisting of two display panels with buttons for moving and rotating the pieces, as well as "pause" and "restart" buttons. The larger display will show the game itself, and the smaller one, in the upper left corner, will show the sprites of the next new pieces.

2) ***Realisation of sprite output and control schemes:*** in a specific block of the Logisim platform, sub-circuits for outputting game sprites will be implemented, as well as systems for controlling these sprites in the game itself.

3) ***Figure sprite rendering scheme:*** in this scheme, after the previous two steps, we will start rendering the figure sprites of our Tetris. In this step we will also implement an algorithm to rotate the shapes based on these sprites.

4) ***Realisation of the collision in the game:*** In the game, we will create a collision system ("**system of contact**") of the figures with the borders of the screen, as well as with other figures. It will be necessary to make the collision so that when you move or rotate a *tetromino* it does not overlap with other tetrominoes and does not extend its parts beyond the top, bottom and sides of the screen. This system will help to clearly collect complete lines of Tetris pieces and let the player collect the necessary points.

5) ***Randomisation system:*** as mentioned above, our game will need a randomisation system for the pieces, which will be built in low-level assembly language on CdM8. After these steps, we will continue to build this code.

6***) Linking the assembler randomiser to the project circuits in Logisim:*** we will use the CdM8 Mark5 processor assembled on the Harvard system to communicate our random generator. All the randomness of our Tetris figures will be tied through it.

7) ***Circuit block for player scoring:*** after the main stages of the project, we will finally build a circuit block on Logisim. In it we will write a visualisation of the gameplay counter.

After all these steps, the project will be ready to be submitted to our board of examiners.