**Creative Audio-visual Coding Project Log**

Date: 31 March 2021

* I have taken a break from the project and this is the first session I have come back to it and plan to consolidate my ideas. Currently I have created an audio-visual piece of art designed around the song ‘raid’ by Madvillain. In the piece the visual responds ‘dancing’ to the music.
* What I like:
  + The style which I’ve created. I think it has an interesting aesthetic.
* Improvements:
  + I want to make it applicable to different songs. For this I will need instrument detection at different frequencies.
  + I want to make the movements more interesting maybe using sin and cos functions. – More fluid not so blocky shapes.
  + Different base song – this one is too repetitive – therefore jazz.

I want it moving through the 3D space into the screen.

Think I am going to change the song to cold coffee – rotate camera moving through?

Date: 1 April 2021

In this session I changed the test song which I would use to Cold Coffee by Galt MacDermot. I then experimented with 3D shapes in particular spheres. I focused on learning how they could be moved and rotated and the use of various camera objects to allow coherent movement through the window. I drew some diamond shapes using lines around the central sphere.

In my new and improved idea, I wanted shapes to appear and fade backwards into the Z plane. This would allow a fresh level of movement in the piece and allow me to implement new shapes and ideas fluidly as if they were actually moving through space rather than just appearing.

I assigned different part of the audio data to variables e.g. ‘bass’ ext. using ‘getEnergy’ objects for specific frequency ranges.

How was this different to before? Before I used peak detected.

Text

Description automatically generated

Date: 2 April 2021.

I experimented more with the rotation and translation in shapes. However, upon testing the shapes didn’t seem to be positioned or moving how they should. I realised that my rotations were all effecting different shapes. Upon researching I found out that ‘push’ and ‘pop’ objects could be used to save and recall translation and rotation settings.

I used ‘bass’ variable to control rotation speed for the diamond/square line shape compositions and the size of the central sphere. I also drew some perpendicular lines on the outer edges of the canvas used the ‘treb’ variable to trigger translation in them. E.g.

Text

Description automatically generated

Date: 8 April 2021.

I added another larger square (line) layer to composition of shapes at the beginning of the piece. With this I created another push(), pull() instance and gave the line objects larger values. Key in this addition, after mapping the ‘bass\_real\_mapped’ to a value of 0 to 1 and assigning it to the bass\_real\_mapped2 variable I multiplied the positions of the lines by this value. Therefore, the quadrilateral dynamically changes size depending on the bass content in a given section in the song. This creates another relationship between the song and the movement of the visual shapes.

In addition, in this session I have been attempting to advance the idea of adding new shape compositions in the piece throughout by translating on the Z axis. I want to do this somewhat in time with the music. My current idea is to clock the peaks of the music and then introduce … different shapes when this clock reaches a threshold. Therefore, the shapes should enter the view of the camera on time. I plan to use the peak.detect() object for this.

Date: 9 April 2021.

To start today off I am still trying to find a way to introduce elements (shape compositions) onto the screen on time with the music. I attempted to calculate BPM using a clocking feature, where if a peak was detected, the clock was increased by one. My plan was to then use an IF statement and calculate time at different specific clocked points i.e., when ‘clock’ moves from 1 to 2. This was necessary because I tried outputting runtime when a peak was detected but it output multiple values as the peaks had length in time. I only wanted the first runtime when a peak was detected. For this to work I used another IF statement inside the first one so that only if the variable e.g., ‘time’ which I had initialised in set-up was ‘0’ then it will be assigned the runtime. Therefore, the variable would not be reassigned.

Graphical user interface, application

Description automatically generated with medium confidence

Therefore, I had managed to calculate times at specific peak points. In each of the IF statements I assigned the values to a variable ‘time’ and ‘time2’. This is shown in the code below.

Text

Description automatically generated

At the bottom of this code, it shows calculating ‘bar\_time’ by subtracting ‘time’ from ‘time2’. This was the time for a bar. ‘Clock == 1’ and ‘Clock == 10’ was decided on because after some of experimentation I worked out that there are ‘9’ peaks in the first bar, the second bar starts on peak ‘10’.

Date: 14 April 2021.

In this session I wanted to take the time calculations which I had done in the previous session and use them artistically, introducing new shape compositions in time with the music. First, I attempted to do this simply to work out how to do it before expanding the premise for more complex ideas.

I wanted to draw a rectangle in on in time on the second bar. For this I used the code below.

Text

Description automatically generated with medium confidence

This took a few different approaches and tests before I realised that I needed to add ‘time’ to ‘bar\_time’. This is because there is a gap in time between the first bar of the song and the first second of the song file. I realised this after testing for different time values i.e., ‘4\*bar\_time’ all seemed to be non-similarly out of time.

Date 15 April 2021.

My next challenge was taking these shapes which I could draw in time with the music and being able to place them in desired points in the z axis, as so far, the shapes appear however the relationship between music and the points where the shapes appeared are not linked.

Date: 17 April 2021.

Today I am thinking of using a more efficient way to contain data of time and desired space on the z axis, an associative array. Basically, to explain in a bit more detail, I need for lots of elements a time which is related to the music e.g., the beginning of a bar of music so I can draw in shapes on beat. In previous sessions this time has looked like ‘time + bar\_time’ or ‘time+(2\*bar\_time)’. I also need a value for the frame count at each of these times so I can translate shapes I am drawing by this value, (on the z axis) or factors of this value, to negate for the camera movement. Thus, making the shape at point of first drawing (when translated by the value) full size to screen wherever the screen is. This is part of my aesthetic goal.

If I use an associative array to store these values it will be much more efficient than using tiered IF statements, also an associative array is necessary because more than one value is required for each element.

I created the associative array called ‘elements’ and within this created sub-arrays ‘bar1’, ‘bar2’ ext. to contain data for these different points. This is the code for the array.

Graphical user interface, text, application

Description automatically generated

In testing I had trouble with the space points not being correct when applied to shapes however I realised that these shapes were being translated by other ‘translate’ functions in the code. I used ‘push()’ ‘pop()’ to stop these unwanted translations.

Graphical user interface, text, application

Description automatically generated

Upon testing I also had problems with when I used this for elements past bar 1 e.g., ‘elements.bar2’ as I had used incorrect calculations for space. Figure … is the correct code but before I used for example ‘2\*(Linetranslate-LineTranslate\_Pre)’. ‘LineTranslate\_Pre’ is necessary as the frame count at the beginning of the first bar and works in a similar way to the ‘time’ object for time. It is important to factor in as an initial value but crucially not multiplied in for each different bar value wrongly like ‘2\*(Linetranslate-LineTranslate\_Pre)’. For example:

* The frame count at the beginning of the first bar is 20 and at the beginning of the first bar is 100.
* 100-20= 80.
* 80=Frames per bar.
* What is the frame count at the beginning of bar 3?
* 20 (because the music starts after 20 frames) + 2 multiplied by 80 (two bars have past). = 180.

I could have done this a lot more simply by printing the frame count at these specific points and then translating shapes by these values however I wanted my code to be somewhat adaptable to other songs.

Date: 21 April 2021.

Today I started implementing the elements arrays and testing them for further values such as bar2, bar3 and bar4. For example, shown below.

Text

Description automatically generated with medium confidence

However, upon testing especially for bar3 and bar4 values I encountered some problems. Firstly, the shapes were being unwantedly translated which caused them to appear in undesired positions. I realised that these shapes were being translated by the previous if statements for example for bar3 shapes bar2 translations. I wanted the translations to be localised within the if statements for each elements.bar(value). To fix this I realised quite quickly, from similar problems encountered previously in this project, I needed to use pop(), pull() objects to create fresh transformation states. Below is an example which includes my ‘treb’ triggered transformations and pull(), pop() objects to contain the transformations.

Text

Description automatically generated

Once this issue was dealt with, I did some more tests and realised that shapes drawn especially using elements.bar3 and elements.bar4 were visually out of time. I conducted further tests to work out what the issue was. I used the code shown below to print run time if a peak was detected and counted and identified what this value was for the first beat of the third bar. I did this in accordance with printing ‘elements.bar3.time’ and realised these times were different. Therefore, there was definitely a problem in the timing feature of my code.

Text

Description automatically generated

I used this same feature of printing run time when a peak was detected and identified what relative values these were for bar1, bar2, bar3.

|  |  |
| --- | --- |
| Time | Bar start |
| 25107 | 3 |
| 13473 | 2 |
| 1538 | 1 |

Bar 3 – Bar 2 = 25107 – 13473 = 11634

Bar 2 – Bar 1 = 13473 – 1538 = 11935

11634 is not equal to 11935. Therefore, the length of different bars is not entirely the same. I had not previously thought about this but because the song which I have created this visual art for is a jazz live song the repetitions of bars will not be exactly in time like an electronic techno track might be.

Therefore, I am going to have to re-evaluate my method for timing in this project. My ‘bar\_time’ and other linked variables are not going to work especially over the course of the track. This on time functionality is one of the most important features in my project. Instead, I will try using adapted clocking and peak detect methods to avoid this change in time lengths between bars being a problem.

I did some test of parameters within the peak detect object to refine its functionality.

Had to adjust frames per peak it was detecting too many peaks quickly.



Graphical user interface, text, application, chat or text message

Description automatically generated

Date: 26 April 2021.

Now I am going to attempt to increase the efficiency of my system by using functions. This is because I realise that I have quite a lot of repeated code so I want to clear this up before I continue adding things to my project.

I attempted to turn the code shown below into a function which I could recall for each of the sections of lines I wanted to create.

Text

Description automatically generated with medium confidence

I converted this code in a function with parameters ‘time’ and ‘space’. ‘time’ replaced ‘elements.bar1.time’ and ‘space’ replaced ‘Linetranslate’. I would then be able to recall this function assigning different values to the parameters using the ‘elements’ associative array. For example for creating a line composition at the beginning of bar two I would call line\_composition(elements.bar2.time, elements.bar2.space).

However, there was a problem upon testing the line compositions were not translated to the right place. Upon testing and realising that ‘Linetranslate’ was undefined I noticed that I had included part of the code assigning ‘Linetranslate’ within the function. This was wrong as I would not be able to use ‘Linetranslate’ as a part of a parameter for the function if it was not assigned its correct value before the function was called. Therefore, I removed this section from the function shown above in the blue semi-circle/bracket and placed it outside the function.

Therefore, this function is shown below.

Text

Description automatically generated

Therefore, I can use the code below to call on this function using the associative array ‘elements’ to fill the parameters.

A screenshot of a computer

Description automatically generated with medium confidence

Date: 30 April 2021.

Today I want to add some new elements to the overall piece, because I only have new action until about halfway through the track. I’m going to add a conditional rotation to the ‘line\_compositions’ so they rotate about the Z axis after a certain time. I wanted the rotation to be related to the music. So, I attempted using the code below.



This meant that the line would move at a rate of run time divided by the different in time between bar 4 and bar 2. However, I during testing I noticed that it was starting this rotation off axis. I wanted the line to start rotating from its position in the 3D space and not jump to a new one and start rotating from there. Just to note I already had been using a conditional IF statement to start this rotation.



I tested individual values such as ‘millis()’, ‘elements.bar4.time’ and ‘elements.bar2.time’ and also looked at the p5.js ‘rotateZ’ reference page available at: https://p5js.org/reference/#/p5/rotateZ (Accessed: 30 April 2021). I realised through these tests and research that if I want to start the rotation from the original position then the argument past into ‘rotateZ’ needs to be ‘0’ when it starts to rotate. I decided to do this by making the numerator ‘0’. I’ve used the code below.

Graphical user interface, text

Description automatically generated

Noting importantly that ‘rotateVal’, which is a fixed variable containing the value for ‘millis()’ when this section of the program is run, is subtracted from ‘millis()’ in the ‘rotateZ’ argument. Therefore the argument for ‘rotateZ’ will start at zero as ‘rotateVal’ = ‘millis()’ and will increase rotating the line as ‘millis()’ increases.

Also, to note I’ve included another pair of conditionals to stop the line rotating after half a rotation or ‘PI/2’. Shown by the orange arrows.

Date: 9 May 2021.

Changed the camera move speed to 0.5 in the z axis because I wanted it to move slower and keep the shapes on the screen for longer. To make this work I also changed ‘Linetranslate’ to = ‘frameCount/2’ rather than just ‘frameCount’. This influences the space part of the elements associative array.

A picture containing diagram

Description automatically generated

I removed lots of necessary elements of my code, such as unused variables.

Date 10 May 2021.

I worked out how I wanted to implement colour into my program. I would analyse the music and work out what frequency was loudest. I would then see what musical note this frequency represented and then implement colour using a specific note to colour conversion. First, I needed to work out how to analyse the music to find a max frequency. I knew it would be related to the ‘P5.FFT’ tools, and a found that specifically the ‘analyze()’ object was of relevance from reading its description in the p5.js references FFT page available at: https://p5js.org/reference/#/p5.FFT (Accessed: 10 May 2021). However, I wasn’t entirely sure how to utilise the object to search through the different bins. I watched a video by *The Coding Train* that was very helpful ‘17.11: Sound Visualization: Frequency Analysis with FFT - p5.js Sound Tutorial’, available at: https://www.youtube.com/watch?v=2O3nm0Nvbi4 (accessed: 10 May 2021). This video explained to me how to use a for loop to search through the bins of the ‘analyze’ object. I used code from the video up to 00:06:27 shown below.

Text

Description automatically generated

Text

Description automatically generated

I added this IF statement. This conditional means that only if the new bin which is being analysed is larger than the value of the largest bin analysed so far will it be stored in the value ‘freq\_heighs’. Therefore, what this means is that once the for loop has cycled through all the bins, the highest value bin will be stored in ‘freq\_heighs’ and what bin this is, is identified through the ‘i\_val’ variable which is assigned to ‘i’ (the number of iterations through the for loop). I then planned to use the variable ‘i\_val’ and work out what the mid-point frequency of the different bins ‘i\_val’ represented were. ‘i\_val’ would control the colour. However, this code didn’t quite work yet because as the for loop would start again for another instance in time ‘freq\_heighs’ and ‘amp ’would be at a max value. Therefore, I added another if statement at the end of the for loop.

Text

Description automatically generated

This re-sets these variables ‘freq\_heighs’ and ‘amp’ to zero at the for loop just before it finishes.

Next, I printed ‘i\_val’ outside of the for loop and noted down all the values which were output. I then conducted some calculation to work out what note would be appropriate to assign to these different bins.

Example calculations:

20,000 – 20 = 19980 = range

19980/1024(bins) = 19.51171875 Hz per bin. I looked at the different bins – printed to the console and worked out the mid-point of these. What note would this value be closest to.

E.g., bin 1

19.51171875 + (19.51171875/2) (this is the mid-point value of a bin)) = 29.267578125

29.267578125 + 20(lower bound of bins frequency) = 49.267578125.

49.267578125 is the midpoint of bin one.

Using <https://pages.mtu.edu/~suits/notefreqs.html> (Accessed: 10 May 2021.) This is closest to G.

|  |  |  |  |
| --- | --- | --- | --- |
| Bin | Mid-point Freq (5. sf) | Note | Colour |
| 0 | 29.756 | A#0/Bb0 | Red (violet) |
| 1 | 49.268 | G1 | Blue |
| 2 | 68.779 | C#2/Db2 | Orange |
| 3 | 88.291 | F2 | Green |
| 4 | 107.804 | A2 | Violet |
| 5 | 127.314 | C3 | Red/Orange |
| 6 | 146.826 | D3 | Orange/Yellow |
| 7 | 166.338 | E3 | Yellow/Green |
| 11 | 244.385 | B3 | Red |
| 12 | 263.896 | C4 | Red/Orange |
| 13 | 283.408 | C#4/Db4 | Orange |
| 14 | 302.920 | D#4/Eb4 | Yellow |
| 15 | 322.432 | E4 | Yellow/Green |
| 16 | 341.943 | F4 | Green |
| 17 | 361.455 | F#4/Gb4 | Blue/Green |
| 18 | 380.967 | F#4/Gb4 | Blue/Green |
| 19 | 400.479 | G4 | Blue |
| 20 | 418.990 | G#4/Ab4 | Violet |
| 21 | 430.502 | A4 | Violet |
| 22 | 459.017 | A#4/Bb4 | Red |
| 25 | 517.549 | C5 | Red/Orange |
| 43 | 868.760 | A5 | Violet |
| 54 | 1083.389 | C6 | Red/Orange |

I then used the function shown to call these different colours when different bins were detected as maximums.

Text

Description automatically generated with medium confidence

Text

Description automatically generated

In addition, I wanted the ability to make the shapes/lines further out from centre display this colour at a slightly later time, in order to create a sense of depth. For this is used the function shown below.

Graphical user interface, text, application

Description automatically generated

Here I set ‘clock\_change’ as zero and also implemented a conditional so that if ‘clock\_change’ equals zero then ‘clock\_change\_time’ = ‘millis()’ or the program run time at that moment. Then if this occurred ‘clock\_change’ was increased by one. The effect this has is that the if statement only runs once in this instance of the function and ‘clock\_change\_time’ remains constant. This is important because to create the delay I need a set time to measure time against. Then I used the second if statement to change the colour if the run time ‘millis()’ has surpassed the static ‘clock\_change\_time’ plus the ‘delay’ parameter. The colour is changed when this happens.

Date: 12 May 2021.

I took the ideas from the central rotating lines and copied them later in the music. However, using the formula for space that I had previously been using, I realised the result was not what it should be.

Logo

Description automatically generated

This code was related to the timing which were not accurate for some periods. I adjusted the code and similar to below.

Text

Description automatically generated