EECS 1710
PROJECT :: Audio-Visual Instrument/ Generative Artwork

Prerequisite - please ensure that you have setup your EECS account and can log into the lab machines prior to starting this lab!

Resources:

Processing Reference:	https://processing.org/reference
Data types:	https://processing.org/reference/#data
Graphics:	https://processing.org/reference/#shape
Sound: (must import)	https://processing.org/reference/libraries/sound/index.html
Images:	https://processing.org/reference/#image
Colour:	https://processing.org/reference/#color
Input (mouse/keys/files)	https://processing.org/reference/#input

** *Also see lecture notes*

The goal of this project is to create a small Processing application that combines audio/visual or image/graphics with basic inputs from the user – as covered throughout various aspects of the course. The application should generally fit into one of the following categories (please see instructor if you are unsure and want to discuss an idea you have):

1) **Create and Audio-Visual Instrument**
    a. The application should link graphic elements (static or generated) in some way to audio outputs, so that a graphic/image-based component drawn in the app window (or the act of drawing a component) allows for some form of trigger/control of an audio response.
    b. The application must support basic inputs via either the keyboard (key presses) and or mouse clicks/motion or utilization of the mouse location (with/without clicks), in order to generate graphic elements and/or sound elements.

NOTE: There is no requirement for any animation, though if you like you can have a visual respond to keypresses or mouse clicks.

Example ideas:

i) create a SIMPLE visual representation of a mixing desk, and include regions that have various functions (for e.g. imagine two record graphics, where clicking on each one will toggle the playback of a sound/music file, and movement of the mouse around the app window maps to mixing between them (i.e. adjustment of amplitudes of the sound files based on mouse movements in one axis; or perhaps to adjust pitch using another axis, etc.)

ii) build upon your audio lab (tone generator), to make a SIMPLE visual keyboard (or similar), with some simple visual representation of a metronome (regular ticking), and perhaps a drum pad (that triggers beats, or starts/stops a regular beat/tick). Or some interesting variant of this (an interactive dukebox or similar). Perhaps several instruments/files that can be displayed and toggled so that they layer together.

2) Create a Generative Artwork

a. The application should spawn some iterated graphic components based on key presses, mouse clicks or some combination of these, or use locations, in order to generate parameters that can be used to control various elements of the graphics/images displayed in the artwork. A large part of this can be automated (even randomly so), linked in part to inputs provided by the user.
b. There should be at least one static/predetermined element, one direct input, and one indirect/randomized input that drives the artwork produced. If using mouse inputs for instance, you could define a position on the screen, or position relative to a part of the screen, or some start and end position (or a set of positions captured as mouse clicks on the app window) – to be used to generate graphics. For example distances/angles between mouse positions could be used to parametrize and generate a regular pattern.

Example ideas:

i) create a SIMPLE generative landscape that the user can interactively construct by key presses or mouse clicks into the app window, that will spawn the drawing/painting of graphics/images representing foliage, or some other useful atomic elements for the "scene/landscape" in mind.
ii) create a set of SIMPLE geometric patterns for which elements of stroke, colour and fill can be readily controlled through inputs from the keys or mouse.
iii) fragment and mix patches from images that are spatially laid out according to some input parameters from the mouse/keyboard, or try to fracture / mix some images according to inputs via the mouse/keyboard (such as dynamically controlling blend/crop etc via the mouse location on the image).

3) Create a very simple one-click game/puzzle

a. The application should allow for mouse based or keyboard based input to move a character/piece around the app window toward some end goal. Do not try to make anything elaborate, rather think about some simple puzzle mechanic, where the movement of a piece can achieve some sort of simple goal – like following a path, collecting elements or avoiding obstacle(s) to reach an end destination.
b. KEEP IT VERY SIMPLE
c. The game could incorporate both visual elements and audio elements (that signal progress and/or when the goal has been reached)

d. Try to keep the game/puzzle to a single, relatively static scene – or some very simple motion (similar to the projectile motion examples in class… or even simpler - linear / step-based motions).

*** there is only a very limited expectation for interaction (limited to mouse/key presses and related) – nothing really more than we have explored in class/labs*

Example ideas:

i) Create a simple jigsaw puzzle, or tile sliding game (to re-construct an image)
ii) A simple platformer with a moving character that moves due to a single click, or has its movement perturbed by a mouse click or keypress.
(static puzzles might be easier, but if not… **lerp()** is your friend !)

SUBMISSION (Due 5:00pm Tue Dec 6th, 2022 – HARD DEADLINE)

YOU MAY WORK IN GROUPS (2-3 students per group). You may submit one

You will have one submission folder:

- sketch folder (`courseProject`) with your project file(s)
- a 1-2 page brief (`readme.pdf` or `readme.docx`) document explaining:
  - what your project is/does
  - how to run/use the project (show at least one graphic/diagram)
  - references of any sources (code/inspiration) used

ALWAYS CITE SOURCES – DO NOT SIMPLY SUBMIT SOMETHING YOU HAVE FOUND ON THE INTERNET – IF YOU WORK WITH SOME CODE YOU HAVE FOUND ONLINE, **YOU MUST ALTER IT SIGNIFICANTLY**! (and still cite sources)

To submit… submit your full sketch directory (with all files and brief inside). Say the project folder is called "`courseProject`/", then to submit, go to the parent directory of the `courseProject` directory, and type:

```
submit 1710 project courseProject/*
```

NOTE: REMEMBER, you can choose to use the web-submit function (see Lab 0 for walkthrough). You will need to find and upload your project *.pde files independently if doing it this way.

Web-submit can also be used to check your submission from the terminal, and can be found at the link: https://webapp.eecs.yorku.ca/submit/

*** DETAILED RUBRIC TO FOLLOW*