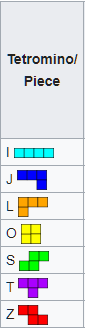
Tetris game overview

1. Falling pieces overview

The tetris gameplay is composed by tetramino pieces that fall from the sky and they stack onto the ground.

There can be 7 kinds of tetramino pieces.



Each tetramino piece is composed by 4 blocks that have different positioning.

Only one tetramino piece falls from the top of the screen at a time.

The tetramino piece can be moved to left or right by arrow keys and can be rotate using space.

1. Dropped pieces

Once the falling piece has reached the bottom or another fall piece the falling piece stops falling and cannot be controlled by the player.

Once the falling piece has stopped moving we will create another pieces called drops.

1. Playing field and progression

The playing field is composed by 9 horizontal rows and about 17 vertical.

Once a row is filled with blocks(drops) the drops on that rows are destroyed and all drops on the upper rows are shifted down.

If drops reach the top of screen is game over.

1. How we code falling piece

First we start with only one piece (the T piece).

There are 4 types of movement.

1. The piece is falling down with constant speed.
2. The piece can be moved to left or right
3. The piece can be rotated
4. The piece falls faster if press down

Overview: Each piece is composed by 4 blocks.

A block is a struct that for the moment contains X, Y and an Id

|  |
| --- |
| type Block struct { |
|  | x f32 |
|  | y f32 |
|  | id i32 |
|  | } |

We draw on screen the 4 blocks by looking and the X and Y values to determine the place of the block.

1. This part can be coded by decreasing slowly all blocks y value every tick:

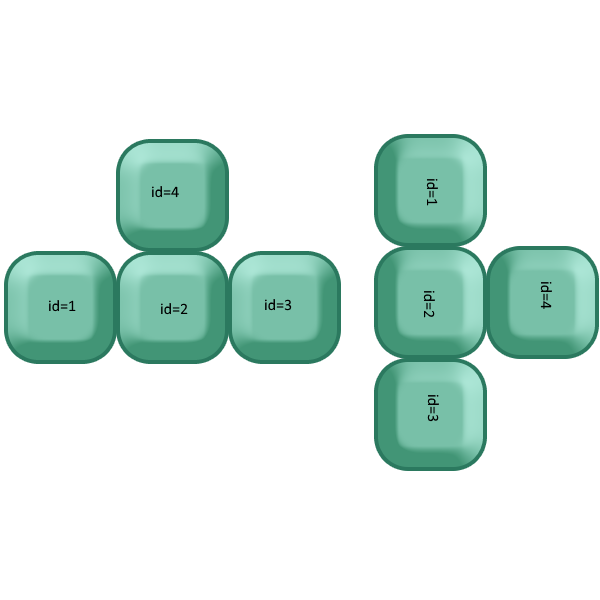
block.y=block.y - 0.001

1. Moving left or right happens at key press:

|  |  |
| --- | --- |
| if (key==263&&action==1){ | |
|  | |  |
|  | | shiftblockstoleft() |
|  | | } |
| Func shiftblockstoleft(){ |
|  | var newBlocks []Block | |
|  |  | |
|  | for c := 0; c < []Block. len(blocks); c++ { | |
|  |  | |
|  |  | |
|  | var block Block | |
|  | block := []Block.read(blocks, c) | |
|  | block.x=block.x - 0.15 | |
|  | newBlocks = []Block.append(newBlocks, block) | |
|  | } | |
|  | blocks = newBlocks | |
|  | } | |

Here we decrease X value of a block by 0.15 witch means a block width

Same thing for right move but instead of – we got a + sign

1. Rotate: when we rotate each block changes position on horizontal and vertical with a predetermined value:

As you can see in the image the block with id =2 stays in the same position , the block with id = 1 takes position of id =4 etc.

We move block with id=1 with 0.15 on X axis and – 0.09 on Y axis

The difference from 0.15 and 0.09 comes from game resolution.

1. For this case we simply increment the 0.001 value at what blocks are falling. This can cause later problems with overlap checks so we postpone it