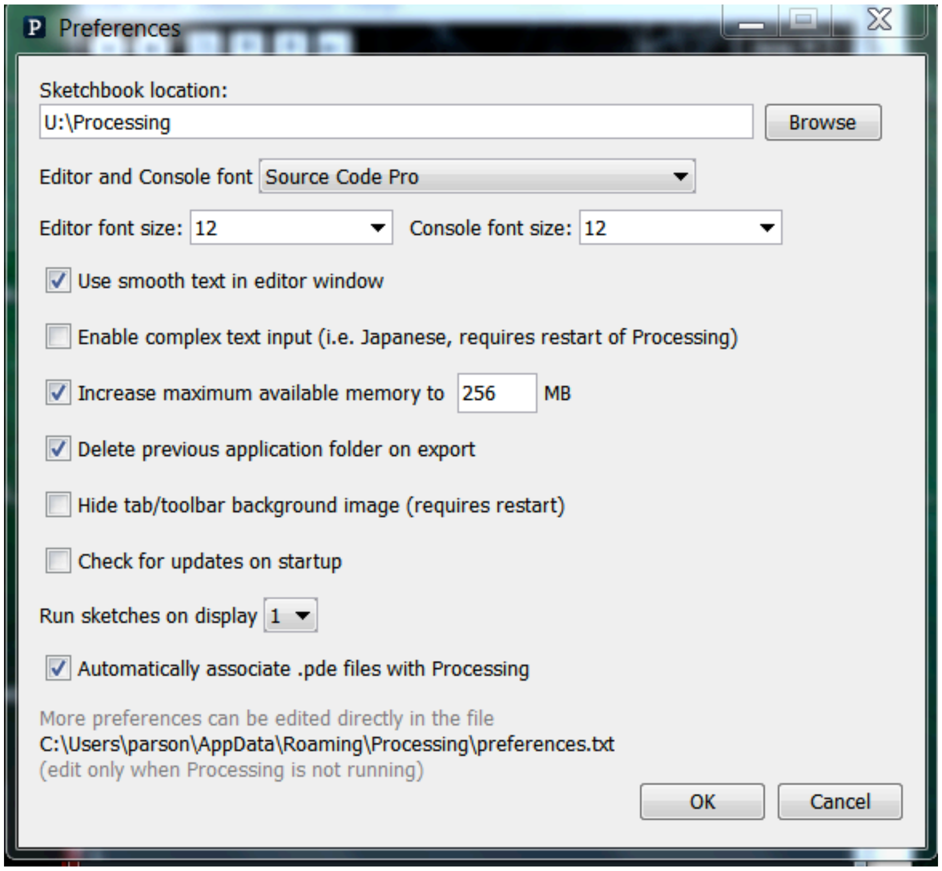
CSC 120 Introduction to Creative Graphical Coding, Spring 2018

Dr. Dale E. Parson, Assignment 1, Implementing and testing an automated avatar in Processing. This assignment is due via **D2L Assignments Assignment 1 avatar** by **11:59 PM on 23 February**.

When using Processing on the Kutztown campus Windows computers, make sure to start out **every time** by setting your Processing Preferences -> Sketchbook Location to U:\Processing. The U:\ drive is a networked drive that will save your work and make it accessible across campus. If you save it to your desktop or the lab PC you are using, you will lose your work when you log out. You must save it to the U:\ drive. If you do not have a folder[1] called Processing under U:\, you must create one using the Windows Explorer. Processing Preferences is under the File menu on Windows.



Sketchbook location: U:\Processing

Editor and Console font: Source Code Pro

Editor font size: 12 Console font size: 12

[✓] Use smooth text in editor window

[ ] Enable complex text input (i.e. Japanese, requires restart of Processing)

[✓] Increase maximum available memory to 256 MB

[✓] Delete previous application folder on export

[ ] Hide tab/toolbar background image (requires restart)

[ ] Check for updates on startup

Run sketches on display 1

[✓] Automatically associate .pde files with Processing

More preferences can be edited directly in the file
C:\Users\parson\AppData\Roaming\Processing\preferences.txt
(edit only when Processing is not running)

If you will be downloading Processing 3.X and running it using an off-campus computer (do not use version 2.X for assignments), you can copy your project sketch named **avatar** to a flash drive on one machine, and then copy it from the flash drive to another Processing sketch folder.

Create your **avatar** folder by running File -> Save As -> **avatar** after setting up your sketch folder.

You can copy and paste the text for my sketch from last semester into your sketch, then save it, as a starting point. It is here:
http://faculty.kutztown.edu/parson/fall2017/avatar2017NoModulo.txt

[1] Another name for a *folder* is a *directory*.

HOWEVER, to earn 100% you must completely redesign the background scenery, the foreground scenery, and your avatar. It must not be a small change to mine; it must be your own visual design. Nevertheless, starting with mine is useful because the setup() function, the required comments denoting background/foreground/avatar body parts, and the overall flow of the sketch will be the same as mine.

Once you have created your **avatar** sketch, write the following code as outlined in class:

1. Create some immobile background scenery. Plot this before the avatar.
   Keep them opaque, differing colors. Comment this section for each background object.

2. Create an avatar that is not the textbook's Zoog and is not Parson's avatar.
   Call **pushMatrix()** before starting the avatar to save the window based x,y coordinate system.
   Call **translate**(…) with coordinates for the center of the avatar's body.
   0,0 is now its reference appoint; use the center of its body for this approximate reference point.
   IDENTIFY WITHIN A COMMENT THE 0,0 POINT WITHIN YOUR AVATAR. ALL X,Y COORDINATES WITHIN THE AVATAR MUST BE RELATIVE TO A 0,0 POINT WITHIN AN AVATAR BODY PART, AFTER THE ABOVE CALL TO translate().
   If your avatar's x,y coordinates for its shapes are not relative to a 0,0 point inside its body, I will deduct 10 points.
   Optional bonus points 1.5 for using **scale**, another 1.5 for using **rotate** for the avatar.
   Use at least one shape from "2D Primitives" (on the manual page) that Zoog does not use.
   Zoog uses ellipse, line, and rect. Note use of rectMode and ellipseMode; I recommend CENTER.
   You may also use ellipse, line, and rect.
   Use variations in color.
   Use variations in alpha.
   Use at least 5 distinct shapes, meaning at least 5 body parts for the avatar.
   Call **popMatrix**() before resuming display of foreground scenery; it restores 0,0 to left,top.
   Comment this section for the start of the avatar, the end of the avatar, and for each body part.

3. Create some immobile foreground scenery. Plot this after the avatar.
   Keep them opaque, varying colors across time. Comment this section for each foreground object.

   You may also have some mobile scenery in the background or foreground at your discretion. Two students had tumbleweeds last semester. If you do use mobile scenery, use pushMatrix(), translate(), and popMatrix() for each, similar to the required used for the avatar. Mobile scenery will be easier to implement in assignments 2 and 3, so you may want to wait.

4. Your avatar must move horizontally and/or vertically at some rate.

5. Your avatar MUST wrap back around to the other side of the window, OR bounce back in opposite, when hits window's edge.

You must use an "if" programming construct to control location and speed, similar to "if" conditional statements in my example solution, and similar to the code below. Using an "if" statement will be on an exam.

```
xloc = xloc + xspeed ;
if (xloc >= width) {
    xloc = 0 ;                    // wrap from right to left side of display.
    xspeed = 1 ;
}
if (xloc < 0) {
   xloc = width – 1 ;       // wrap from left to right side of display
   xspeed = -1 ;
}
```

// Use a similar approach with yloc, yspeed, and the display **height** variable for vertical movement.

The above code wraps around the screen. An alternative is to bounce off of the edges, like this:

```
xloc = xloc + xspeed ;
if (xloc >=width) {
    xloc = width - 1 ;
    xspeed = -1 ;
}                         // bounce back from right side of display.
if (xloc < 0) {
   xloc = 0 ;
   xspeed = 1 ;
}           // bounce back from left side of display.
```

// Use a similar approach with yloc and the display height variable for vertical movement.

6. Your avatar must have some "body part" that wiggles, grows/shrinks, changes color periodically, or moves in some manner, without becoming disconnected from the avatar. Step 6 may use the modulo operator (%), which gives the remainder of division, to wrap back around to its starting point. It may use "if" instead. You can use a variable pair to control wiggle offset and speed, similar to xloc and xspeed examples above.

Instead of using **width** or **height** in a formula such as those above, you could use expressions like (width – 20) or (width + 20) to allow the avatar to move slightly "off stage".

Use colors and alpha (stroke() and fill() and background()), strokeWeight(), and a variety of shapes. Get into creating a composition. Consult https://processing.org/reference/ and use some functions to make your animation exciting.

Grading: Each of the above 6 steps is worth 13% each (78% total).

Out of the remaining 22%, half (11%) are for quality and thoroughness of your visual design in replacing my background, foreground, and avatar. This must not look like a small set of tweaks to my sketch. You must design your own visual elements.

The other 11% are for correctly using a 0,0 reference point within your avatar's body as required IN CAPS in the above specifications, and for commenting as required above. Start your sketch with this documentation section:

```
/**********************************************************/
/* Author:
/* Creation Date:
/* Due Date:
/* Course:
/* Professor Name:
/* Assignment:
/* Sketch name:
/* Purpose:
**********************************************************/
```

We will have some time in class for working on this project. If you do not get it done in class, you will have to complete it as homework. I expect it to be to me by the due date via D2L. **<u>I will deduct 10% for each day it is late</u>**. Also, re-read the above requirements when you turn it in, to ensure that you don't miss anything. If you make changes after turning it in, just turn in another copy of your sketch via D2L. I will look at the last one that you turn in.

<u>TURNING IT IN</u>: When your work is completed, and you have re-read and satisfied the project requirements, you can use the Windows Explorer to find the file **avatar.pde** in your **U:\Processing\avatar** folder. Drag **avatar.pde** into the **Assignment 1 avatar assignment dropbox** under our course's D2L account by the due date. If you find you have created an error, you can drop an updated avatar.pde into the dropbox. **Assignment 1 avatar** is under **Assessments -> Assignments** in our D2L account. If you are working on a laptop or your machine at home, turn it in via D2L in the same way.