**Chapter 4 - Implementation**

**chapter 4.1 - Choice of language**

Language choice should be considered with proper thought as the whole project could take longer and become more complex if we choose an inappropriate language. Before considering the language i’m going to use, i have to consider what i want to achieve. if we know clearly what we want to achieve then language choice becomes more clearer.

The objective was to make a software for teaching pixel group processing, knowing this objective i know that i will need to have a GUI so my language of choice has to have a GUI library. Another thing i had to consider was i needed to teach people so theres going to be some text element describing the content and it needs to be accessible. it wouldn't make sense to design this in C++ or java even thought we could its these languages are not designed for that sort of thing and its not relatively portably unless compiled to an executable.

Since my application has to be heavily interactive to convey our ideas across to the learner, i thought of javascript which can be used to to make manipulations of the HTML DOM and its fairly accessible via a web browser so the learner doesn’t have to waste time downloading files and there wouldn’t be any cross platform issues.

in addition with the release of HTML5 theres lots of support for interactivity and new features such as canvas’s which i will be making use of in my project.

**frameworks**

Using a framework will allow me to streamline my production and allows me to have more freedom to think of the design rather than reinventing everything from scratch which is somewhat bad because the code is not tested as extensively as those in existing frameworks and framework libraries tend to have more optimised code meaning i can get better performance.

Since i am using javascript, i came across a javascript framework called P5.js it is essentially like the more known processing but its integrated with Javascript code. P5.js is a fairly good framework particularly aimed at Educators and artists. i choose this framework because it fits in directly with what my project is trying to achieve (educate) and i can express myself more vividly.

just to show you how easy it is to draw something onto the screen i will demonstrate a pure javascript and a P5.js equivalent.

Code

1

2

3

4

5

6

7

8

9

10

11

12

13

function setup(){

createCanvas(400,400);

backgroundcolor('red');

}

function draw(){

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Code

<!DOCTYPE html>

<html>

<body onload='myFunction()'>

<script>

function myFunction() {

var canvas = document.createElement("canvas");

canvas

var ctx = canvas.getContext("2d");

ctx.fillStyle = "red";

ctx.fillRect(10, 10, 400, 400);

document.body.appendChild(canvas);

}

</script>

</body>

</html>

P5.JS

As you can see the P5.JS version is much more clean and declarative rather than a more imperative approach of specifying what we want to do.

**chapter 3.2 - Kernel convolution Algorithm**

Design is very important in this project as the appropriate design is the only way for this project to be successful on the user end. in the following paragraphs i will be detailing over my algorithms and design choices.

First and foremost the main pixel processing algorithm will be made, this is the core of my overall project but not the most important part as it would be useless with a poor web design.

Essentially a kernel convolution is basically taking a pixel from the source image and doing some multiplication with nearest neighbours and outputting it to a target source (as illustrated in fig 1).

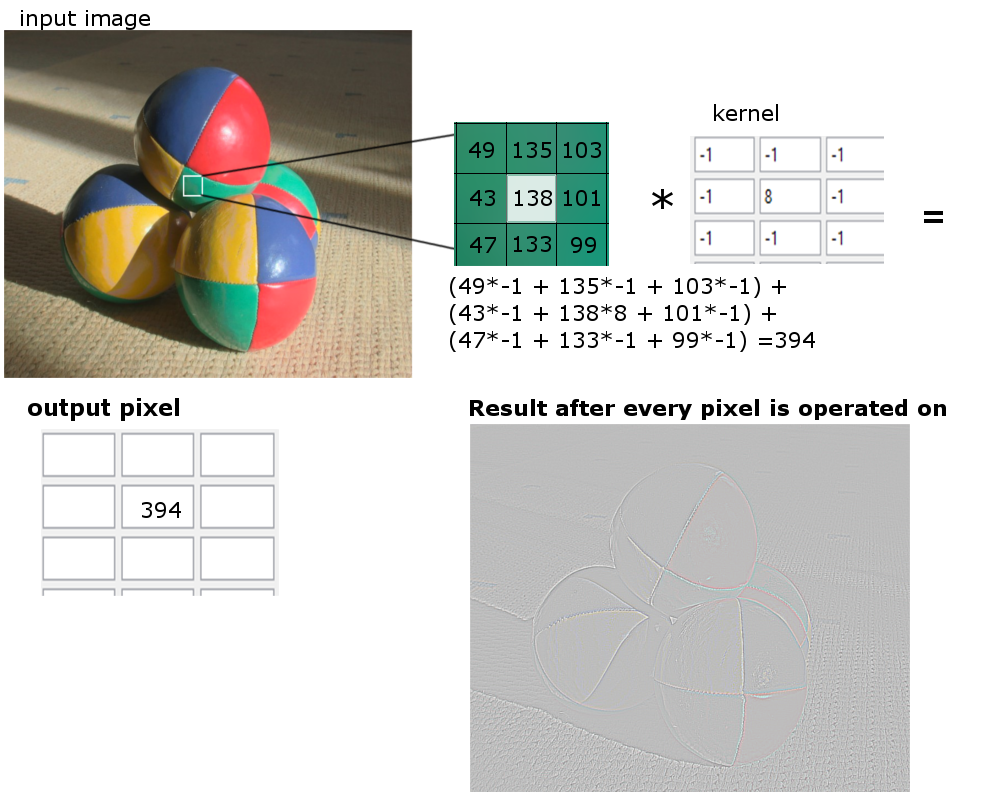


fig 1

We must output the image to a different source because if we put it back into the original image then it would cause side effects as we will be going over pixels that we just edited. my way to tackle this was to output my edited pixels to a HTML canvas that was introduced in HTML. I will be making extensive use of this element in my project.

Now that we know a little bit about pixel convolution i can describe how my algorithm works. first we have to load the image into javascript so that we have all the data we need we can use the getImageData method to get all the pixel values. once we have all the values then we can precede in the convolution algorithm once thats done we just output the new pixel data array into the output source which would be our canvas.

The pseudocode for the algorithm is as follows i will only show the Pseudocode for generating the top left nearest pixel, as its quite repetitive;

1

2

3

4

5

6

7

8

9

10

11

12

13

PseudoCode

FOR i = 0 to ImageArray.length, i = i + 4

width = width - 1

red,green,blue = i , i+1 , i+2;

TopLeftR = imageArray[reds -width]

TopLeftG = imageArray[green-width]

TopLeftB = imageArray[blue-width]

newRed = (topLeftR \* 1) + (topMidR \* 0) + (topRightR \* -1) + (midLeftR \* 0) + (middleR \* 0) + (midRightR \* 0) + (botLeftR \* -1) + (botMidR \* 0) + (botRightR \* 1);

Put new red into a new imageArray

canvas.putImageData(newImageArray)

This algorithm is very simple in nature but essentially what its doing is taking the imageArray which separate each pixel sequentially by using 4 indexes for red green blue alpha (alpha is ignored in this chapter) hence why i incremented the loop by 4 every time its ran. the red would be the ith index followed by green being i+1 index, blue being i+2 index. This just wraps back round and the cycle repeats for the next pixel until we have looped out of all the array. this is great for performance because we are not checking every array so its 4 times faster than usual but i still think with tinkering or using a completely different approach, better performance can be reached.

My method of finding the neighbouring pixel is basically wrapping the index round till i get the index directly above it or whatever neighbour i’m trying to find, for example if the width of the image is 800 and i’m currently on the 1600 i could get the above pixel by subtracting 800 and i could find the bottom by adding 800. Using this approach i was able to come up with ways to find the top left, top middle, top right .e.t.c pixels.

line 7-8 is the calculation we perform to calculate the new pixels value then once we’ve calculated the red green and blue channel we can output it as a pixel then move on to the next pixel until we reach end of the loop.