CS106 W20 - Assignment 04

Input and Output

Due: Friday, January 31, 11:59 PM

Begin with the starter code: A04_StarterCode

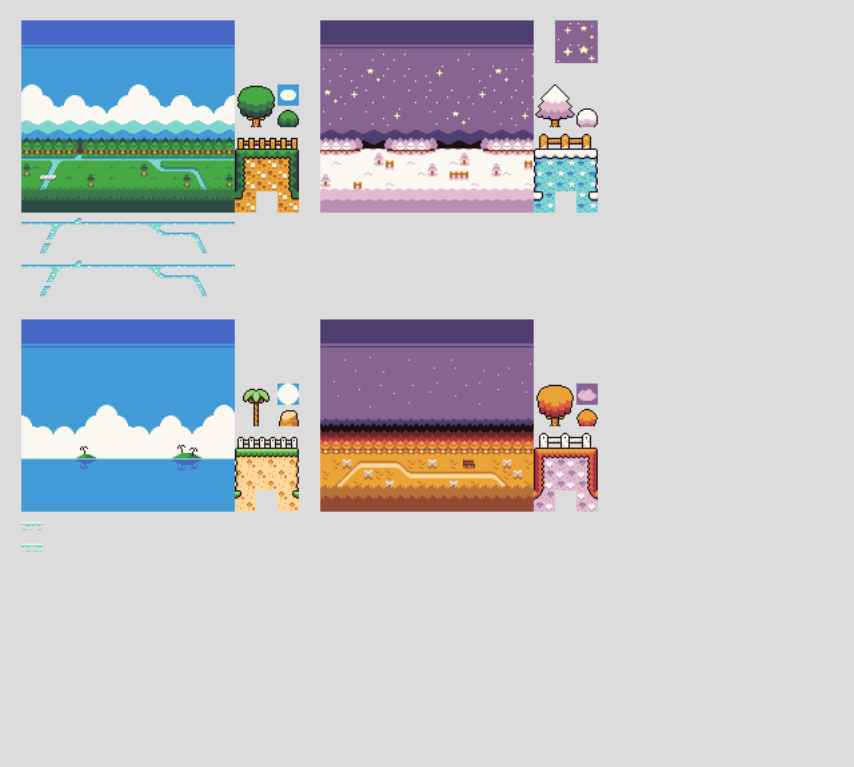
Create a folder: CS106_A04

Platformer [10 marks]

Many platformers (and indeed, many video games in general) are *tile-based*—behind the scenes, the graphical elements are broken down into a small set of tiles, which are reassembled like puzzle pieces into different configurations to define the game's levels.

In this question, you'll write a JavaScript p5 sketch that can manipulate tiles to draw one screen from a hypothetical platformer game. Proceed as follows:

1. Open "A04_StarterCode" in the Processing IDE. Save it as "A4_basic" inside the "CS106_A04" directory. Examine the code:
    a. The sketch window is 640×576. (This is four times the resolution of the original Nintendo Gameboy.)
    b. The preload() function loads the tileset from the data directory.
    c. In draw(), the tileset image is displayed in the sketch window. You can delete this line of code later. It is included in the starter code so that you see something on the screen. The code is: "image(tileset, 0, 0);".
    d. When you run the code you should see the following:

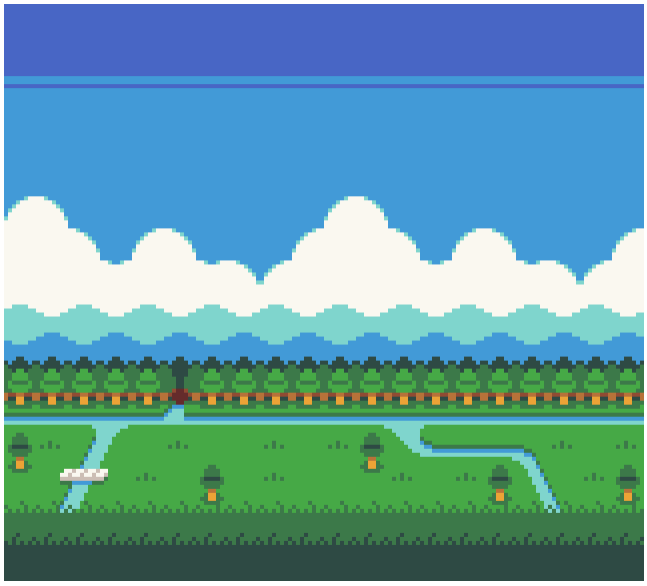


2. All the graphical assets for this tile-based game travel together in a single *tileset* image, which you can see above. This tileset incorporates four variations of a background image and simple platformer tiles, inspired by Gameboy-era art. The original is in the public domain, courtesy of GrafxKid on OpenGameArt. Tiles.png has four graphical themes: Grassland, Winter night, Tropical seaside, and Autumn evening.

3. You can now delete the line of code “image(tileset, 0, 0);” as it is not needed for this assignment. In the draw() function, draw the Grassland-themed background image from the top-left quadrant of Tiles.png, scaled so that it covers the entire sketch window. Use the built-in function copy(). Note that the top-left pixel of the background has coordinates (16,16), and the background has the same dimensions as the Gameboy screen: 160×144.

4. You may notice that the background image looks a bit blurry. That's because JavaScript p5 is trying to smooth it out as it scales it up by a factor of four on the way from the source image to the sketch window. Fortunately, there's a solution. Add a call to noSmooth() (with no arguments) at the bottom of setup(). That will make the sketch look more like classic, chunky pixel art.

   When you run the code you should see the following:

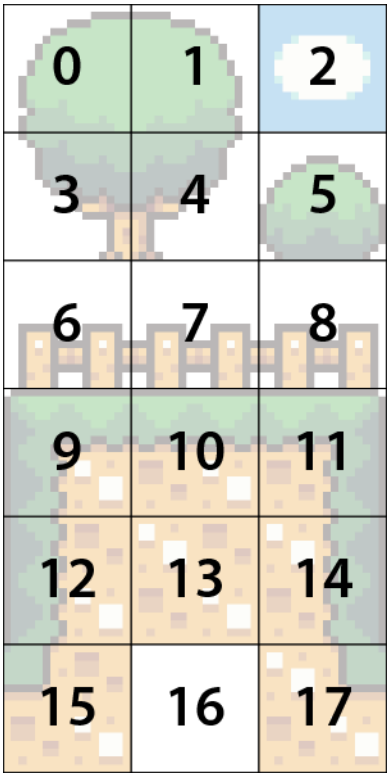


5. We're going to want to start drawing individual tiles to the sketch window. In order to do that, let's make sure we agree on how the tileset and sketch window are divided into grids.

   The sketch window is divided into a 10×9 grid of cells, each one of size 64×64 pixels. Here's a copy of the background image above, with superimposed *x* and *y* grid coordinates.

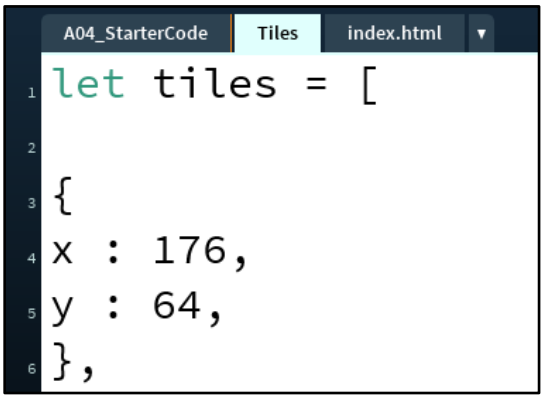
(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(5,0)	(6,0)	(7,0)	(8,0)	(9,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)	(8,1)	(9,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)	(7,2)	(8,2)	(9,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)	(7,3)	(8,3)	(9,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)	(7,4)	(8,4)	(9,4)
(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)	(7,5)	(8,5)	(9,5)
(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)	(7,6)	(8,6)	(9,6)
(0,7)	(1,7)	(2,7)	(3,7)	(4,7)	(5,7)	(6,7)	(7,7)	(8,7)	(9,7)
(0,8)	(1,8)	(2,8)	(3,8)	(4,8)	(5,8)	(6,8)	(7,8)	(8,8)	(9,8)

6. The platformer tiles all live in Tiles.png, in a little cluster to the right of the background image. They are all 16×16 pixel tiles. We'll simply number them consecutively from top-left to bottom-right. The numbered tiles and the x and y coordinates of the upper-left corner of each tile is shown below.



Tile:0: (176, 64)
Tile:1: (192, 64)
Tile:2: (208, 64)
Tile:3: (176, 80)
Tile:4: (192, 80)
Tile:5: (208, 80)
Tile:6: (176, 96)
Tile:7: (192, 96)
Tile:8: (208, 96)
Tile:9: (176, 112)
Tile:10: (192, 112)
Tile:11: (208, 112)
Tile:12: (176, 128)
Tile:13: (192, 128)
Tile:14: (208, 128)
Tile:15: (176, 144)
Tile:16: (192, 144)
Tile:17: (208, 144)

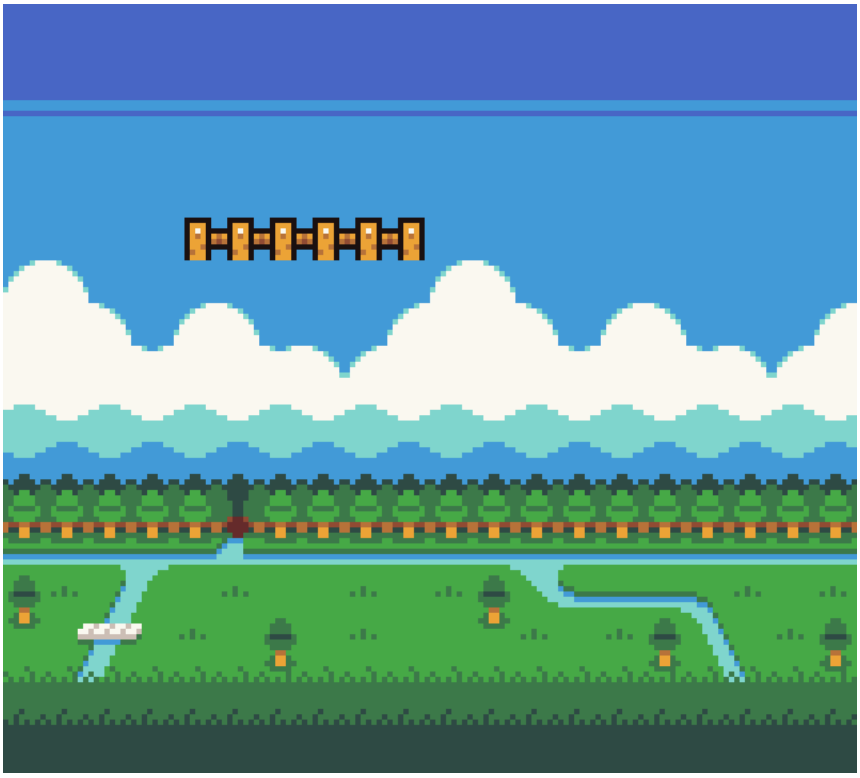
In the starter code, the x and y location of each tile is stored in a container array called “tiles”. This is in the starter code in a separate folder in the Processing IDE called “Tiles”. Look at the code in that folder to understand the layout of the array “Tiles”. The top left of your Processing IDE window should look as follows:



7. Drawing one piece of a level will consist of copying a 16×16 tile out of the tileset, scaling it by a factor of four, and pasting it into a 64×64 grid cell in the sketch window.

   Write a helper function "drawOneCell" that takes three integers as parameters: a tile number from 0 to 17, according to the numbering in the image above, and the *x* and *y* coordinates of a cell number in the sketch window, from (0,0) to (9,8) as in the first image above. The function should use copy() to transfer the numbered game tile into the corresponding cell in the sketch window. It does not return a value. For example, the following three calls to the function "drawOneCell" produce the following image:

```
drawOneCell(6, 2, 2);
drawOneCell(7, 3, 2);
drawOneCell(8, 4, 2);
```



   Test your "drawOneCell" function to convince yourself that it works. Try adding a couple of different calls to "drawOneCell" at the end of draw(), verifying that you get the expected tile in the expected cell.

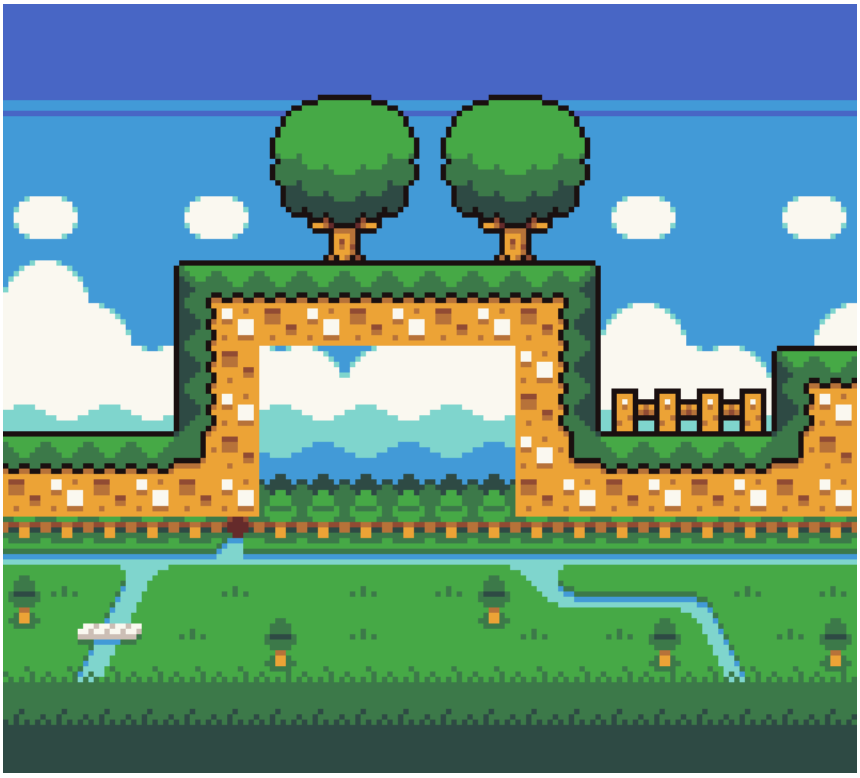
8. Next, you're going to add functionality to draw a level from a description stored in an external text file. That text file is already loaded for you in preload() in the starter code:

   ```
   level = loadStrings("data/level.txt");
   ```

   The file contains a sequence of lines, each one made up of three integers. The first two integers are the *x* and *y* coordinates of a cell in the sketch window. The third integer is the number of the game tile to put in that cell.

9. Tie the previous step into the rest of the game. In your draw() function, loop over all the lines in the text file (using the array "level" from the previous step). For each line, extract the three numbers and use your helper function "drawOneCell" to draw the tile in the right spot in the sketch window.

   For example, when rendering the test level.txt file provided above, the contents of your sketch window should look exactly like this:



10. Replace the provided level.txt with a new, original level of your own design. The level should use at least 12 placed tiles, and at least 6 distinct tile types. It should make logical use of the tiles: ideally, adjacent tiles should "fit together" as they were designed to do, creating a plausible game screen.

Save your code from steps 1-10 in A4_basic.

Requirements and Grading

General Correctness

- One mark will be deducted for files or directories named incorrectly (the zip file, etc.)
- One mark will be deducted if the wrong canvas size is used.
- One or more marks will be deducted if the program crashes (depending on the severity).

Assignments that do not run may receive a grade of 0. Even if you don't complete the entire assignment, don't leave it in a broken state. Make sure it runs so we can find ways to give you part marks.

[2 marks] Coding Style and Efficiency

Coding style is evaluated in both sketches.

Follow the course coding style for whitespace and comments. Consult the "Code Style Guide" on Learn. For example:

- Comment your code appropriately. Avoid superfluous comments.
- Correctly and consistently indent your code blocks.
- Use correct inline spacing for variable declaration and assignment.
- Use good line spacing to chunk sections of your code.
- Pay special attention to inline spacing for your conditional statements.
- Use semicolons.
- Use let or const, never var.

One or more marks may be deducted for solutions that have obvious inefficiencies.

- Variables that are declared or assigned, but not used.
- Unnecessarily variables that are duplicates of other variables.
- Unnecessarily repeating the same code in multiple places.
- Too many "magic numbers": the same number appears in many places indicating a variable should have been used instead.

[2 marks] Functionality or Visual Design Enhancements

Once you have basic functionality above working, enhance the functionality or the visual design in a sketch called A4_enhanced.
These basic requirements must still be followed:

- The canvas size must be the same size as A4_basic.
- It must maintain the functionality from a4_basic.

Here is an enhancement idea:

The steps 1-10 above are collectively worth about 90% of the marks for this question. For the final 10%, you can implement one additional feature: when the user presses the space key, rotate between the seasons. Tiles.png contains four distinct background images and four copies of the tiles; the space key should cycle through them repeatedly. If you've set up your code correctly, this feature does not require significant changes. You can introduce a couple of global variables that act as a global offset for the origin in all your copy() operations. Then you just need a keyPressed() function that modifies those global variables. In the files tiles.png each season is of size 160x144, and the upper left corner of the season images are at:

1. X = 16, y = 16 (this is top-left, grassland)
2. X = 240, y = 16 (this is top-right, winter night)
3. X = 16, y = 240 , (this is bottom-left, tropical seaside)
4. X = 240, y = 240 , (this is bottom-right, autumn evening)

If you do an enhancement, name your sketch above A4_enhanced, and save it in CS106_A04.

It's ok to not add enhancements or be extra creative if you're running out of time: a correct basic solution with excellent coding style should still achieve a grade of 86% (12 out of 14).

Restrictions

In general, you may not use any functions, libraries, or statements not covered in lecture or labs unless not specifically exempted below or in a post by a TA or instructor on this assignment discussion board. For example:

- NO translate(), rotate(), or scale() functions
- No classes
- You MAY use bezier, arc, and other standard drawing functions

If in doubt, make a post to ask about using a specific statement of function.
Functionality marks will be deducted for using forbidden functions/statements.

Submitting

Zip your assignment folder CS106_A04, and submit it the correct assignment dropbox. Consult "How to Submit" on Learn for more information on how to create a ZIP.

It is your responsibility to submit to the correct dropbox with the correct files before the deadline. Otherwise you will receive a mark of 0