CSC 480 Object-Oriented Multimedia Programming, Fall 2021

Dr. Dale E. Parson, Assignment 3, manipulating pixel-based "photos" of the canvas.
This assignment is due via **D2L Dropbox <u>Assignment 3</u>** due by **11:59 PM on Thursday November 4**.
10% penalty for each day it is late.

See earlier assignment handouts for downloading & setting up Processing 3.x.
Make sure to set your Sketchbook directory to a persistent location at the start of each session to ensure that your work is not erased when you log out.

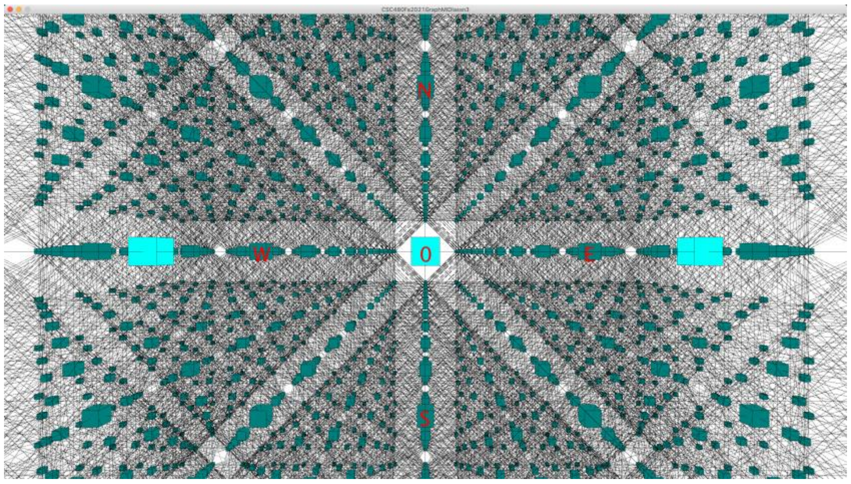
The handout starting code is at this link. Download this ZIP file and unzip it into your sketch directory as sketch folder CSC480Fa2021GraphMIDIassn3.

https://acad.kutztown.edu/~parson/CSC480Fa2021GraphMIDIassn3.zip

It contains the following 4 .pde files. Submit them all, individually to D2L by the assignment deadline.

CSC480Fa2021GraphMIDIassn3.pde	The main sketch. Add you name at the top.
Graph.pde	A 3D graph of nodes. STUDENT work comments are here.
Navigation.pde	3D interactive navigation.
MIDI.pde	MIDI data structures and helper functions.

There are STUDENT comments in upper case in Graph.pde every place that you have to write code. I have numbered them in approximate order of work. Do not change the keyboard commands, although you may optionally add new ones if you document them in the main sketch tab and let me know to look for them. Here is a screen shot from the handout code. The video below has much more information.



Screen shot of the 3D graph, see the video for better information

There is a video overview of this animated assignment here.

https://kutztown.zoom.us/rec/share/yRyk-CiAOKPbaGBQzQ9el0fuBsSH7Q1h26Cy6C3V9uF22enaEGKuwscqTss_92n1.ydp8cwp5I3Em52RR

The setup() function builds the graph and the first call to draw() connects the default MIDI software synthesizer. Sometimes connecting to MIDI times out with this error message in the Processing IDE:

RuntimeException: Waited 5000ms for: <a750231, 4b2fd0f>[count 2, qsz 0, owner

Usually if I wait a few seconds and try again it works. I have never had to restart Processing or reboot the machine. It would time out more when setup() tried to connect to the MIDI library's synth.

Here are the STUDENT requirements and points from Graph.pde. Edit the Graph tab and search from the top for STUDENT in upper case. The comments appear at the appropriate places in the code. Each requirement is worth 10% of the project, and there is an optional 10-point bonus opportunity. Make sure to put your name at the top of the main sketch.

```
/*
  STUDENT 1 10%: Make a 16-element float array HUES[] similar to the instruments[]
  array in the MIDI tab, with the HSB colors (hues) you want to use for your
  graph nodes. Each Z layer gets its own hue. Mine are all cyan in the
  handout code. Once your HUES[] array is ready, change the hue argument to the
  fill() call in Node.display() to use the hue for that Node's Z layer, using
  the same index into your HUES[] array that the handout code uses for the
  PROGRAM_CHANGE, which is the instrument.
*/

/*
  STUDENT 2 10%: In the MIDI tab change the 16 numbers in the instruments[] array
  to select instrument sounds of your choice. Although General MIDI instruments
  are documented under the "GM Patches" heading here
  http://midi.teragonaudio.com/tutr/gm.htm , not all of those 128 distinct
  instruments are implemented in the OpenJDK MIDI library we are using.
  Select some instruments that sustain until they receive NOTE_OFF. Not all
  need to sustain. The numbers on that page run [1, 128] so you must subtract
  one in code to maintain values in the range [0,127].
*/

      /*
        STUDENT 3 10%: Set tmpx and tmpy respectively to the mean of
        x, y and xeye, yeye respectively, thereby displacing
        the Node display() halfway between its resting position and the
        camera POV. Do not displace tmpz. Use these formulas for tmpx and tmpy:
          tmpx = round((xeye-width/2+x)/2) ;
          tmpy = round((yeye-height/2+y)/2) ;
        Those are necessary because xeye and yeye are relative to 0,0 at the
        left, upper of the display, while a Node's x,y is relative to the
        center of the display.
        This displacement occurs only when the Node is sounding.
      */

// STUDENT 4 10%: Map the worldzrotate amount to a value in the range [0,127]
        // by calling radiansToLimit() defined at the bottom of this tab, and use that
        // as the 4th argument to an added call to
        // sendMIDI(ShortMessage.CONTROL_CHANGE, zseqno % 16, modwheel, ...)
        // see http://midi.teragonaudio.com/tech/midispec.htm Modulation wheel.
        // Make sure to test via holding down the 'z' or 'Z' key for a full
        // rotation. Maximum effect should occur at the 180 degree rotation.


        // STUDENT 5 10%: Map the worldxrotate amount to a value in the range [0,127]
        // by calling radiansToLimit() defined at the bottom of this tab, and use that
        // as the 4th argument to an added call to
        // sendMIDI(ShortMessage.CONTROL_CHANGE, zseqno % 16, YOURCHOICE1, ...),
        // where YOURCHOICE1 is an unused constant value defined in the line after modwheel above,
        // from this MIDI CONTROL_CHANGE spec, that you can hear with this synth.
        // Test by holding down the 'x' or 'X' key.
        // Use sketch ConcentricCirclesIntervals to explore CONTROLs that actually work.
        // I have found only 1 fx between Effects_Level and Phaser_Level can be heard.
        // http://midi.teragonaudio.com/tech/midispec.htm

        // STUDENT 6 10%: Map the worldyrotate amount to a value in the range [0,127]
        // by calling radiansToLimit() defined at the bottom of this tab, and use that
        // as the 4th argument to an added call to
        // sendMIDI(ShortMessage.CONTROL_CHANGE, zseqno % 16, YOURCHOICE2, ...),
        // where YOURCHOICE2 is an unused constant value defined in the line after modwheel above,
        // from this MIDI CONTROL_CHANGE spec, that you can hear with this synth.
        // Test by holding down the 'y' or 'Y' key.
        // Use sketch ConcentricCirclesIntervals to explore CONTROLs that actually work.
        // I have found only 1 fx between Sound_Variation and Sound_Brightness can be heard.
        // http://midi.teragonaudio.com/tech/midispec.htm
        // The Volume CONTROLler is another option. I recommend mapping the return value
        // [0, 127] from radiansToLimit to the range [48, 80] using Processing's map
        // function to limit silence and over-driving, ONLY IF YOU USE VOLUME.

/*
      STUDENT 7 10%: When this Node isSounding, make a shape more elborate than a simple box.
      It could be multiple boxes rotated in different directions, optionally with
      some of those boxes being asymmetrical cuboids, or this box plus some 2D shapes.
      I avoided sphere() because too many spheres slow down the frameRate too much.
      Use coordinates as offsets to the tmpx, tmpy, tmpz translated reference above.

      STUDENT 8 10%: When this Node isSounding, rotateY the shape(s) below continuously by
      1 degree each time. This requires adding a float variable rotateYamount to this
      class, initialized to 0.0, incremented by constant degree in the Navigation
      tab each time isSounding is true, and reset to 0.0 when isSounding is false.

      STUDENT 9 10%: When this Node isSounding, scale the shape(s) below continuously by
      incrementing or decrementing by .02 each time, never allowing the scale to go below
      0.5 or above 2.0. This requires adding two float variables scaleAmount and
      scaleSpeed to this class, scaleAmount initialized to 1.0, incremented by scaleSpeed
      each time isSounding is true, and reset to 1.0 when isSounding is false.
      scaleSpeed is initialized to .02, set to - scaleSpeed when scaleAmount reaches or
      exceeds 2.0, and reset to abs(scaleSpeed) when scaleAmount reaches or goes below
      0.5. Reset scaleSpeed to 0.02 when isSounding is false.

      STUDENT 10 10%: After STUDENT 9 is working, modify the call to scale to use
      scaleAmount for the X argument, 1.0 / scaleAmount for the Y argument, and 1.0
      for the Z argument. scale() can take 3 different arguments for 3D.

      STUDENT BONUS POINTS +10%: Take a subset of instruments defined in array instruments[]
      and make then rhythmic by turning them NOTE_ON and NOTE_OFF using a regular timing.
      With a frameRate of 10 as initialized in setup(), the value "frameCount % 10"
      will be 0 every tenth of a second. You can use that as your time base, for example
      ((frameCount % 10) == 0) for NOTE_ON and ((frameCount % 10) == 5) for
      NOTE_OFF. Doubling 10 and 5 in those expressions cuts tempo in half. DO NOT DO
      THIS FOR ALL INSTRUMENTS. Some instruments that sustain should be left to sustain.
      If you do for these BONUS points, add a comment telling me to at the top of the
      main sketch where I mentioned this.
```

```
*/
```