Lab 02- Image and color basics

Discussion is based on F.S. Hill Chapter 02,10 and color theory

BSCS – 514 Computer Graphics
Course Supervisor Dr. Humera Tariq

Lab 02 Objectives /Tasks

1. Recall canvas and add click handler
2. Image Browser in Java script
3. HTML vs. Canvas
4. Statement let vs. var
5. File handlers
6. Image and histogram plot
7. Discussion of Maze Modeling/Generation

Step I : Recall Canvas and click handler

Name	Date modified	Type
_Reading Material	21/08/2019 19:06	File folder
scripts	21/08/2019 19:16	File folder
2019 CG Lab 02.pptx	21/08/2019 19:00	Microsoft PowerP...
index.html	21/08/2019 19:17	HTML Document

index.html | color.js | keyboard_timer.js

```
1
2
3   window.onLoad = myInit(); // First peform initialization
4
5
6   function myInit(){
7
8
9       myDisplay();
10  }
11
12
13  function myDisplay() {
14
15      alert("inside myDisplay function"); // short cut to avoid onload ??????
16
17
18  }
```

Step 2. Button and Click Handler

You may use the HTML <input type="submit" /> tag and style it via CSS, which would be the standard way. Any mouse events and so on are then handled without further configuration by the browser.

We are not using css in CG Course

A button on a canvas is simply a Rectangle. We need to write following functions to add and handle button click:

- ✓ Variable or Function to draw Rectangle
- ✓ Function to check whether a point is inside a rectangle
- ✓ Function to get the mouse position on canvas
- ✓ Binding the click event with the canvas

Rectangle and isInside(.....)

```
function myDisplay() {

    alert("inside myDisplay function"); // short cut to avoid onload ??????

     var cvs = document.getElementById("mycanvas")
     var ctx = cvs.getContext('2d');

     //The rectangle should have x,y,width,height properties
     var rect = {
          x:250,
          y:350,
          width:200,
          height:100
    };

    }

//Function to check whether a point is inside a rectangle
function isInside(mouse_pos, rect){
    return pos.x > rect.x && pos.x < rect.x+rect.width && pos.y < rect.y+rect.height && pos.y > rect.y
}
```

1

2

getMousePostion(...)

```
//Function to check whether a point is inside a rectangle
function isInside(mouse_pos, rect){
    return pos.x > rect.x && pos.x < rect.x+rect.width && pos.y < rect.y+rect.height && pos.y > rect.y
}

function getMousePos(canvas, event) {
    var rect = canvas.getBoundingClientRect();
    return {
        x: event.clientX - rect.left,
        y: event.clientY - rect.top
    };
}
```

2

3

Binding click event with canvas

Error No.1 Reference to canvas not found

Error No.2 Reference to pos not found

Confusion: Inside outside test seems confusing as our rectangle is invisible

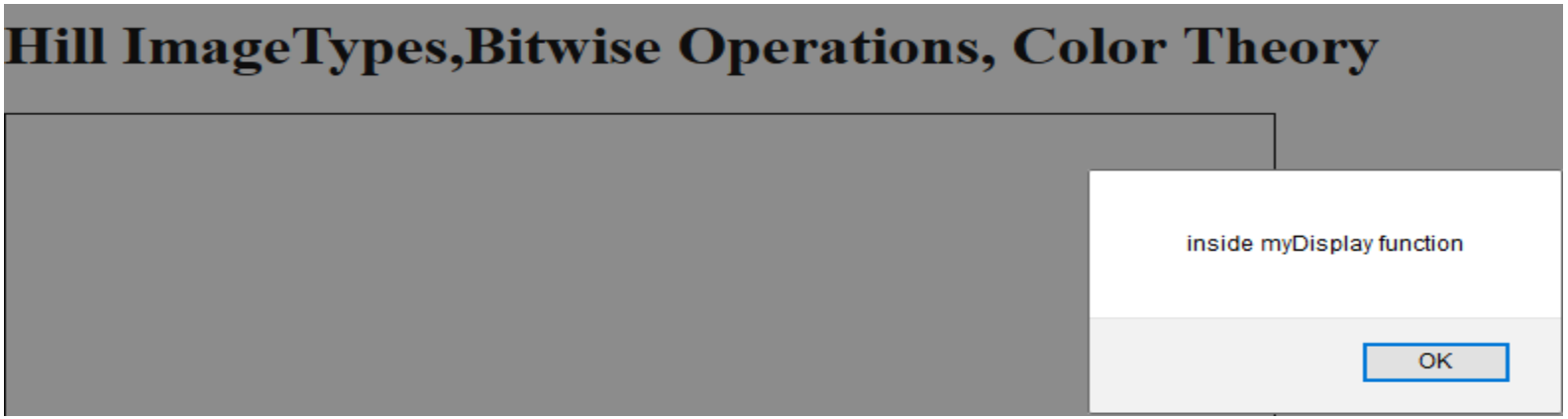
```
function myDisplay() {

    alert("inside myDisplay function"); // short cut to avoid onload ??????

     var cvs = document.getElementById("mycanvas")
     var ctx = cvs.getContext('2d');

     //The rectangle should have x,y,width,height properties
     var rect = {
          x:0,
          y:0,
          width:200,
          height:100
    };

    //Binding the click event on the canvas
    cvs.addEventListener('click', function(evt) {
    var mousePos = getMousePos(cvs, evt);

    if (isInside(mousePos,rect)) {
        alert('clicked inside rect');
    }else{
        alert('clicked outside rect');
    }
    }, false);

    }
```

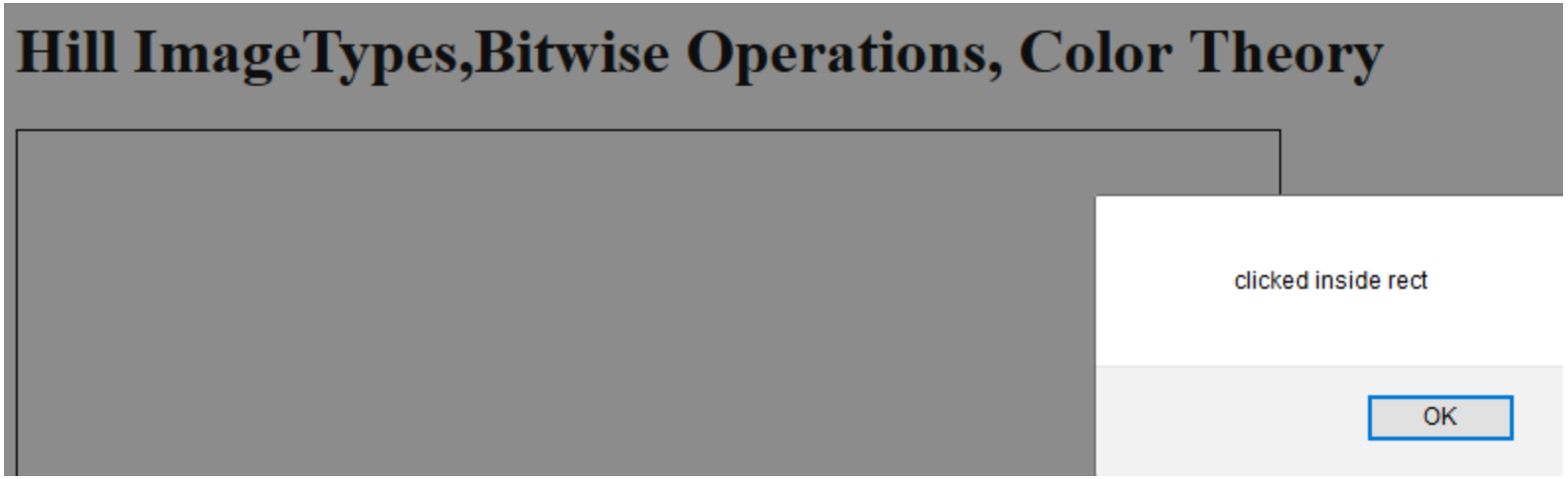
4

Click handler is inside myDisplay()

Refresh index.html in browser

5





Make Rectangle Visible

6

```
//create your shape data in a Path2D object
 const path = new Path2D()
 path.rect(0, 0, 200, 100)
 path.rect(25,72,32,32)
 path.closePath()

//draw your shape data to the context
ctx.fillStyle = "#FFFFFF"
ctx.fillStyle = "rgba(225,225,225,0.5)"
ctx.fill(path)
ctx.lineWidth = 2
ctx.strokeStyle = "#000000"
ctx.stroke(path)
```

Hill ImageTypes,Bitwise Operations, Color Theory

Step II : <input> tag for image loading

```
<body>

<h1> F.S.Hill ImageTypes,Bitwise Operations, Color Theory  </h1>

<input type="file" id="imageFile" accept=".png, .jpg, .jpeg"></input>

<canvas  id="mycanvas" width="640" height="480" style="border:1px solid"


<p>  Enter default content here </p>

</canvas>

<!-- <p> Above this is canvas area </p> -->
 <p> Press Enter to play your first animation </p>


<!--<script  src="scripts/Excercise.js"> </script> -->
<script  src="scripts/color2.js"> </script>

</body>
```

1

2

handleFiles(...)

```
function myInit(){

    document.getElementById("imageFile").addEventListener("change", handleFiles);

    myDisplay();
}
```

2

```
function handleFiles() {
  var theGoods = document.getElementById('imageFile').files[0];
  var img = new Image();
  var reader = new FileReader();

  reader.addEventListener("load", function() { img.src = reader.result; });

  img.onload = function() { calcAndGraph(img); }

  if (theGoods) { reader.readAsDataURL(theGoods); }
}
```

3

handleFiles(...)

```
function myInit(){

    document.getElementById("imageFile").addEventListener("change", handleFiles);

    myDisplay();
}
```

2

```
function handleFiles() {
  var theGoods = document.getElementById('imageFile').files[0];
  var img = new Image();
  var reader = new FileReader();

  reader.addEventListener("load", function() { img.src = reader.result; });

  img.onload = function() { calcAndGraph(img); }

  if (theGoods) { reader.readAsDataURL(theGoods); }
}
```

3

4

calAndGraph(...)

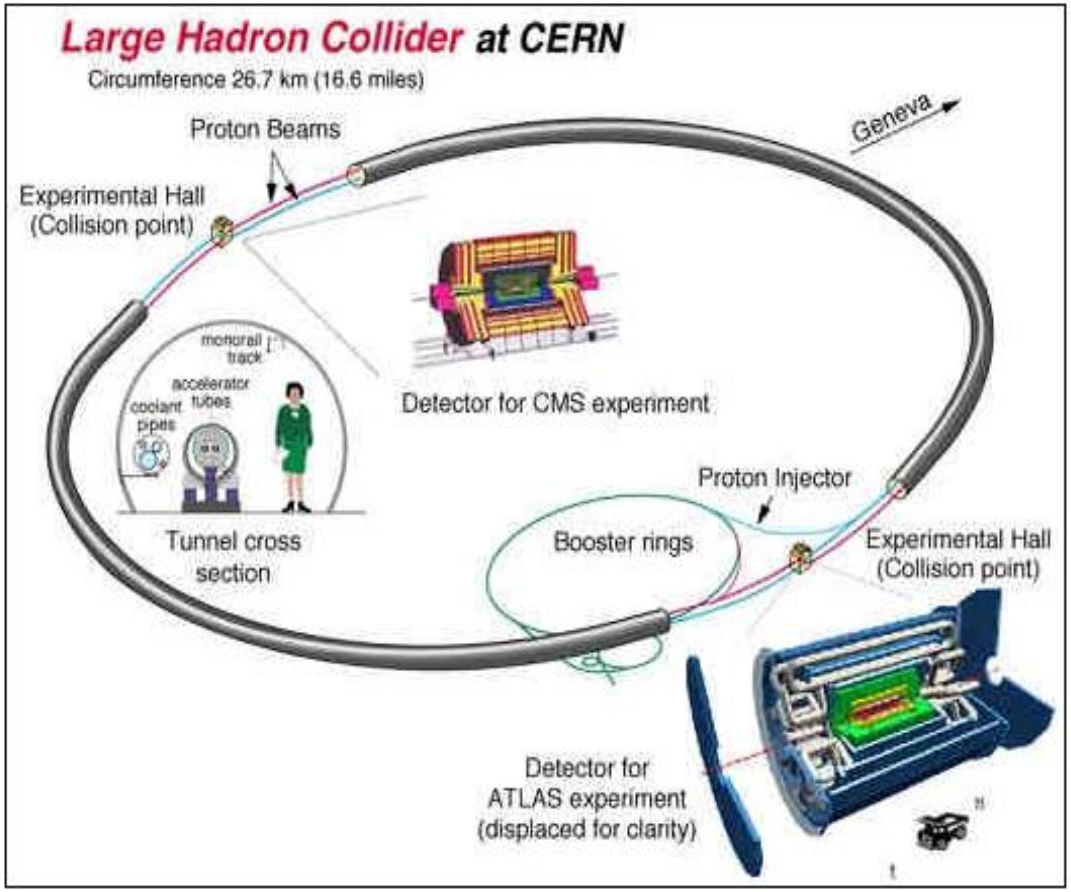
5

```
function calcAndGraph(img) {
  let rD={}, gD={}, bD={};
  let cv = document.getElementById("mycanvas");
  let ctx = cv.getContext("2d");
  cv.width = img.width;
  cv.height = img.height;
  ctx.drawImage(img, 0, 0);
  const iD=ctx.getImageData(0, 0, cv.width, cv.height).data;

 for (var i=0; i<256; i++) { rD[i]=0; gD[i]=0; bD[i]=0; }

  for (var i=0; i<iD.length; i+=4) {
    rD[iD[i]]++;
    gD[iD[i+1]]++;
    bD[iD[i+2]]++;
  }
  histogram({rD, gD,bD});
}
```

6

F.S.Hill ImageTypes,Bitwise Operations, Color Theory

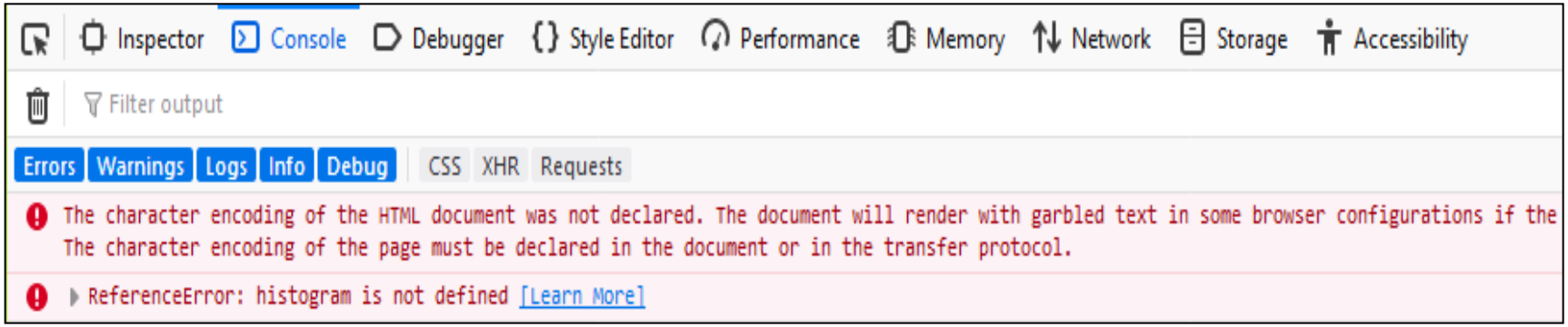
Browse... LHC.jpg



histogram(…) simplest code

https://thmsdnnr.com/projects/2018/03/09/draw-photo-histograms-d3-canvas.html

7



Lab Practice 02

- Try to code for display image histogram. See a good sample output
  http://mihai.sucan.ro/coding/svg-or-canvas/histogram.html
- Study Bitwise operations from chap 10 and try to code
- Study RGB and HSV color space from Book and try some sample code

Major Assignment 01

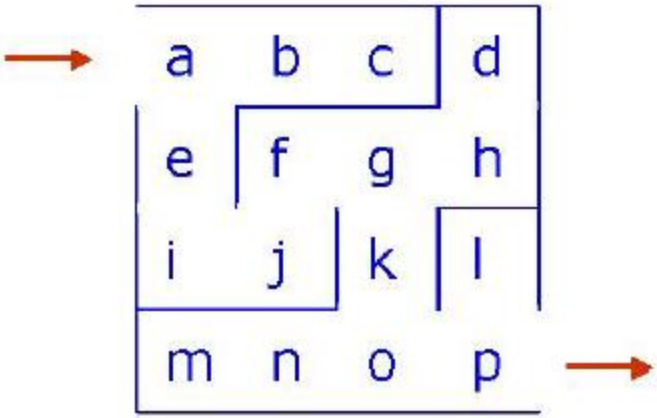
Case Study 2.7 Building and Running Mazes

Due Week: Last week of August 2019

Note: Submit in your lab timings

A maze is a graph with two special nodes.

Maze Modeling/Generation, Solving /Path



$$\left\langle \left\{ \begin{array}{l} \langle a,e\rangle, \langle e,i\rangle, \langle i,j\rangle, \\ \langle f,j\rangle, \langle f,g\rangle, \langle g,h\rangle, \\ \langle d,h\rangle, \langle g,k\rangle, \langle a,b\rangle \\ \langle m,n\rangle, \langle n,o\rangle, \langle b,c\rangle \\ \langle k,o\rangle, \langle o,p\rangle, \langle l,p\rangle \end{array} \right\}, a, p \right\rangle$$