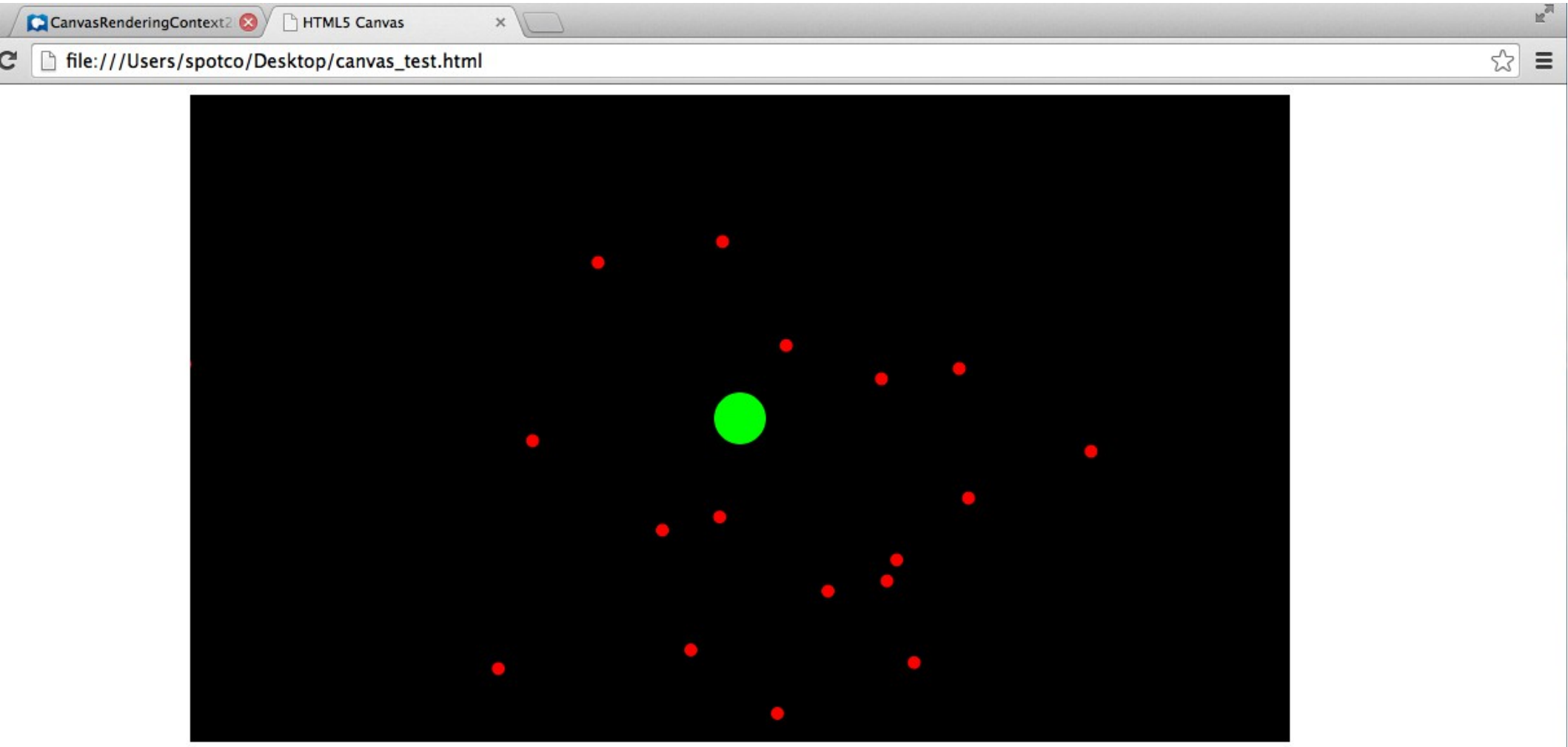
UW GAME DEV CLUB

HTML5 Canvas – Let's make a physics simulator!



UW GAME DEV CLUB

We'll be doing everything in “HTML5 Canvas” now.

(It's very similar to the GamePanel setup, and all the same ideas still apply).

Pros:
It's much easier to share your games (just open a website on your browser!)
HTML5 Canvas's Context2D is slightly more powerful than java's Graphics
Javascript may be faster to get stuff done with (takes less writing)
Chrome dev tools are really good

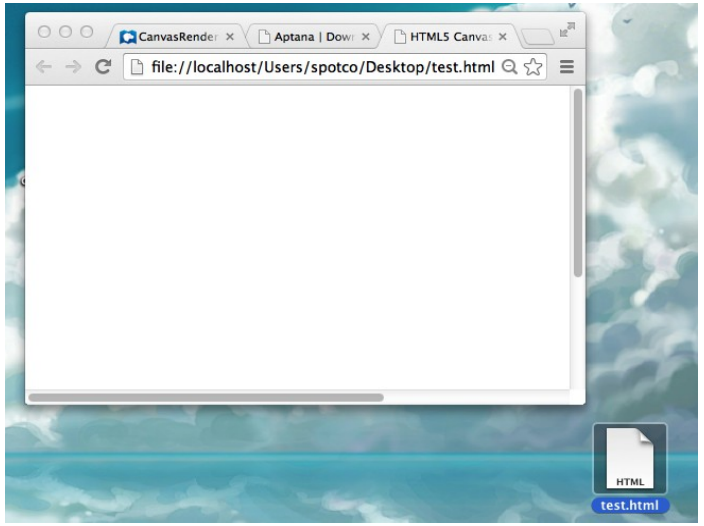
Not java

Cons:
New language, javascript, which may be harder to do stuff in
Most errors will come at runtime
Javascript kinda sucks in its own ways
Performance is kinda iffy

UW GAME DEV CLUB

Basic setup

```
<!DOCTYPE html><html>
<head>
    <title>HTML5 Canvas</title>
    <script>
//code goes here
    </script>
</head>
<body><canvas id="game"  width="850px" height="500px"
style="margin:auto;display:block;"></canvas></body>
</html>
```



It won't look like too much

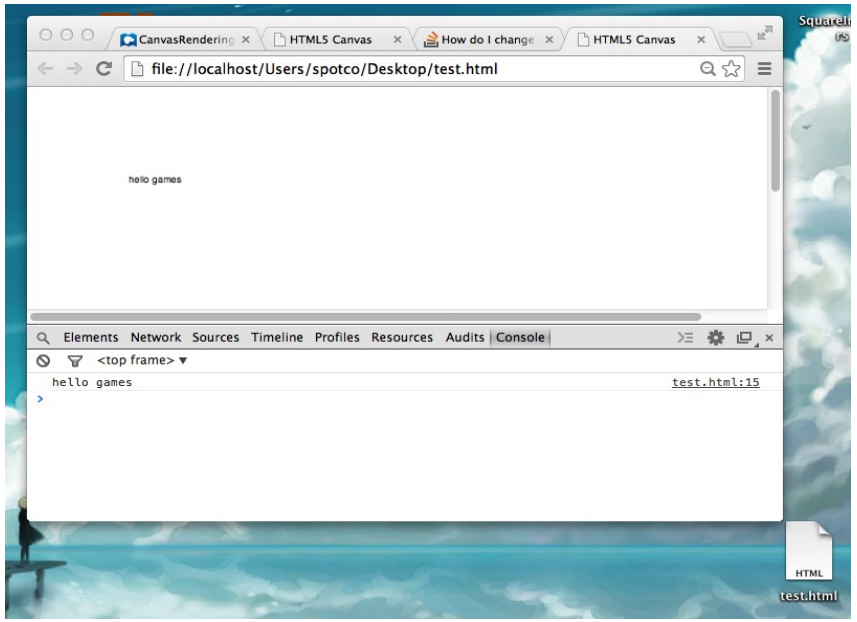
UW GAME DEV CLUB

“Hello Game”

```
<!DOCTYPE html><html>
<head>
    <title>HTML5 Canvas</title>
    <script>
var _g;
var _canvas;

window.onload = function() {
     _canvas = document.getElementById("game");
     _g = _canvas.getContext("2d");
     _g.fillText("hello games",100,100);

     console.log("hello games");
}
    </script>
</head>
<body><canvas id="game"  width="850px" height="500px"
style="margin:auto;display:block;"></canvas></body>
</html>
```



Prints a message to the console (in chrome dev tools)
Draws a message to the screen

UW GAME DEV CLUB

Javascript in 5 slides

```
var x = 5;
var y = "games"

console.log(y);

for (var i = 0; i < 10; i++) {
      console.log(i+" video games");
}

function games(v) {
      console.log("where da "+ v +" games at");
}

games("vidheo");
```

Declare all variables with “var”

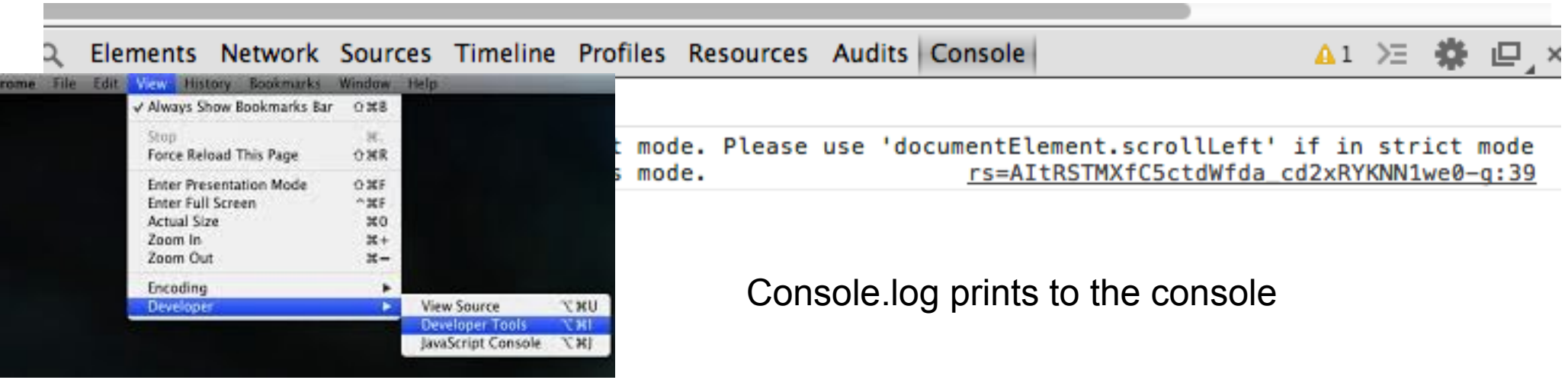
No types (that you need to declare)

“console.log” instead of println

For/while/if look the same as java

Functions look pretty similar to java

Functions call similar to java



Console.log prints to the console

UW GAME DEV CLUB

Javascript in 5 slides

```
window.onload = function() {

	var array = [1,2,3];
	array.push(4);

	for (var i = 0; i < array.length; i++) {
		console.log(array[i]);
	}

	array.splice(0,1);

	console.log(array); //this will print [2,3,4]

}
```

Do all you initialization code in this “window.onload” function

Make arrays like this

Add to arrays with push

Looping through them is still the same as java

Remove elements from arrays with array.splice(INDEX,HOW_MANY)

UW GAME DEV CLUB

Javascript in 5 slides

```
window.onload = function() {

	var obj = {};
	obj["games"] = 0;

	var obj2 = {};
	obj2.games = 0;

	var obj3 = {
		games: 0
	};

	//obj, obj2 and obj3 are all the same

	var obj4 = {
		games: 0,
		print_games:function(){
			console.log(this.games)
		}
	};
	obj4.print_games();

}
```

You make hashmap-dictionary-objects like this

You'll use this instead of classes in javascript

You can do object-oriented programming with these

UW GAME DEV CLUB

Javascript in 5 slides

```
window.onload = function() {
      setInterval(update,20); //update every 20ms
}

function update() {
      console.log("update");
}
```

Here's the update cycle.

Note, in javascript, you can pass around functions like parameters.

Ex:

```
function update() { console.log("update");}
function caller(f) {
      f();
}
caller(update);
```

UW GAME DEV CLUB

Javascript in 5 slides

Here's keyboard/mouse input:

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

window.onload = function() {
      _canvas.addEventListener('mousedown',mouse_down);
      _canvas.addEventListener('mouseup',mouse_up);
      _canvas.addEventListener('mousemove',mouse_move);
      window.addEventListener('keydown', key_down);
      window.addEventListener('keyup',function(e) {
            console.log(e.keyCode);
      });
}

function mouse_down(e){
      var mouse_pos = get_canvas_cursor_position(_canvas,e);
      console.log(mouse_pos.x,mouse_pos.y)
}
function mouse_up(e) {var mouse_pos = get_canvas_cursor_position(_canvas,e);}
function mouse_move(e) { var mouse_pos = get_canvas_cursor_position(_canvas,e);}
function key_down(e) { console.log(e.keyCode) }
```

(You can either do it separate function or inline style)

UW GAME DEV CLUB

```
var _canvas;
var _g;

window.onload = function() {
      _canvas = document.getElementById("game");
      _g = _canvas.getContext("2d");
      _g.fillText("hello games",100,100);
}
```

The "_g" object is a "CanvasRenderingContext2D" object

```
_g.clearRect(0,0,WID,HEI);

_g.fillStyle = "rgb(255,0,0)";
_g.fillRect(X,Y,WID,HEI);
```

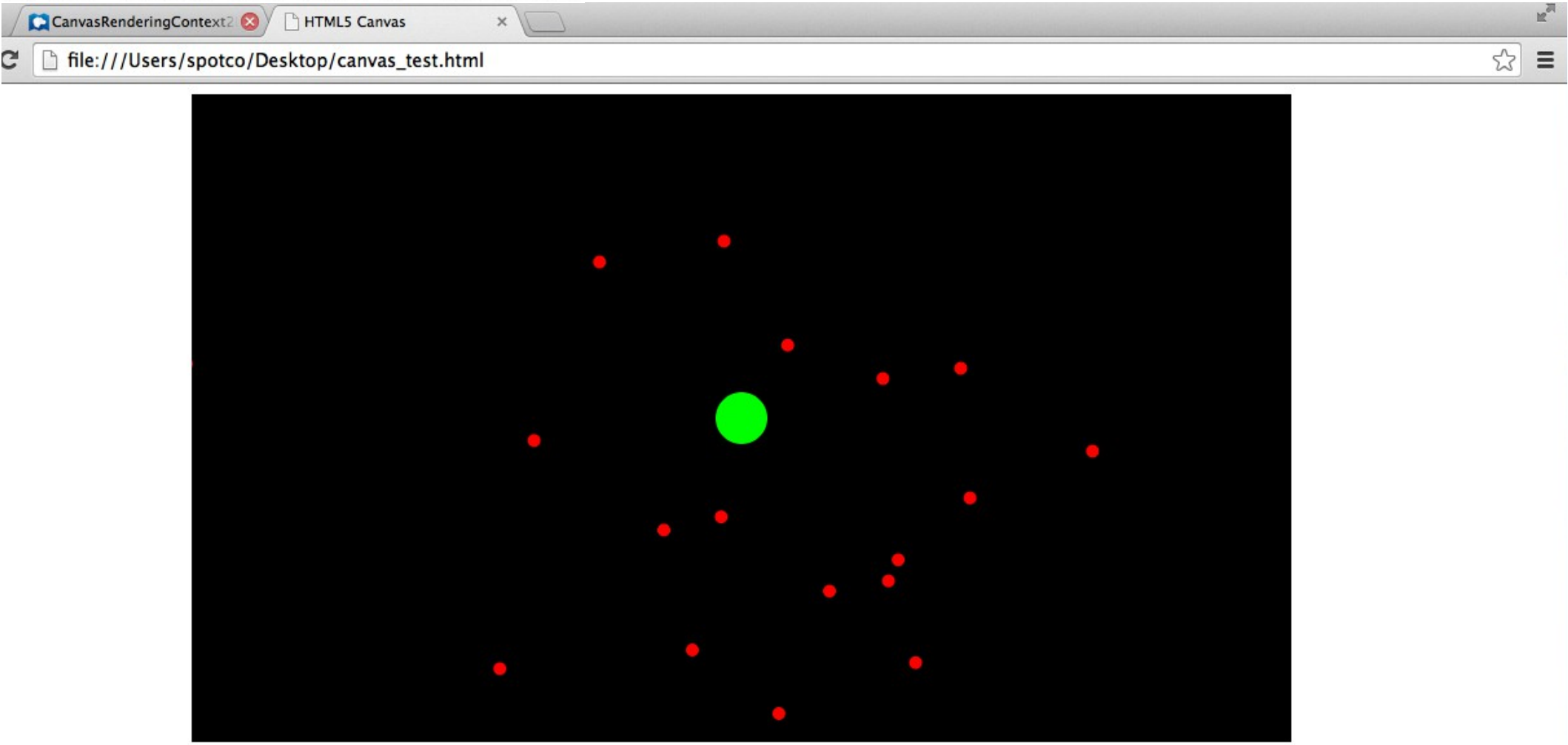
Here's how to
-clear the screen
-draw a filled rectangle
-filled circle with the _g object

Note the fillStyle needs to be a CSS string

```
_g.fillStyle = "rgb(0,255,0)";
_g.beginPath();
_g.arc(X,Y,RADIUS,STARTING_ANGLE,ENDING ANGLE);
//_g.arc(X,Y,RADIUS,0,Math.PI*2);
_g.closePath();
_g.fill();
```

https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D

UW GAME DEV CLUB



Let's make this “game”:
Big green planet in the center
Small red planets “orbit” around it

UW GAME DEV CLUB

```
var _gameobjs = [];

function make_object(x,y) {
      var rtv = {};
      rtv.x = x;
      rtv.y = y;
      rtv.vx = Math.random()*5-2.5;
      rtv.vy = Math.random()*20-10;
      return rtv;
}

//...
      _gameobjs.push(make_object(100,100));
```

Insights:

Make an array to store your “planet objects”

Make a method that makes a “planet” object, with x,y,vx and vy fields.

```
 //...

 for(var i_obj = 0; i_obj < _gameobjs.length; i_obj++) {
       var itr_obj = _gameobjs[i_obj];

       itr_obj.x += itr_obj.vx;
       itr_obj.y += itr_obj.vy;

       _g.fillStyle = COLOR.RED;
       _g.beginPath();
       _g.arc(itr_obj.x,itr_obj.y,5,0,Math.PI*2);
       _g.closePath();
       _g.fill();
 }
```

Loop through your game objects in your update cycle

Draw every object, as well as applying its vx and vy

UW GAME DEV CLUB

Orbital Gravity

```
var intensity = 1/distance_to_gravity_center(itr_obj.x,itr_obj.y) * 100;

var delta = {x:get_gravity_center().x-itr_obj.x, y:get_gravity_center().y-itr_obj.y};

var delta_len = Math.sqrt(Math.pow(delta.x,2)+Math.pow(delta.y,2));

var normalized_delta = {x:delta.x/delta_len,y:delta.y/delta_len};

itr_obj.vx += normalized_delta.x*intensity;
itr_obj.vy += normalized_delta.y*intensity;
```

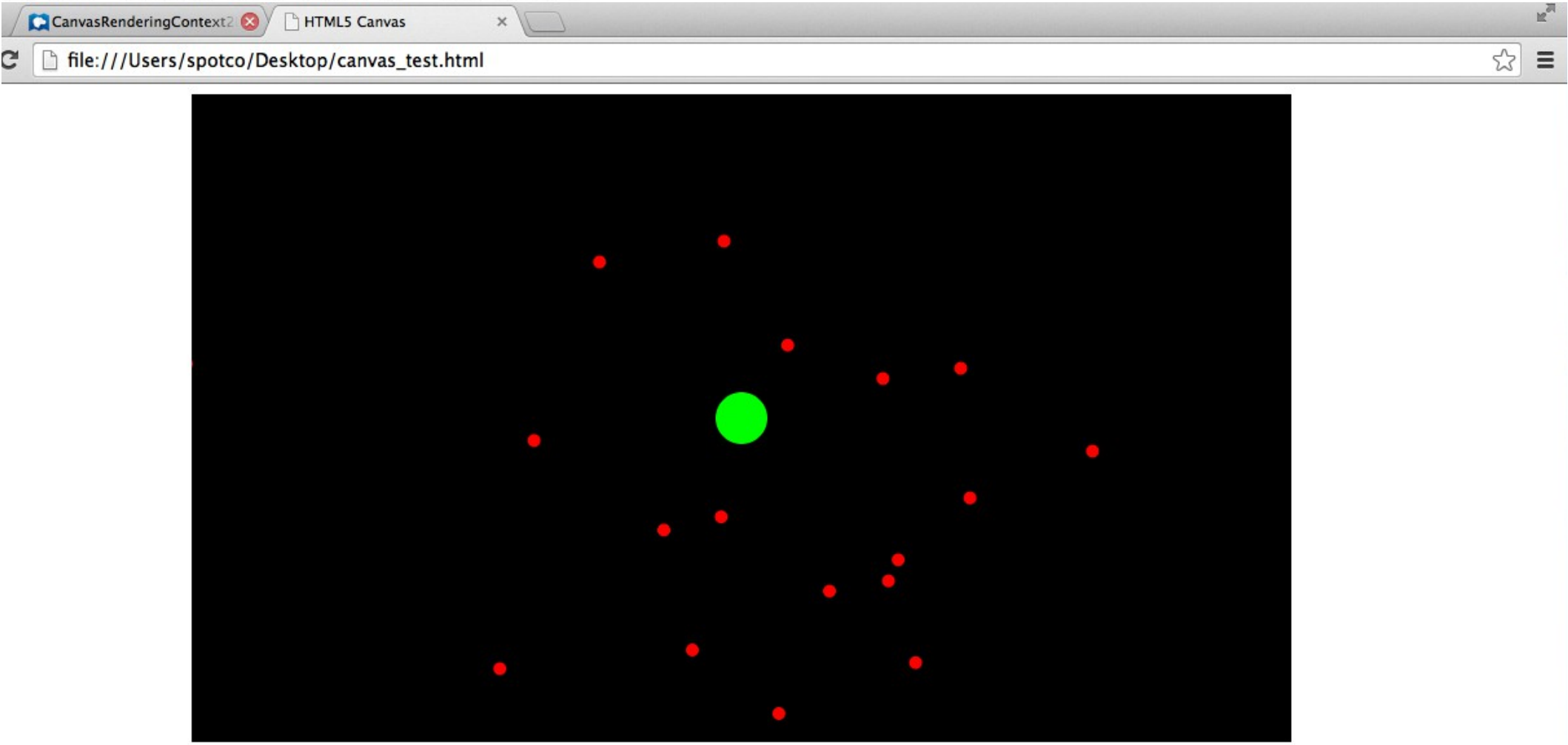
Calculate the distance from the object to your gravity center.
Force is inversely proportional to distance (100 is just a number that looked fine)

Calculate the object's difference (delta) x and y to the gravity center.

Normalize this as a 2d vector (divide both x and y by the vector length)

Apply changes to object's vx and vy in the direction of the difference,
scaled by intensity

UW GAME DEV CLUB



Activity: How would you make this into a game?

https://gist.github.com/spotco/8586743