Home Exam 3 for PG3400 - Basic C Programming

Vamsidhar Reddy Gaddam

November 2014

Abstract

In the following document, you will find the questions, hints and instructions for the third[final] Home Exam for PG3400. This exercise helps you with designing a game based on a limited set of tools. In this exercise, we will work on a single-threaded tetris game. The brave can venture at making it multi-threaded for fun. We will use *ncurses* library for UI.

1 Instructions

- You are encouraged to discuss about the problem and possible solutions, but your code is expected to be independent.
- Discussions on the forum might be particularly useful since I follow them too.
- The deadline is 23^{rd} of November and it is enforced by the system.
- Any hint of plagiarism will be heavily penalized.
- The style of the code will also be considered in grading.

2 Introduction

In this home exam we will design and develop the very first version of the game *Tetris*. In any game, the following are the most common blocks:

- ReadUserInput
- UpdateGameState
- RenderGame

Ofcourse, the more advanced games have concurrency but it is not a strict requirement depending on the game. Usual cases of concurrency are found in input, sound and rendering.

The game state itself is in most cases atomic. In our home exam, we will consider at a strict sequential game. The brave can consider multi-threading for the bonus task.

Tetris is one of those games that nobody needs an explanation for them. However we will define some basic rules to keep the implementation simple.

3 Global canvas

The background/canvas in tetris can be considered as an array of size wxh. Typically the w = 10 and h = 20. But it is required to be a variable and initialized at the start of the game. You can have either static blocks or empty spaces on this canvas.

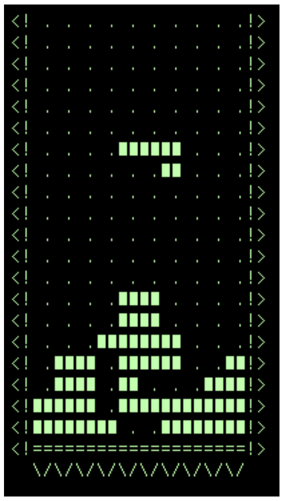

Figure 1: Example of a canvas from *http://en.wikipedia.org/wiki/Tetris*

You can use a global array initialized once in the beginning to represent a canvas. You can use '1' as a static block and '0' as empty space[these are just recommendations, you are free to implement the way you want]. Then you have one moving brick on the canvas.

4 Moving bricks

At any point of time, there is atmost one moving block/brick in the game. We can have one representation of a brick with 3x3 elements, a rotation state and the position state. Figure 2 provides various blocks that can be constructed with a simple 3x3 square. You can also create other blocks using the same 3x3 structure like 'T', 'J' and 'L'. Please note that I used colors to just provide a nice visualization, I do not expect you to make a color game!

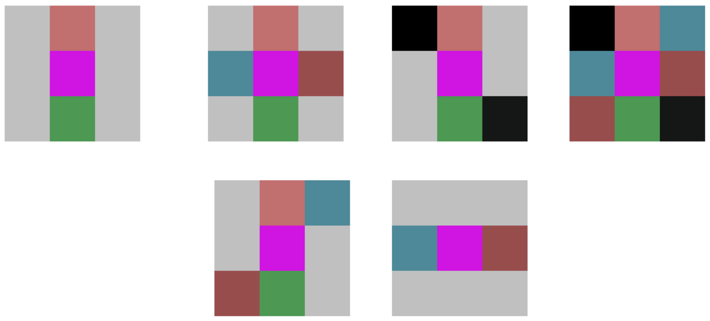

Figure 2: Example of brick construction from 3×3 square blocks.

The rotation state can be used to perform a soft rotation, you can also perform a hard rotation - in which case you change the actual elements of the block. These are implementation details, you should have the freedom to choose.

The position state holds the current position of the brick on the game canvas. Since the bricks are mobile, we have to store the position along with the brick. Let us assume that your brick is a 3x3 array B.

You can have the first brick as following :

```
B[0][0] = 0; B[0][1] = 1; B[0][2] = 0;
B[1][0] = 0; B[1][1] = 1; B[1][2] = 0;
B[2][0] = 0; B[2][1] = 1; B[2][2] = 0;
```

Operations on brick :

Move left, right, down - DO NOT MOVE UP: By doing this you just change the position of the brick.

rotate only in one direction [original version]: Let us take the same brick B[i][j]. You can perform a clock-wise rotation and get a new block C[i][j] as the following :

```
for all i,j
  C[i][j] = B[2-j][i]
```

Bonus : You can experiment with higher block sizes. For example a 4x4 block gives you more possibilities in the block types. However the rotation changes.

5 Brick interaction with canvas

The moving brick interacts with the canvas during the game. When one of the blocks touches any static block of the canvas with their bottom face, the block should go into the canvas and become part of it. However a floating block does not become a part of canvas. It is only rendered at the time of display.

When a brick becomes a part of the canvas, a new random brick is added to the game.

6 Game loop

During the game loop, the basic tasks are to take input, modify brick, update state and display the game. While taking user input, you should take care that the keys have ASCII codes.

During this loop, you must check for complete rows in the canvas. When a row is complete you should remove it and add it as a score. When a row is removed there are two possibilities of updating the canvas [1].

The game loops at certain fps. For example, $10fps$ implies that the game waits for about $100ms$ and then updates with default parameters. The default action is usually to move a block by one step downwards.

7 Display

When there is an empty space, a white-space character should be drawn. When there is a block to be painted, it can be painted with '#' or any character of your choice. A really simple display is to just show the static canvas and the floating brick at the position. However you can try to explore other possibilities like showing the next block to be displayed and the current score. Usually, this is the last call in the game loop.

8 Bonus

A game of tetris is always accompanied by a beautiful classic piece called *Korobeiniki.* A midi file and mp3 file are included for your convenience. Try to play this in the *background* in loop.

You can try to be really creative and create your first game in C for fun. Good luck with it!

[1] http://en.wikipedia.org/wiki/Tetris#Gravity