

More Javascript

HTML Canvas

file:testcanvas.html

- An area you can draw in using standard X,Y coordinates
- Default is to have a pixel independent coordinate system that is 300 wide x 150 high
- “Both the canvas itself and the canvas element have a width and height, they're separate things”
- To draw:
 - get “context”
 - beginPath
 - put stuff onto context
 - closePath
 - fill: define fill and fill
 - lines: define stroke and stroke

```
<html><head>
  <style>
    .maind {
      border: 3px solid #734400;
      margin-top: 5%;
      margin-left: 5%;
      width: 80%;
      height: 80%;
    }
  </style></head><body>
  <script src="../JQ/jquery-1.9.1.min.js"></script>
  <canvas id="canvas" class="maind"></canvas>
  <script>
    $(document).ready(
      function() {
        cnvas = document.getElementById("canvas");
        cnvas.width=cnvas.offsetWidth;
        cnvas.height=cnvas.offsetHeight;
        let ctx = cnvas.getContext("2d");
        ctx.beginPath();
        ctx.arc(100, 100, 100, 0, 2 * Math.PI);
        ctx.closePath();
        ctx.fillStyle = "rgba(200,200,0,0.6)";
        ctx.fill();
        ctx.strokeStyle = "#11111155";
        ctx.lineWidth=5;
        ctx.stroke();
      }
    );
  </script></body></html>
```

Animating the Canvas

- Draw, erase, draw, erase ...
- Minor bug... last circle leaves a bit of itself

file: balls.html

```
var canvas = document.getElementById("canvas");
function makeBall() {
    ballcount = ballcount + 1;
    canvas = document.getElementById("canvas");
    tx = canvas.clientWidth;
    ty = canvas.clientHeight;
    ball = new Object();
    ball.radius = Math.random() * (tx * 0.1) + 14;
    ball.x = Math.random() * (tx - 2 * ball.radius) + ball.radius;
    ball.y = Math.random() * (ty - 2 * ball.radius) + ball.radius;
    ball.color = randomColor();
    ball.speed = Math.random() * ball.radius * 0.33 + 1;
    ball.counter = ballcount;
    return ball;
    //drawBall(ball);
}
function startBall() {
    balls.push(makeBall());
    if (animating == 0) {
        console.log("Interval start");
        animating = 1;
        window.requestAnimationFrame(drawBalls);
    }
}
```

```
function drawBall(ctx, ball) {
    ctx.beginPath();
    ctx.arc(ball.x, ball.y, ball.radius, 0, 2 * Math.PI);
    ctx.closePath();
    ctx.fillStyle = ball.color;
    ctx.fill();
    ctx.stroke();
    ball.x = ball.x + ball.speed;
}
function drawBalls() {
    var tx = canvas.clientWidth;
    var ty = canvas.clientHeight;
    var ctx = canvas.getContext("2d");
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    for (i = balls.length - 1; i >= 0; i--) {
        drawBall(ctx, balls[i]);
        // if the ball moves off the screen, remove it from the list
        if ((balls[i].x - balls[i].radius) > canvas.width) {
            balls.splice(i, 1);
        }
    }
    if (balls.length > 0)
        window.requestAnimationFrame(drawBalls);
    else
        animating = 0;
}
```

Animating the canvas 2

file: canvasclick.html

- Really ugly code
 - Limited to circles
 - Not easily extensible
 - Lost the onclick function from divs
- Use JS objects!
- XX.prototype adds method to class
 - this.XX get value of XX for the instance of class
- Within method
 - return value could be more useful
- Similar objects for Triangle and N-Gon
- Problem: code assumes all objects have methods within, draw and offcanvas
 - Solution: prototype inheritance

```
function Ball() {
  let tx = canvas.width;
  let ty = canvas.height;
  this.radius=Math.random() *(tx*0.1) + 14;
  this.x=Math.random() * (tx - 2*this.radius) + this.radius;
  this.y=Math.random() * (ty - 2*this.radius) + this.radius;
  this.color=randomColor();
  this.speed = Math.random()*this.radius*0.1+0.0001,
  this.counter = ballcount;
  ballcount = ballcount+1;
}
Ball.prototype.within = function(xx, yy) {
  let cx = this.x;
  let cy = this.y;
  let dxy = Math.sqrt((cx-xx)*(cx-xx) + (cy-yy)*(cy-yy));
  console.log(cx + " " + cy + " " + this.radius + " " + dxy);
  if (dxy<this.radius)
    return this.counter;
  else
    return -1;
};
Ball.prototype.draw = function(ctx) {
  ctx.beginPath();
  ctx.arc(this.x, this.y, this.radius, 0, 2 * Math.PI);
  //console.log(this.counter + " " + this.x + " " + this.y + " " + this.radius);
  ctx.closePath();
  ctx.fillStyle = this.color;
  ctx.fill();
  ctx.stroke();
  this.x = this.x + this.speed;
}
Ball.prototype.offCanvas = function(mx, my) {
  return mx < (this.x-this.radius);
}
```

Similar code for
Triangle and Ngon

Prototype Inheritance

- Create a class “Shape” that has at least stub versions of all methods
- Make Triangle, Ball and Ngon inherit from Shape
- Make each call the Shape “constructor”

```
function Shape() {  
    this.counter = ballcount;  
    ballcount = ballcount+1;  
    this.p1x=0;  
}  
Shape.prototype.within = function(xx,yy) {  
    return -1;  
}  
Shape.prototype.draw = function() {  
}  
Shape.prototype.offCanvas = function(mx, my) {  
    return this.p1x>mx;  
}
```

```
Triangle.prototype = new Shape();  
function Triangle() {  
    Shape.call();  
    let tx = $(canvas).innerWidth();  
    let ty = $(canvas).innerHeight();  
}
```

.....

Still Animating

file:canvasclick.html

- Left N-Gon contains as an exercise
- Now, add a “listener” for mousedown events within the canvas element
- Then get the location of the click and ask each object if it occurred within it.

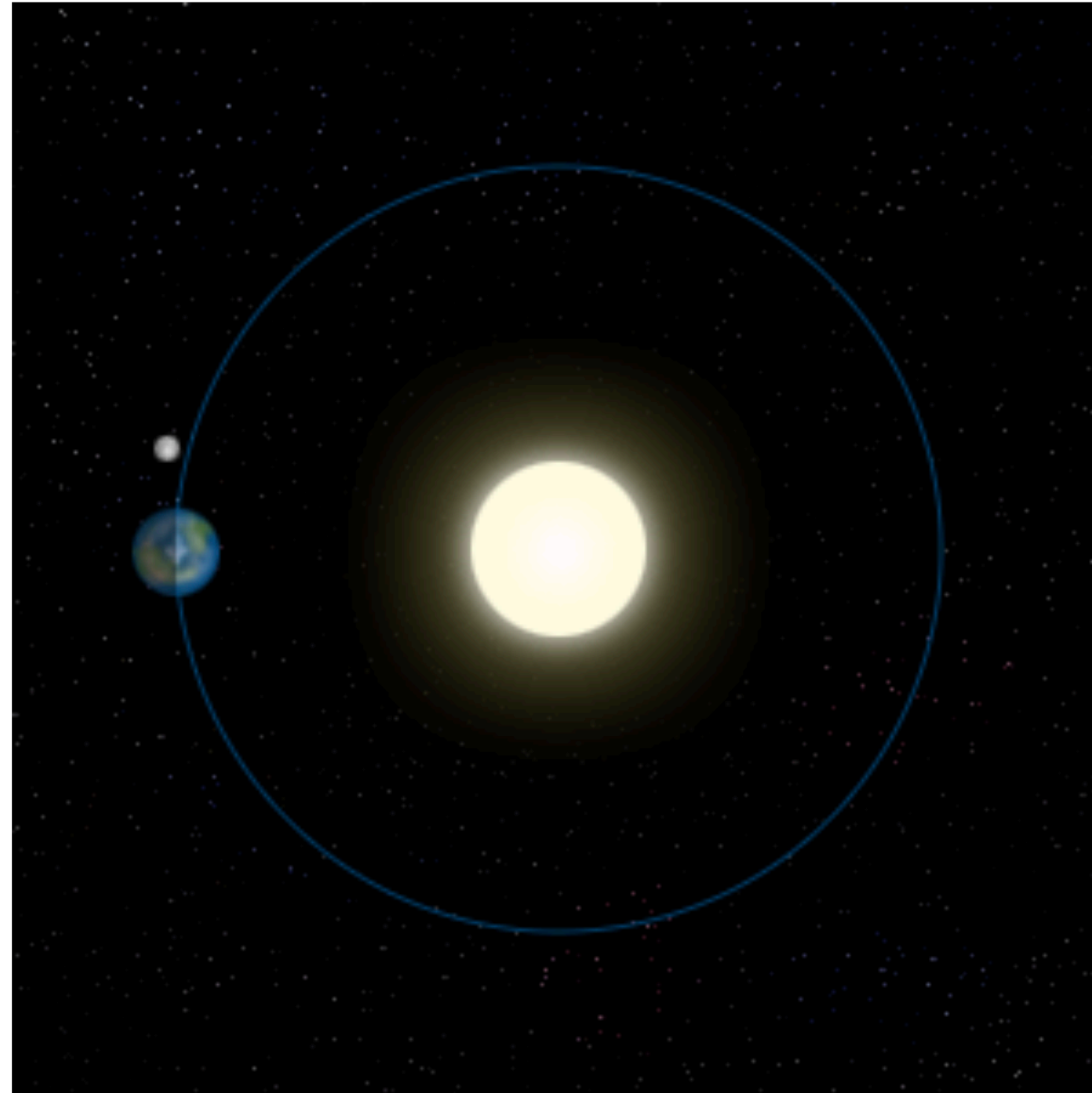
```
$(canvas).on("mousedown", function(e)
{
    getMousePosition(canvas, e);
});
);
function getMousePosition(canvas, event) {
    let rect = canvas.getBoundingClientRect();
    let x = event.clientX - rect.left;
    let y = event.clientY - rect.top;
    //console.log("Coordinate x: " + x, "Coordinate y: " + y);
    for (ball of shapes) {
        let bb = ball.within(x,y);
        //console.log(bb);
        if (bb>=0) {
            console.log("In Shape " + bb);
        }
    }
}
```

```
function startShape(shapeId) {
    if (shapeId==0)
        shapes.push(new Ball());
    if (shapeId==7) {
        let s = 2 + Math.ceil(Math.random()*10);
        shapes.push(new Ngon(s));
    }
    if (shapeId==3)
        shapes.push(new Triangle());
    if (animating==0) {
        animating=1;
        window.requestAnimationFrame(drawShapes);
    }
}

function drawShapes() {
    let tx = canvas.width;
    let ty = canvas.height;
    let ctx = canvas.getContext("2d");
    ctx.clearRect(0, 0, tx, ty);
    for (shape of shapes) {
        shape.draw(ctx);
    }
    for (i=shapes.length-1; i>=0; i--) {
        if (shapes[i].offCanvas(tx,ty)) {
            console.log("Off screen " + i);
            shapes.splice(i,1);
        }
    }
    if (shapes.length>0)
        window.requestAnimationFrame(drawShapes);
    else
        animating=0;
}
```

Last Animating

- A LOT of JS animation not discussed
- A digital Orrery
- `file: ssystem.html`



JS / PHP / MySql Systems

Hit Counting

- Goal:
 - allow a page to show the number of times it has been visited
- What data do I need to do this?
- Tool:
 - PHP Sessions?
 - SQL?
 - If SQL table design?

Hit Counters

- file:hitcounto.php



```
<title>Visit counter</title>
</head><body><div class="bigger"> <div class="c2">
<?php print updateAndGetCount(100); ?>
</div></div><?php function updateAndGetCount($iidd) {
    $servername = "localhost";
    $username = "gtstudent";
    $password = "";
    $dbname = "count";
    $conn = new mysqli($servername, $username, $password, $dbname);
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    $sql = "SELECT max(count) as c FROM count WHERE id = $iidd";
    $result = $conn->query($sql);
    $row = $result->fetch_assoc();
    if ($row==null || $row["c"]==null) {
        $visits = 1;
    }
    else {
        $visits = $row["c"];
        $visits++;
    }
    $sql = "INSERT INTO count (id, count) VALUES ( $iidd , $visits);";
    $result = $conn->query($sql);
    $conn->close();
    return $visits;
} ?>
</body> </html>
```

Improvement to Hit counting

- Should not be the whole page
- Should not just insert into table as that will accumulate a lot of junk
 - file: hitcount.php
 - at least addresses junk problem
 - might be better to use SQL update
 - file: hitcount2.php

Still better HitCount

file:pagewithhitcount.html

- Rather than making the whole page a hit count, create a small bit to do the hitcount
- Use javascript fetch to get data from PHP
- The html and the hitcount do not have to come from the same place!
- PHP does not generate HTML, just returns the number of hits

```
<script>
  const baseURL = "http://comet.cs.brynmawr.edu/~gtowell/380/Lec10/";
  function getHit() {
    const data = { id: 12 };
    let fd = new FormData();
    for(var i in data){
      fd.append(i,data[i]);
    }
    fetch(baseURL+"hitcountwidget.php", {
      method: 'POST',
      mode:"cors",
      body: fd,
    }).then(function(response) {
      response.text().then(function(text) {
        console.log($("#div.hcc").text() + text);
        $("#div.hcc").text("HC:"+text);
      });
    });
  }

  $(document).ready(function() {
    getHit();
  })
</script>
<div id="12" class="hcc"></div>
<div style="height:calc(100% - 50px); margin-top:0px"> ...
```

Javascript Fetch

- Fetch is a Promise
- First “then” occurs on receiving headers
- In this case body might contain JSON or plain text
 - So examine headers to determine what the body will contain.
 - Invoke a new Promise to get the body of the response and parse appropriately
 - THEN handle the parsed result.

```
fetch(myRequest).then(function(response) {  
  const contentType = response.headers.get("content-type");  
  if (contentType && contentType.indexOf("application/json")  
    return response.json().then(function(json) {  
      // process your JSON data further  
    });  
  } else {  
    return response.text().then(function(text) {  
      // this is text, do something with it  
    });  
  }  
});
```

More Promising

Two functions in then
depending on call to resolve
or reject in promise

file:eventloop3.html

```
<html>
  <head>
    <script src="../../JQ/jquery-1.9.1.min.js"></script>
  </head>
  <body>
    <div id="countout"></div>
    <button onclick="push()" id="mybutton">Push Me</button>
    <script>
      var pushCount=0;
      $(document).ready(function() {
        $("#countout").html("Count " + pushCount);
      });
      function push() {
        gtsleep2(1000).then(function(val) {
          pushCount++;
          $("#countout").html(val + " " + pushCount);
        },
        function(reason) {
          $("#countout").html("Rejected " + reason + " " + pushCount);
        });
      }
      function gtsleep2(ms)
      {
        return(new Promise(function(resolve, reject) {
          setTimeout(function() { resolve("success"); }, ms);
        }));
      }
    </script></body></html>
```

Best yet HitCount

file: pagewithhitcount2.html

- Put all of the javascript and supporting CSS into hitcountscript.js and hitcountstyle.css
- Then user only needs to add an element with an attribute hitcountid (along with <link and <script)
 - With a little work could put all css into js file
 - With a little more work, no JQuery

```
<html>
  <head>
    <script src="../JQ/jquery-1.9.1.min.js"></script>
    <link rel="stylesheet" href="hitcountstyle.css">

  </head>
  <body>
    <script src="hitcountscript.js"></script>
    <div hitcountid="12" class="hcc"></div>
    rest of page
```