CSC 330 —Programming Languages
Fall 2018
Assignment No. 6
Version 1.1

Note 1 **This assignment is to be done individually**

Note 2 You can discuss the assignment with others. Talk about what you need to do, not how it is done. And do not share code.

- Due date: As indicated in Connex.
- This assignment is worth 1% of your total course mark.
- Submit electronically via Connex a single file with your solutions.

Objectives

After completing this assignment, you will have experience maintaining and application that uses object-oriented programming and Ruby.

Introduction

Tetris. The classic game. I spent too many hours pressing arrows and space looking at blocks fall. I even had dreams in which I played Tetris! In this assignment we will be enhancing an implementation of Tetris in Ruby.

Much of the work in this assignment is understanding the provided code. There are parts you will need to understand very well and other parts you will not need to understand[1]. Part of the challenge is figuring out which parts do what. This experience is fairly realistic: this is the way software development in the real world works.

Warnings

- Make sure you can run the game before you start. Installation instructions will very depending on what OS you use. You might need to install Ruby (in some operating systems it is already installed). You might also need to install the ruby-tk library. Please see `https://www.tutorialspoint.com/ruby/ruby_tk_guide.htm` for details. So please, make sure you can run it before you start your work (how many assignments do you start by playing a classic video game?).
- For Linux users only: I am running version 2.3 of Ruby on Ubuntu 18.04. I installed (as root) ActiveTCL (from)`https://www.activestate.com/activetcl/downloads`) and then run `gem install tk` (as user). After that I could run the program.
- You can run the provided code with `ruby hw6runner.rb original` from the command line.
- You **should only** modify `hw6assignment.rb`. You should not modify any other file.

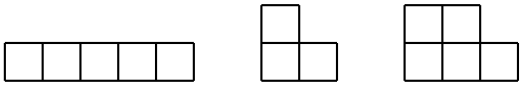
[1] This is an important skill. Think of it as a surgery. The best maintainers know what they need to understand, what to ignore and make the minimal changes needed to complete the job without breaking anything.

- Don't "test" your game too much. I think half of the time I spent doing this assignment (or maybe even more) was "testing it". And I **hate** testing!
- The instructor part of me tells me you should write unit-test for your new functionality. But the "real" me says: just play it and have fun (no, I did not write a single tests for you this time).

Your task, should you choose to accept it

A. In your game, the player can press the 'u' key to make the piece that is falling rotate 180 degrees. Note: it is normal for this to make some pieces appear to move sideways slightly.

B. In your game, instead of the pieces being randomly (and uniformly) chosen from the 7 classic pieces, the pieces are randomly (and uniformly) chosen from 10 pieces. They are the classic 7 and these 3:



The initial rotation for each piece is also chosen randomly.

C. In your game, the player can press the 'c' key to cheat: If the score is less than 100, nothing happens. Else the player loses 100 points (cheating costs you) and the next piece that appears will be:

The piece after is again chosen randomly from the 10 above (unless, of course, the player hits 'c' while the "cheat piece" is falling and still has a large enough score). Hitting 'c' multiple times while a single piece is falling should behave no differently than hitting it once.

Requirements

Follow these guidelines:

- Your game should have all the features of the original Tetris game, as well as the enhancements.
- The subclasses you create must start with `My` followed by the name of the original class. For example, your Tetris class should be called `MyTetris`. We have provided empty class definitions for you to complete. **You do not need any other classes.**
- **Do not add to or modify** any classes defined in other files or the standard library.
- It is required that your board `MyBoard` has a next piece method that provides the same functionality that Board's next piece provides, which is that it sets `@current_block` to the next piece that will fall, which might or might not be the cheat piece.
- You must have a `MyPiece` class and it must define a class constant that contains exactly the ten "normal" pieces (i.e., the initial seven plus the three additional pieces from enhancement two). It must not contain your cheat piece. It must be in the same format as the `All_Pieces` array in the provided code.

- All your new pieces, including your cheat piece, must use the same format as the provided pieces. Hint: Be particularly careful to have enough nesting in your arrays or your game may be subtly, almost imperceptibly, incorrect.
- Do not use the Tk library directly in any way. The only use of Tk should occur indirectly by using instances of classes defined in `hw6graphics.rb` as needed (only a little is needed). Do not have require 'tk' in your `hw6assignment.rb` file (or any other use of require for that matter).

Advice

- Make sure you can run the game before you start modifying the code.
- For the piece you are adding that has 5 squares in it and is not a line, be sure to add the piece as pictured and not its mirror image.
- It takes time to understand code you are given. **Be patient** and work methodically and incrementally. You might try changing things to see what happens, but be sure you undo any changes you make to the provided code. Insert puts statements to help you trace the program.
- My sample solution is 100 lines (including the provided code and empty lines). As always, that is just a rough guideline; your solution might be longer or shorter.
- While you should not copy code unless necessary, some copying will be necessary. After all, the original game may not have functionality broken down into overridable methods the way you would want and you are not allowed to change the provided code. This is a fairly realistic situation.
- The program should run the original game **unchanged** and run your **enhanced** Tetris (depending on the command line option).

Provided Code

There are four Ruby files involved:

1. The Ruby code in `hw6provided.rb` implements a simple but fully functioning Tetris game. If you have never played Tetris see `http://www.tetris.com/how-to-play-tetris/index.aspx`
2. In `hw6assignment.rb`, you will create a second game that is Tetris with some enhancements, described below. This is the only file you will submit.
3. The Ruby code in `hw6runner.rb` is the main starting point. From the command-line (not within irb), you can run ruby `hw6runner.rb` to play a Tetris game that includes your enhancements. To use the original game rather than your code in `hw6assignment.rb`, run `ruby hw6runner.rb original`. Look inside `hw6runner.rb` to understand how the program gets executed.
4. The Ruby code in `hw6graphics.rb` provides a simple graphics library, tailored to Tetris. It is used by the Tetris game in `hw6provided.rb`. Your code can also use the classes and methods in `hw6graphics.rb` (as well as the classes and methods in `hw6provided.rb`), except for the methods marked with comments as not to be called by student code. Your code **cannot use** the Tk graphics library directly.

How to run the game

Do not use the REPL to load `hw6runner.rb` and start a game. You can use the REPL for testing individual methods and exploring the program, but to launch a game, go into the command line and run `ruby hw6runner.rb` (or to make sure the original unenhanced game still works correctly, run `ruby hw6runner.rb original`). Make sure the 4 Ruby files are in the same directory and run the ruby command (or irb when exploring) from that directory.

For example, under linux I can run the game from the command line as:

```
ruby hw6runner.rb enhanced
```

or

```
ruby hw6runner.rb original
```

Notice how I explicitly indicate the version of ruby. That is because I have several versions installed in my laptop. You might not need to do that.

Challenge Problem

This time we are going to have a challenge problem. It can be worth up to 100% of your assignment grade (I know it is not much). But if you have the time and the inclination, you might be able to do something interesting that you are proud of, and that you can show to potential employers as your ability to program in Ruby and to demonstrate that you are willing to go beyond what is required of you. And more important, you will have an excuse to write a game in which you will `you will learn`.

1. First, in `hw6assignment.rb`, create classes `MyTetrisChallenge`, `MyPieceChallenge`, and `MyBoardChalleng` that subclass the classes you modified to complete your enhancements.
2. Modify `hw6runner.rb` so that running `ruby hw6runner.rb challenge` uses these new classes.
3. In these classes add a fourth enhancement to your game that is interesting and not similar to the enhancements already required.
4. In a short text file `challenge.txt`, explain what your enhancement is and how you implemented it. Enhancements that are particularly easy will not necessarily receive extra credit; aim for something at least as substantial as the required enhancements and perhaps affecting a different part of the game. The sky is the limit.
5. Make sure your main solution (the one that runs with `ruby hw6runner.rb`) is not affected —your challenge enhancements should only be in your new subclasses.

Your challenge problem may use the Tk graphics library directly if you wish — you are not limited to the functionality in `hw6graphics.rb`. Do not share your enhancement with anyone unless they already have their own. We do not want to see the same thing twice except by coincidence.

Evaluation

Solutions should be:

1. Correct. We will play your game and we don't like games that crash or act in unexpected ways.
2. In good style, including indentation and line breaks.

As usual, submit your solution via connex. If you submit a challenge, then submit also the challenge.txt file.

Have fun!