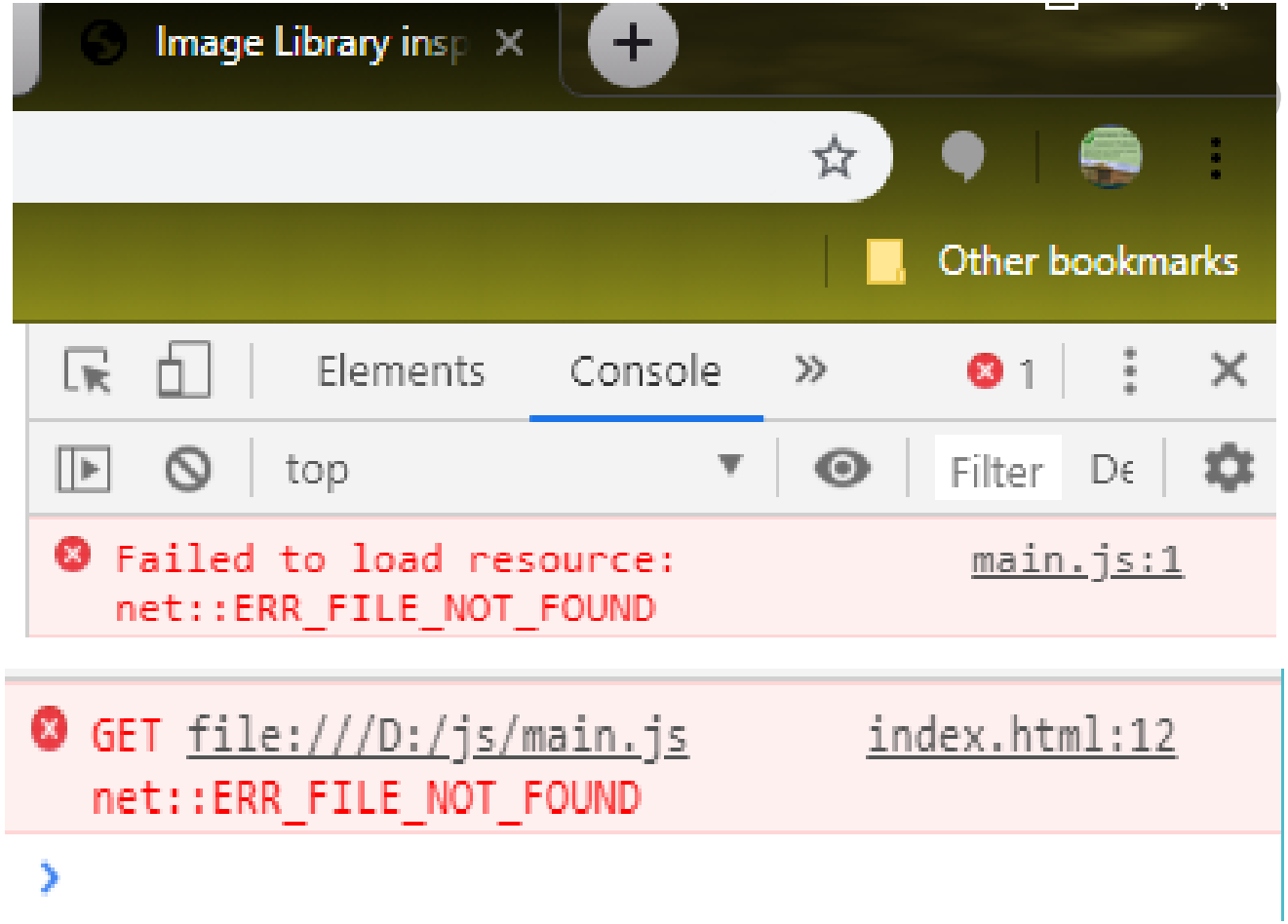
Lab 06- Lab 07
Modeling, Animation and Boundary Collision

Discussion is based on F.S. Hill Chapter 02,03,04,10

Revision 01:
- Running HTML code through Server i.e. http:// protocal

Error Observed When I run using file with out server or http protocol

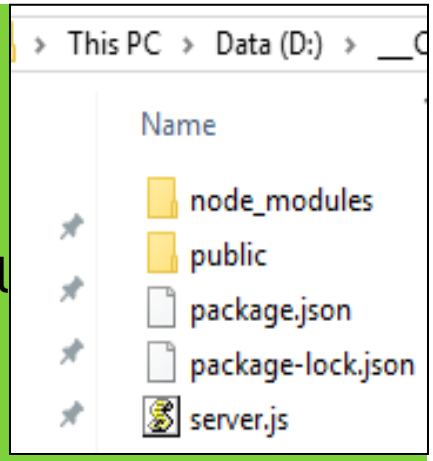


Lab 06-Lab 07 Recall server installation and code structure

I already show you how to set up a JavaScript project from scratch in Lab03-Lab04. Afterward, we can continue by advancing the project to a frontend (e.g. WebGL,React.js) or backend (e.g. Node.js with Express) application.
You are advised to recall following points and visit

1) Download node.js from web. Then install it
2) Create a directory for the code let say D:/Lab0304
   (make sure directory name dosen't have space)
3) Now go to Lab0304 Directory
4) Run the command npm init then press enter until it will
   write .json file.
5) Then run the command npm install express --save
6) Create a folder named public in directory Lab01
7) Create three folders named (css, images, js) in public folder. Move the images and js files in corresponding directories
8) Create index.html file in public folder
9) Change the js files path in index.html to the following

```
<script type="text/javascript"  src="/js/global.js"> </script>
<script type="text/javascript" src="/js/color2.js"> </script>
<script type="text/javascript" src="/js/histogram.js"> </script>
```



Alternatives to run .js application on server

1) Install live server extension on visual code
2) Just Right click on your HTML code and click on “run on live server”

Note: The extension can be installed by using extension tab available on VS code.

https://threejs.org/docs/#manual/en/introduction/How-to-run-things-locally

1: Run server.js

```
D:\CGLab0304>node server.js
Server started at http://localhost:8081
Press CTRL + C to shutdown
```

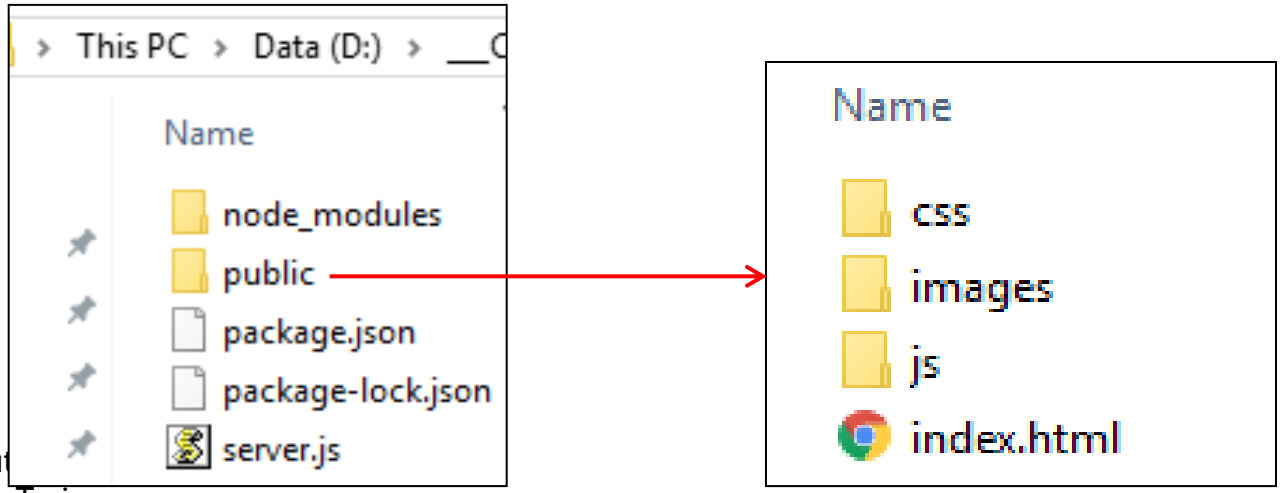
2: View html locally in browser

Revision 02:
- Modern Java script Project Code Structure and folder organization

Getting Started: Must observe Project Structure

I wanted the file structure for an intial project to be simple:

```
.
├── package.json # lists dependencies for easy re-install
├── src
│   ├── index.html # html to use as template for generated output html
│   ├── js
│   │   └── main.js # entry point for our application
│   └── shaders
│       └── ... # glsl files go here
└── node_modules/ # folder containing our dependencies
```



Recall index.html

```html
<html>
    <head>
        <title>Image Library inspired by FS-Hill</title>


    </head>
    <body>
        <canvas id="glcanvas" width="640" height="480" >
                 Your Browser does not support html canvas.
        </canvas>


        <script type="text/javascript"  src="/js/main.js"> </script>
    </body>
</html>
```

Revision 03:
Highest level CG Pipeline perceived by Dr. Humera Tariq

CG Pipeline: Action Phase

Action | Camera | Lights

1 Modeling

- Equation of Line, Circle…..
- 2D/3D shapes/curves mathematics and construction
- 2D Image is an example of readymade model (Binary, Grayscale, RGB, Hyperspectral)

Animation 2

- Change of Position a.k.a Translation (classically an addition problem)
- Change of Size a.k.a. Scaling (classically a multiplication problem)
- Change of Orientation a.k.a Rotation (classically a trigonometric problem)

Its time to start coding!

main.js, car.js

Model = Car Image main.js
Animation = Change of position i.e. Translation

index.html | main.js | Point.js | Car.js

1

```
var canvas = document.getElementById("glcanvas");
context = canvas.getContext("2d");
```

```
//Make sure that image.jpg has been loaded before you call drawImage()?
//Because Canvas needs a preloaded image in order to draw/display it in itself.

var cultus= new Car(20,10);

 function update()
 {
   console.log(canvas.width);
   console.log(canvas.height); // just checking access to canvas
   cultus.advancePosition(0.91,0.4); // (dx,dy)
 }

function render()
{
  context.beginPath();
  context.clearRect(0,0, canvas.width, canvas.height);
  context.drawImage(cultus.img, cultus.x, cultus.y);
  context.closePath();
 }
```

3

Car

2

// has Position
float x,y or Point P
// is an image
Image img

// constructor (…)
// lerp (….)
//advancePosition(…)

Class Car and usage of lerp to advance position

car.js

```
// When you use class keyword then don't use function keyword inside class
// class keyword = function applied on all functions inside class
class Car {

    constructor(x,y) {

        alert("inside constructor");
        this.x = x;
        this.y = y;

       const img = new Image();
       img.src = 'images/car.jpg';
       img.onload = () => {
           context.drawImage(img, 0, 0);
      };
      this.img = img;

    } // end constructor


      lerp (start, end, amt){
      return (1-amt)*start+amt*end
    }

} // end class

Car.prototype.advancePosition = function(x,y)  // just checking prototype feature in js
 {
      this.x = this.lerp(this.x, this.x + x, 1); // keyword this is very important
      this.y = this.lerp(this.y, this.y + y, 1);  // keyword this is very important
 }
```

5

4

Build a mainLoop() or gameLoop() for animation

Chap3 Topic: Achieving smooth Animation

Cahp10 Topic: off screen memory

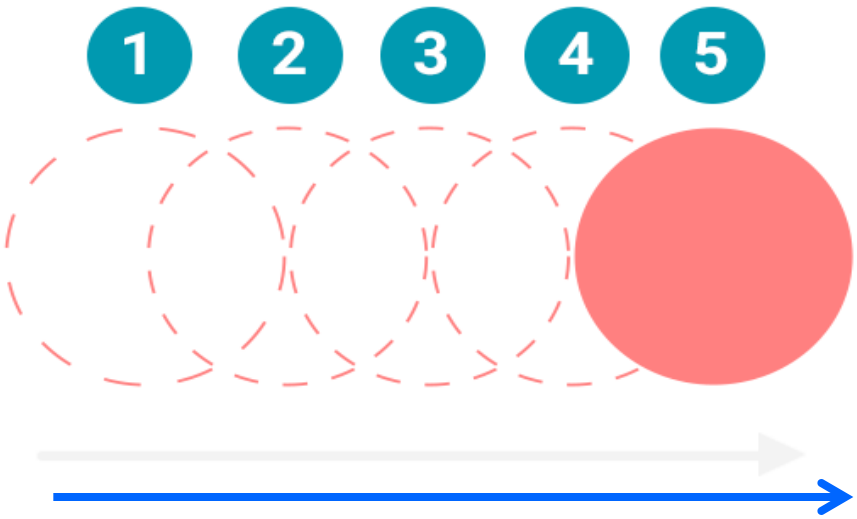
main.js

6

```
function gameLoop(){
  //requestAnimationFrame is a callback just like glutTimerFunc(1000,render,id) and glutIdleFunc()
  requestAnimationFrame(mainLoop);
  render();   // render () draws pixels on screen or output window
  update();   //update() function suffers from the issue that it is dependent on the frame rate.
};
```

7

```
// Start things off
requestAnimationFrame(gameLoop);
```

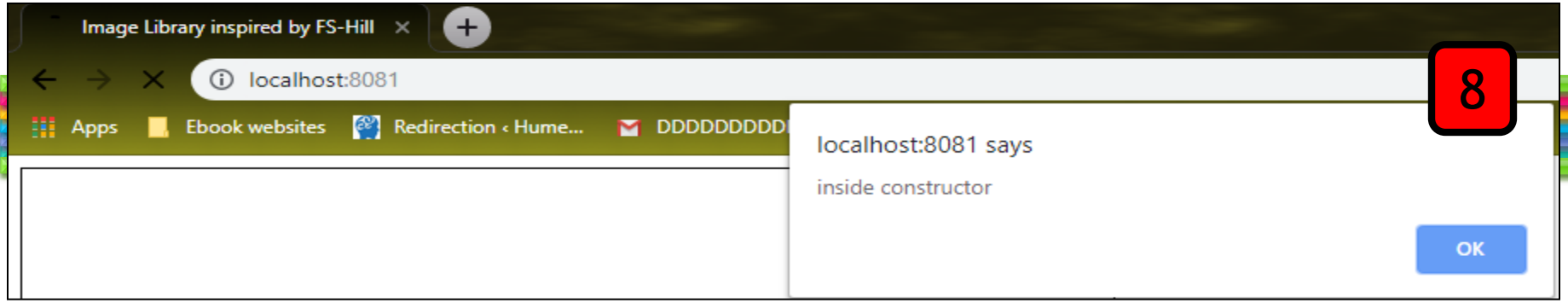


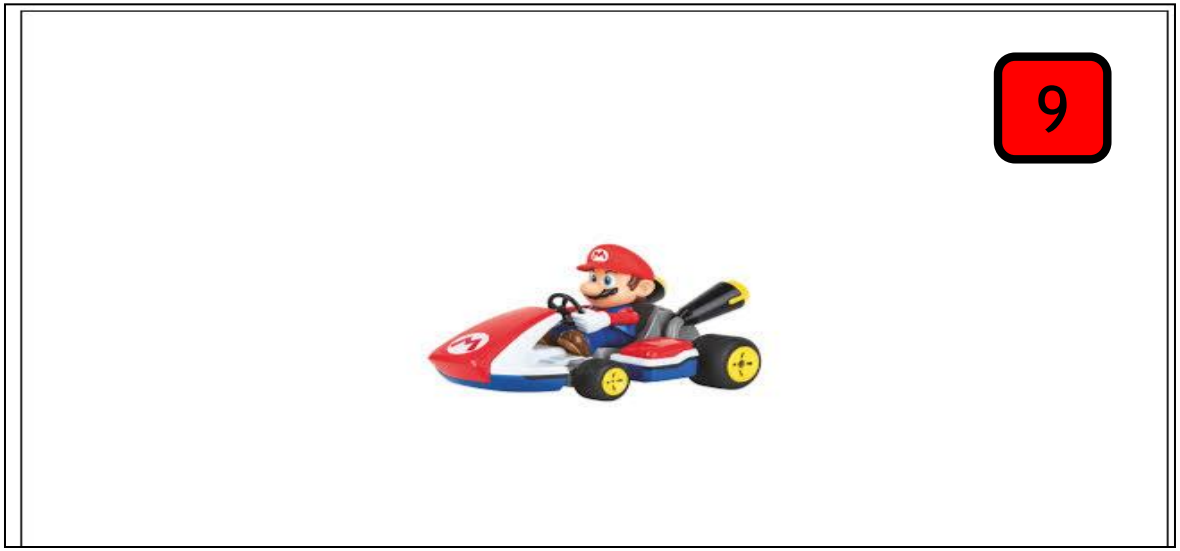
Questions:

1) What happens when mainLoop is renamed as gameLoop at later stage or viceversa ?
2) How you explain importance of explicit calling of callback requestAnimationFrame(mainLoop)

Problem : The car moves and become invisible at canvas or viewport boundary

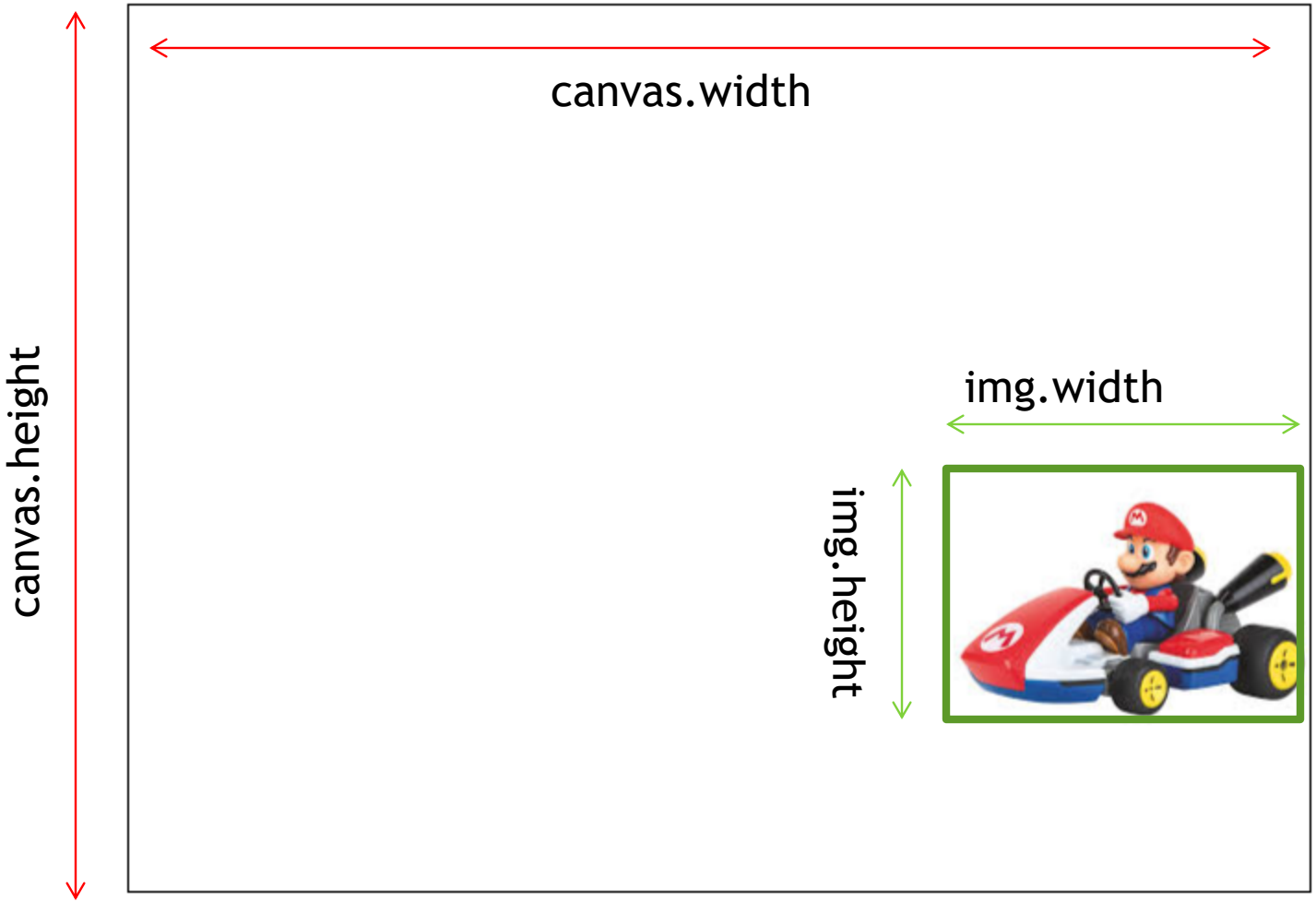
main.js





Solution : Stop the moving car at boundary i.e. boundary collision

main.js



10

Problem: To properly visualize collision detection, I have draw rectangle around image and face improper buffer clearing main.js

```
//draw Rectangle arouond image a.k.a bounding box
context.beginPath();
context.rect(cultus.x, cultus.y,cultus.img.width,cultus.img.height);
context.strokeStyle = 'black';
context.stroke();
context.closePath();
```

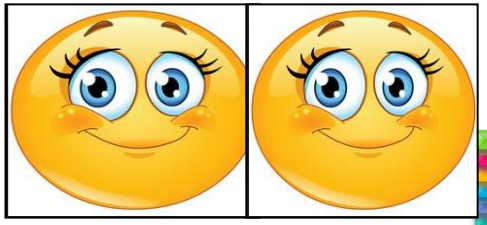
11

//The reason is that the destination canvas is not cleaned when blitting from the offscreen buffer.

Solution: Modify render() for proper screen clearing main.js

```
function clearCanvas(cvs) {
    const ctx = cvs.getContext('2d');
    ctx.save();
    ctx.globalCompositeOperation = 'copy';
    ctx.strokeStyle = 'transparent';
    ctx.beginPath();
    ctx.lineTo(0, 0);
    ctx.stroke();
    ctx.restore();
}
```

12



Basically, it saves the current state of the context, and draws a transparent pixel with copy as `globalCompositeOperation`. Then, restores the previous context state.

```
function render()
{
    //context.clearRect(0,0, canvas.width, canvas.height);

    //i = c.createImageData(canvas.width, canvas.height);
    //c.putImageData(i, 0, 0); // clear context by putting empty image data

    clearCanvas(canvas);
    context.drawImage(cultus.img, cultus.x, cultus.y);

    //draw Rectangle arouond image a.k.a bounding box
    context.beginPath();
    context.rect(cultus.x, cultus.y,cultus.img.width,cultus.img.height);
    context.strokeStyle = 'black';
    context.stroke();
    context.closePath();
}
```

13

Solution: Modify render() for boundary collision

And the problem begins…… main.js

Car.x and Car.y is continuously increasing and I am not able to stop it by simple boundary check at (0,0,640,480)

```
//draw Rectangle arouond image a.k.a bounding box
context.beginPath();
context.rect(cultus.x, cultus.y,cultus.img.width,cultus.img.height);
context.strokeStyle = 'black';
context.stroke();
context.closePath();

 //boundary collision detection for Left and Bottom Edge
if(cultus.x > canvas.width - cultus.img.width || cultus.y > canvas.heigh
    console.log(cultus.x);
  console.log(cultus.y); // just checking access to canvas
   context.translate(-1,-1); // stop car at boundary
}
//if(cultus.x < 0 || cultus.y > cultus.img.height)//will true at bottom most {
//if(cultus.x < 0 || cultus.y < cultus.img.height/4) //will true at bottom
if (cultus.x > 800 || cultus.y > 500 )  {

    console.log(cultus.x);
  console.log(cultus.y); // just checking access to canvas
   context.translate(-1,1); // stop car at boundary

}
```

14

cultus.x and cultus.y are continuously increasing even after applying translate(-1,-1)

Solution: Modify render() for boundary collision
And the problem begins…… main.js
less than zero doesn't work for me

```
//draw Rectangle arouond image a.k.a bounding box
context.beginPath();
context.rect(cultus.x, cultus.y,cultus.img.width,cultus.img.height);
context.strokeStyle = 'black';
context.stroke();
context.closePath();

 //boundary collision detection for Left and Bottom Edge
if(cultus.x > canvas.width - cultus.img.width || cultus.y > canvas.heigh
    console.log(cultus.x);
  console.log(cultus.y); // just checking access to canvas
   context.translate(-1,-1); // stop car at boundary
}
//if(cultus.x < 0 || cultus.y > cultus.img.height)//will true at bottom most {
//if(cultus.x < 0 || cultus.y < cultus.img.height/4) //will true at bottom
if (cultus.x > 800 || cultus.y > 500 )  {

    console.log(cultus.x);
  console.log(cultus.y); // just checking access to canv
   context.translate(-1,1); // stop car at boundary

}
```

14

First check runs absolutely fine but after that cultus.x and cultus.y are continuously increasing even after applying translate(-1,-1)

Problem persist and I have to put check multiple checks for every collision. This mainly happens due to combination of lerp(…) and translate(-1,-1)


```
//if(cultus.x < 0 || cultus.y > cultus.img.height)//will true at bottom most {
//if(cultus.x < 0 || cultus.y < cultus.img.height/4) //will true at bottom
if (cultus.x > 800 || cultus.y > 500 )  {

    console.log("The value of x is" + cultus.x);
  console.log("The value of y is" + cultus.y); // just checking access to canvas
   context.translate(-1,1); // stop car at boundary

}
if (cultus.x > 1200 || cultus.y > 900 )  {

    console.log("The value of x is" +cultus.x);
  console.log("The value of y is " + cultus.y); // just checking access to canvas
   context.translate(1,1); // stop car at boundary
}
if (cultus.x > 1400 || cultus.y > 1000 )  {

    console.log("The value of x is" +cultus.x);
  console.log("The value of y is" + cultus.y); // just checking access to canvas
   context.translate(1,-5); // stop car at boundary

}
```

379.19999999999567	main.js:85
799.8399999999962	main.js:84
379.59999999999565	main.js:85
800.7499999999961	main.js:84
379.9999999999956	main.js:85
The value of x is800.7499999999961	main.js:92
The value of y is379.9999999999956	main.js:93
801.6599999999961	main.js:84
380.3999999999956	main.js:85

825.3199999999953	main.js:84
390.799999999995	main.js:85
The value of x is825.3199999999953	main.js:92
The value of y is390.799999999995	main.js:93

Boundary Collision Problem
Before solving above problem lets focus a little on animation loop

Problem: How to start and end animation interactively i.e. on keypress or mouseclick

In the past, animations were performed using setTimeout() or setInterval(). You perform a little bit of an animation, and you call setTimeout() to repeat again this code in a few milliseconds from now:

The problem here is that in order for the animations to be smooth the browser often has to paint frames quicker than the screen can display then (most computer screens have a refresh rate of 60 frames per second or FPS). This results in unnecessary computation. Another problem with using setInterval or setTimeout is that the animations will continue to run even if the page is not visible to the user.

requestAnimationFrame() gives a more predictable way to hook into the browser render cycle. One can Create really smooth animations in JavaScript by having your animation using the requestAnimationFrame function. It is an equivalent to glutIdleFunc(…) or glutTimerFunc(…) if you are reading from text book.

Solution: Toggling or Switching logic to start and end animation interactively i.e. on keypress


```
var canvas = document.getElementById("glcanvas");
context = canvas.getContext("2d");

var animate = true ;
var animateID;

window.onkeydown = function() {

    animate = !animate; // flips or toggle the  state

    if(animate) window.requestAnimationFrame(mainLoop);

    else cancelAnimationFrame(animateID);
};
```

1

```
function mainLoop(){  // you can use any userdefined name for this function
  //requestAnimationFrame is a callback just like glutTimerFunc(1000,render,id) and glutIdleFunc()
   if (animate)
       animateID = window.requestAnimationFrame(mainLoop);
   render();   // render () draws pixels on screen or output window
   update();   //update() function suffers from the issue that it is dependent on the frame rate.
};

window.requestAnimationFrame(mainLoop); // start animation
```

2

Solution: Toggling or Switching logic to start and end animation interactively i.e. on mouse click

Onclick events must call functions like: onclick= click;" with the parenthesis at the end. Some browsers may try to submit on button clicks if you don't define type="button".

```
// register event handlers
window.onclick = click;
function click(evt) {
      animate = !animate; // flips or toggle the  state

     if(animate) window.requestAnimationFrame(mainLoop);

     else cancelAnimationFrame(animateID);

}
```

Check by clicking on canvas1

Solution: start and stop button

1

```
<body>

    <canvas id="glcanvas" width="640" height="480" style= "border:1px solid #000000" >
            Your Browser does not support html canvas.
    </canvas>

     <div id="animated" style="left: 549.5px;" background="green">Hello there.</div>
     <button onclick="start()">Start</button>
     <button onclick="stop()">Stop</button>

     <script type="text/javascript" src="/js/Point.js"> </script>
     <script type="text/javascript" src="/js/Car.js"> </script>
     <script type="text/javascript"  src="/js/main.js"> </script>
</body>
```

2

```
function start() {
            animate = true;
            starttime = Date.now();
            animateId = window.requestAnimationFrame(mainLoop);

        }
function stop() {
            if (animateId) {
                window.cancelAnimationFrame(animateId);
            }
        animate = false;

        }
```

3

```
function mainLoop(){  // you can use any userdefined name for this function
  //requestAnimationFrame is a callback just like glutTimerFunc(1000,render,id) and glutIdleFunc()
   //if (animate)
       //animateID = window.requestAnimationFrame(mainLoop);

   if (animate) {
       //elm.style.left = ((Date.now() - starttime) / 4 % 600) + "px";
       animateId = window.requestAnimationFrame(mainLoop);
   }

   render();   // render () draws pixels on screen or output window
   update();   //update() function suffers from the issue that it is dependent on the frame rate.
};
```

Output from start and stop button main.js

4

```
// register event handlers
/*
window.onclick = click;
function click(evt) {
     animate = !animate; // flips or toggle the  state

    if(animate) window.requestAnimationFrame(mainLoop);

    else cancelAnimationFrame(animateID);

}
*/
```

5

6

```
function mainLoop(){  // you can use any userdefined name for this functi
  //requestAnimationFrame is a callback just like glutTimerFunc(1000,render,id) and glutIdleFunc()
   //if (animate)
       //animateID = window.requestAnimationFrame(mainLoop);

   if (animate) {
       elm.style.left = ((Date.now() - starttime) / 4 % 600) + "px";
       //animateId = window.requestAnimationFrame(mainLoop);
   }

   render();   // render () draws pixels on screen or output window
   update();   //update() function suffers from the issue that it is dependent on the frame rate.
};
```

What is impact of commenting requestAnimation Frame

Back to Collision Problem

Recall Problem of writing multiple checks for every collision. This mainly happens due to combination of lerp(…) and translate(-1,-1)

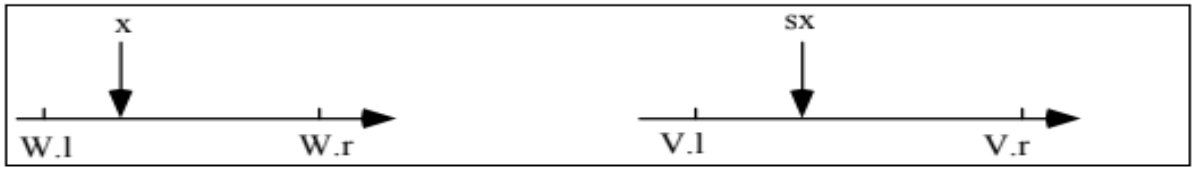

```
//if(cultus.x < 0 || cultus.y > cultus.img.height)//will true at bottom most {
//if(cultus.x < 0 || cultus.y < cultus.img.height/4) //will true at bottom
if (cultus.x > 800 || cultus.y > 500 )  {

    console.log("The value of x is" + cultus.x);
  console.log("The value of y is" + cultus.y); // just checking access to canvas
   context.translate(-1,1); // stop car at boundary

}
if (cultus.x > 1200 || cultus.y > 900 )  {

    console.log("The value of x is" +cultus.x);
  console.log("The value of y is " + cultus.y); // just checking access to canvas
   context.translate(1,1); // stop car at boundary
}
if (cultus.x > 1400 || cultus.y > 1000 )  {

    console.log("The value of x is" +cultus.x);
  console.log("The value of y is" + cultus.y); // just checking access to canvas
   context.translate(1,-5); // stop car at boundary

}
```

379.19999999999567	main.js:85
799.8399999999962	main.js:84
379.59999999999565	main.js:85
800.7499999999961	main.js:84
379.9999999999956	main.js:85
The value of x is800.7499999999961	main.js:92
The value of y is379.9999999999956	main.js:93
801.6599999999961	main.js:84
380.3999999999956	main.js:85

825.3199999999953	main.js:84
390.799999999995	main.js:85
The value of x is825.3199999999953	main.js:92
The value of y is390.799999999995	main.js:93

Solution: Chap 3 WW to VP Mapping Scaling and Shifting

How can A, B, C, and D be determined? Consider first the mapping for x. As shown in Figure 3.5, proportionality dictates that $(sx - V.l)$ is the same fraction of the total $(V.r - V.l)$ as $(x - W.l)$ is of the total $(W.r - W.l)$, so that



Computer Graphics Chap 3 09/21/99 5:38 PM page 4

Figure 3.5. Proportionality in mapping x to sx.

$$\frac{sx - V.l}{V.r - V.l} = \frac{x - W.l}{W.r - W.l}$$

or

$$sx = \frac{V.r - V.l}{W.r - W.l}x + (V.l - \frac{V.r - V.l}{W.r - W.l}W.l)$$

- Disable all previous multiple checks and try using new variables A, B, C and D.
-Enable following check and identify the collision problem.

Run this code and comment on problem faced in boundary collision

1

```
A = canvas.width/cultus.x;
C= -cultus.img.width;
B = canvas.height/cultus.y;
D = (canvas.height-cultus.img.height) - B*cultus.y;

/*
if(cultus.x > canvas.width - cultus.img.width || cultus.y > canvas.height - cultus.img.height){
    //console.log(cultus.x);
    //console.log(cultus.y); // just checking access to canvas
   context.translate(-1,-1); // stop car at boundary

   cultus.x = A*cultus.x+C; console.log (cultus.y)
   cultus.y = B*cultus.y+D;  console.log(cultus.y);
}
*/
```

2

And my logic from book rocks again .....

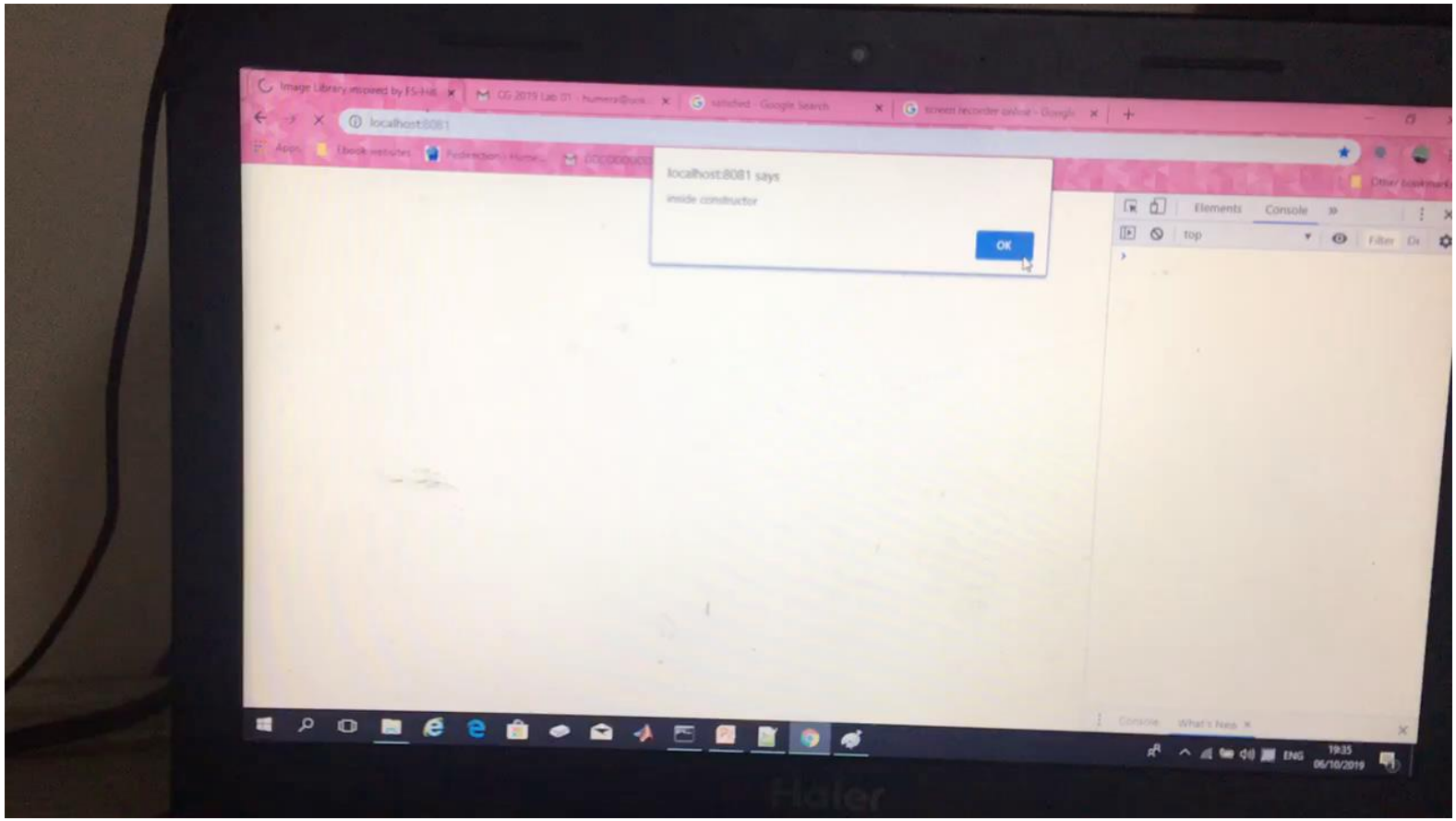
1

```
A = canvas.width/cultus.x;
C= -cultus.img.width;
B = canvas.height/cultus.y;
D = (canvas.height-cultus.img.height) - B*cultus.y;
```

2

```
if(cultus.x > canvas.width - cultus.img.width){
    //console.log(cultus.x);
    //console.log(cultus.y); // just checking access to canvas
   context.translate(-1,0); // stop car at boundary

   cultus.x = A*cultus.x+C; console.log (cultus.y)
  }
if(cultus.y > canvas.height - cultus.img.height){
    //console.log(cultus.x);
    //console.log(cultus.y); // just checking access to canvas
   context.translate(0,-1); // stop car at boundary

   cultus.y = B*cultus.y+D;  console.log(cultus.y);
}
```

Output



Your Task

- Extend given lab to apply check on all four boundaries or walls of the canvas i.e.

- Collision against top wall
- Collision against left wall

Next Lab 07-Lab08

1- Chap 10 Raster Tool for Images Image averaging, lerp, blend, bitwise, Transformation on images

2- Bresenhams's Algorithm

3- First WebGL Canvas (vertices and polygons)