Final Project: Tetris

Ashenafee Mandefro (1007071151)
Li Quan Soh (1003357565)

April 3, 2024

About:

Tetris is a classic puzzle game where players manipulate falling tetromino shapes to create complete horizontal lines, which disappear and earn points. Players can rotate and move the falling pieces to fit them into gaps, aiming to prevent the stack from reaching the top of the screen.

For this specific project, we have implemented a variation of it using MIPS Assembly Language. We have decided to only implement the I-block.

How to Play/ Controls:

W: Rotate blocks

A: Shift block left

S: Shift block down

D: Shift block right

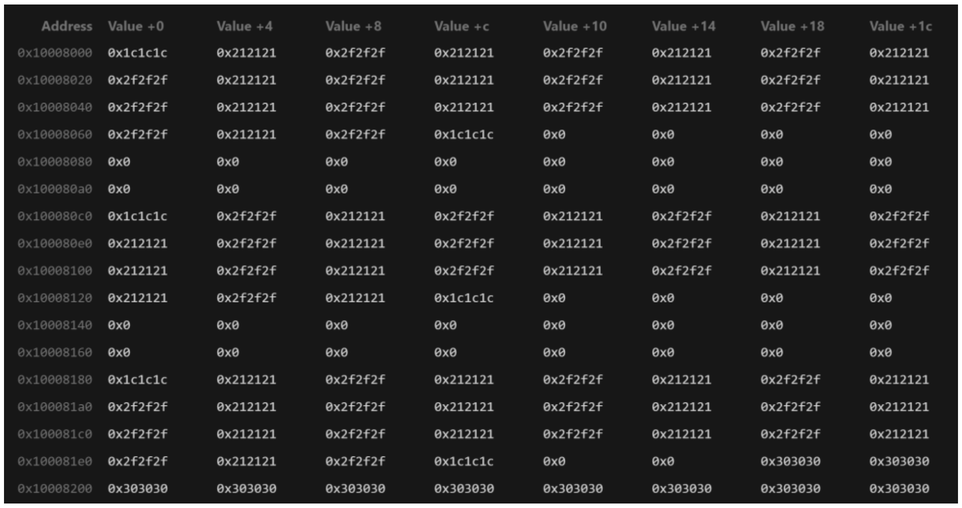
R: Restart/ Reset the game

Q: Quit the game (X Screen)

Milestones and Features Achieved

Milestone 1: Completed

- Static tetris game container with checkerboard background
- Single piece of I-block
- Snapshot of the memory when the game is running



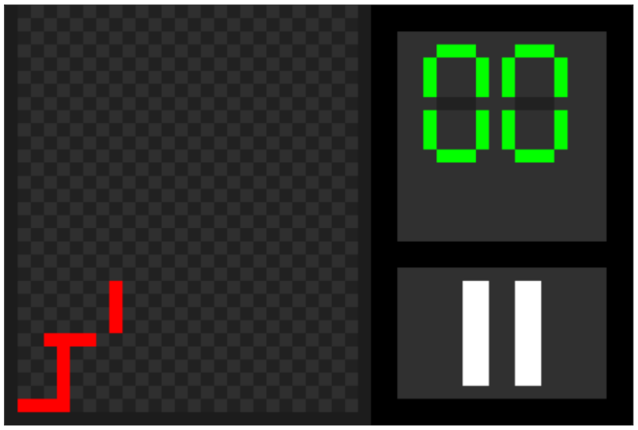
```
  Address   Value +0  Value +4  Value +8  Value +c  Value +10 Value +14 Value +18 Value +1c
0x10008000  0x1c1c1c  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121
0x10008020  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121
0x10008040  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121
0x10008060  0x2f2f2f  0x212121  0x2f2f2f  0x1c1c1c  0x0       0x0       0x0       0x0
0x10008080  0x0       0x0       0x0       0x0       0x0       0x0       0x0       0x0
0x100080a0  0x0       0x0       0x0       0x0       0x0       0x0       0x0       0x0
0x100080c0  0x1c1c1c  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f
0x100080e0  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f
0x10008100  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f
0x10008120  0x212121  0x2f2f2f  0x212121  0x1c1c1c  0x0       0x0       0x0       0x0
0x10008140  0x0       0x0       0x0       0x0       0x0       0x0       0x0       0x0
0x10008160  0x0       0x0       0x0       0x0       0x0       0x0       0x0       0x0
0x10008180  0x1c1c1c  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121
0x100081a0  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121
0x100081c0  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121  0x2f2f2f  0x212121
0x100081e0  0x2f2f2f  0x212121  0x2f2f2f  0x1c1c1c  0x0       0x0       0x303030  0x303030
0x10008200  0x303030  0x303030  0x303030  0x303030  0x303030  0x303030  0x303030  0x303030
```

Milestone 2: Completed

- Movements implemented
  - W to rotate
  - S to shift current tetromino down
  - A to shift current tetromino left
  - D to shift current tetromino right
- Re-painting of screen
  - When moving the current tetromino, we made sure to re-paint the checkerboard background
- Quitting game
  - Player can quit the game at any moment

Milestone 3: Completed

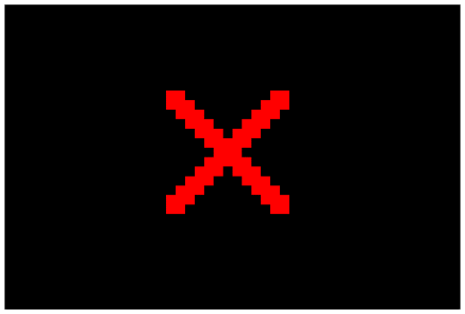
- Collision detection against existing tetrominoes and walls/ floor
  - If the movement is illegal, the tetromino will not follow the move
  - Once the tetromino reached the floor or rest on top of existing blocks, then a new tetromino will spawn



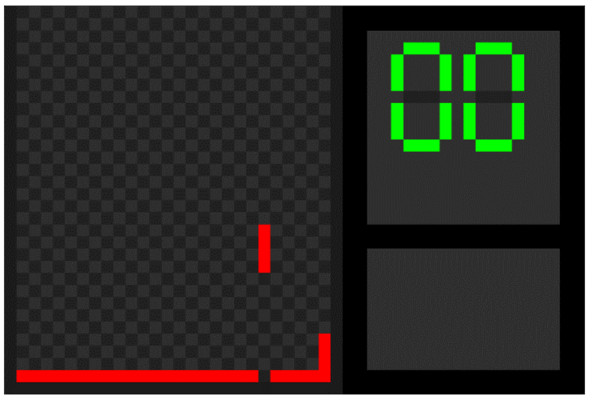
- Line clears when full line of tetromino pixels detected
  - all existing blocks in the container is shifted down

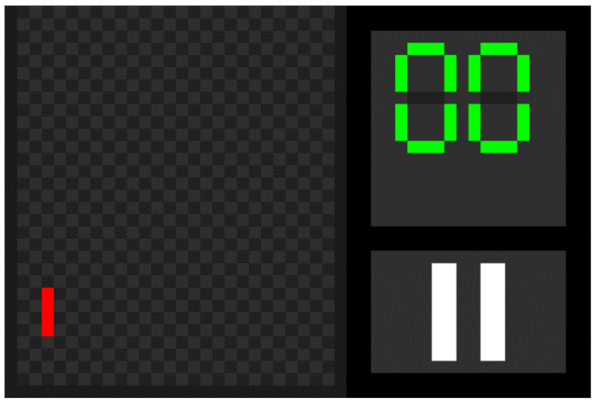
Milestone 4 & 5 Features: Completed (4 EASY & 2 HARD Features)

- Game Over (X Screen) and allowing “Retry” to restart the game (EASY)



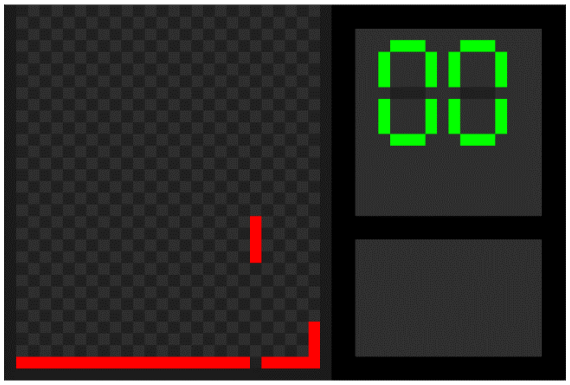
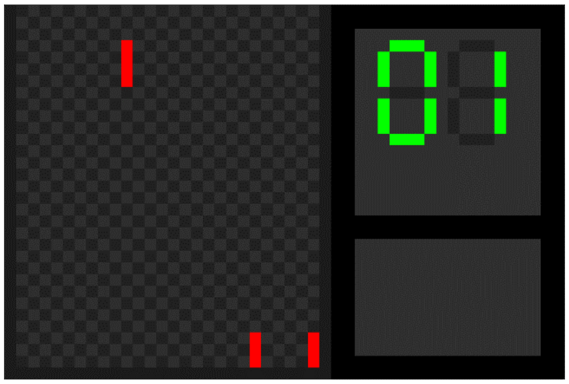
- Pausing and unpausing the game (EASY): denoted by the bottom right screen





- Sound effects for dropping tetrominoes (EASY)
  - valid move
  - invalid move
  - line clear

    - game over
- Implemented gravity (EASY)
    - current block will continuously shift down by itself at a given interval
- Score tracking and display based on numbers of line completion (HARD):
  before vs after

- Playing background music: Tetris Main Theme (aka "Korobeiniki") (HARD)
    - Theme will be played while game is being played
    - It will stop when player quits or pauses the game, OR when the game is over

Key Items stored in Memory

- Bitmap Width and Height
- Address of Display
- Address of Keyboard
- Colors
    - background color 1, 2 & 3
    - border color
    - white, red, green, blue
- Current Block Orientation: vertical or horizontal for I-block
- X and Y coordinates of each of the 4 pixels of the I-block
- Snapshot of the Arena
- Game Score
- Game Pause State (whether game is paused or unpaused)
- Background musical theme sequence, length and position

Key Implementations/ Core Functions

Painting on the Display

The core of the game relies on the PAINT subroutine. It takes in six arguments: the X and Y coordinates to start painting at, the number of iterations to paint "right" and "down" from the starting coordinates, and the two colors to alternate between. This subroutine was initially designed to be used for painting the checkerboard background of the Tetris arena, but we noticed that we can set two input colours as the same and generalize this subroutine to be used for painting any area with a single colour.

```
# Subroutine - PAINT
# Paints a given area on the connected bitmap display in a checkered pattern,
# alternating between two colors both horizontally and vertically.
#
# Arguments
#  - $a0: The X coordinate to start painting at.
#  - $a1: The number of iterations to paint "right" from $a0 (X).
#  - $a2: The Y coordinate to start painting at.
#  - $a3: The number of iterations to paint "down" from $a2 (Y).
#
# Stack Arguments
#  - $t8: The first color in the pattern.
#  - $t9: The second color in the pattern.
```

Moving the Tetrominos

Movement was handled in the *INPUT_LOOP* subroutine, which calls on *KBRD_INPUT* subroutine to handle branching to different handlers. There were seven handlers in *KBRD_INPUT* to handle a response to each of the seven input keys:

- *Q_RESPONSE*: Quit the game.
- *R_RESPONSE:* Reset the game.
- *P0_RESPONSE:* Pause the game.
- *W_RESPONSE*: Rotate the tetromino.
- *LEFT_RESPONSE*: Move the tetromino left.
- *DOWN_RESPONSE*: Move the tetromino down.’
- *RIGHT_RESPONSE*: Move the tetromino right.

To move the correct tetromino, the "state" of the current piece is maintained using a couple of memory allocations to store information on the current piece. Please note that in subsequent discussion, a "segment" of a tetromino means one of the four squares that make up the tetromino. In the current state of the game, the "I" block is the only one that's implemented. Its segment definitions are below:

```
OI1
[ ] 1     OI2 [ ][ ][ ][ ]
[ ] 2          1  2  3  4
[ ] 3
[ ] 4
```

- *currentBlockOrientation*: The current orientation of the tetromino (OI1 or OI2).
- *currentBlockX*: The current X coordinate of the tetromino, given by the location of segment 1.
- *currentBlockY*: The current Y coordinate of the tetromino, given by the location of segment 1.
- *currentBlockSegments*: The segments of the tetromino, where the X/Y position of each segment are continuous before the next segment (i.e., S1X, S1Y, S2X, S2Y, S3X, S3Y, S4X, S4Y)