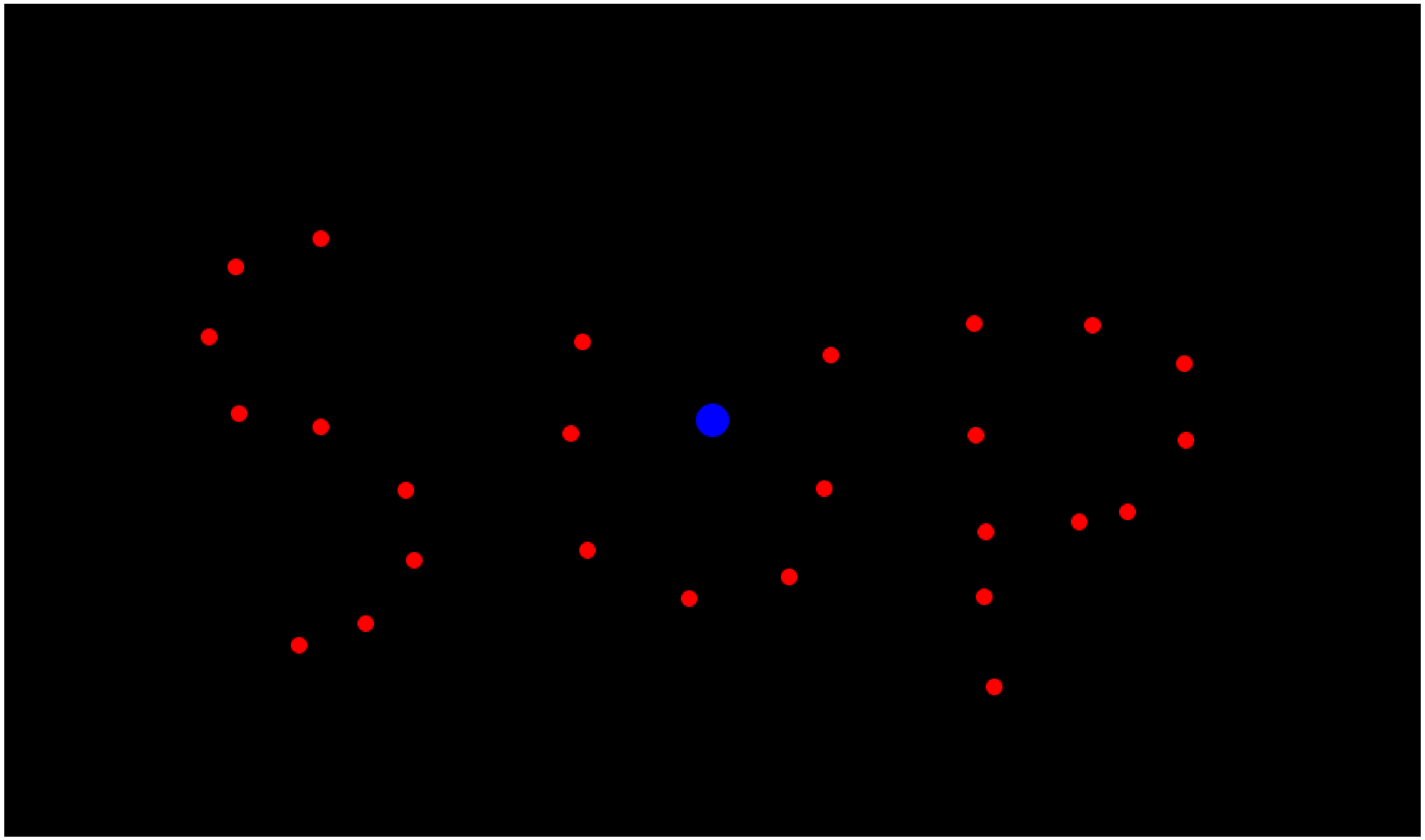
UW GAME DEV CLUB

Javascript canvas miscellany:
- Organizing javascript logic
- Using push/pop matrix (and making a player-following camera)
- Loading levels



UW GAME DEV CLUB

A common separation pattern:

```
<script src="util.js"></script>
<script src="draw.js"></script>
<script src="levels.js"></script>
<script src="game.js"></script>
```

Util.js – Anything pertaining to keyboard/mouse input and other commonly used functions (maybe a vector library?)

Draw.js – Drawing library code (see next slides)

Levels.js – Any json levels for the game (see slides at end)

Game.js – All the “game logic”, which will use the libraries linked

UW GAME DEV CLUB

Simplifying input

```
var KEYBOARD = {
	LEFT: 37,
	UP: 38,
	RIGHT: 39,
	DOWN: 40
};

var KEYS_DOWN = {};
function key_down(e) {
	KEYS_DOWN[e.keyCode] = true;
}
function key_up(e) {
	KEYS_DOWN[e.keyCode] = false;
}
```

← Hopefully this top part will be in your util/shared js file

```
//...

	window.addEventListener('keydown', key_down);
	window.addEventListener('keyup',key_up);

//...

function update() {
	if (KEYS_DOWN[KEYBOARD.UP]) {
		_player_y -= 5;
	}
}
```

← And this would be in your game logic js file

UW GAME DEV CLUB

Simplifying drawing (I like my methods better)

```
var GLIB = {
	clear:function() {
		_g.clearRect(0,0,WID,HEI);
	},
	draw_circle:function(x,y,rad,color) {
		_g.fillStyle = color;
		_g.beginPath();
		_g.arc(x,y,rad,0,Math.PI*2);
		_g.closePath();
		_g.fill();
	},
	draw_text:function(x,y,text) {
		_g.textAlign = "center";
		_g.fillStyle = COLOR.BLACK;
		_g.fillText(text,x,y);
	},
	draw_rect:function(x,y,wid,hei,color) {
		_g.fillStyle = color;
		_g.fillRect(x,y,wid,hei);
	}
};
```

← This would go in drawing js

```
//...

function update() {
	GLIB.draw_circle(50,50,10,COLOR.RED);
}
```

← And this in your game logic js

UW GAME DEV CLUB

Context global state operations

```
_g.translate(50,50)
```

This will shift everything you draw by 50px “x” and 50px “y”, forever.

How is this useful?

Other useful methods include
```
_g.scale(scale_x,scale_y)
_g.rotate(angle)
```

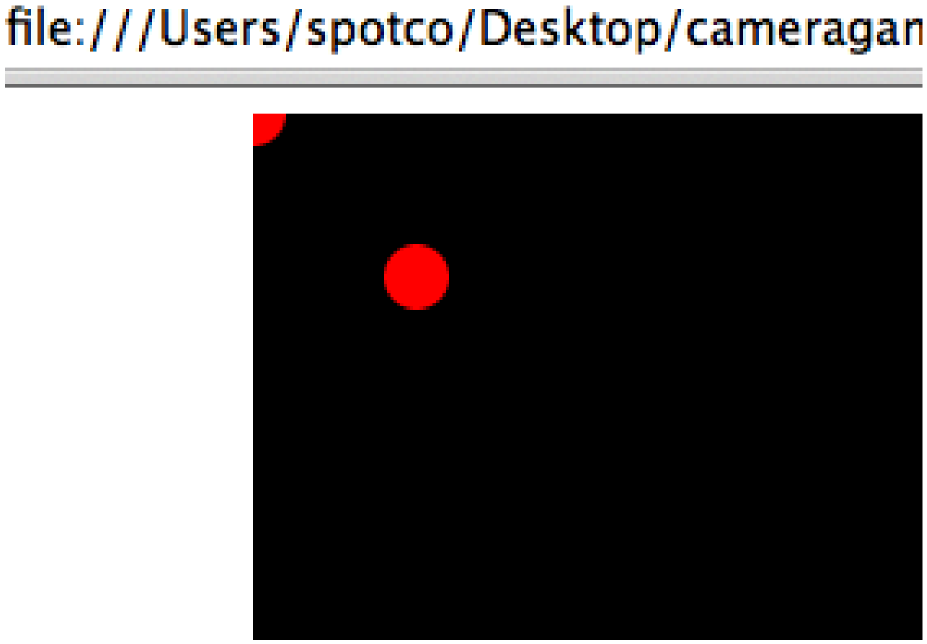
See:

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

UW GAME DEV CLUB

Context save/restore

```
_g.save();
     _g.translate(50,50);
     GLIB.draw_circle(0,0,10,COLOR.RED);
_g.restore();

GLIB.draw_circle(0,0,10,COLOR.RED);
```



This will draw 2 circles, one at (50,50) and one at (0,0).

Save() will save the current "transformation" of the canvas

Restore() will restore the current "transformation" of the canvas to the last save()

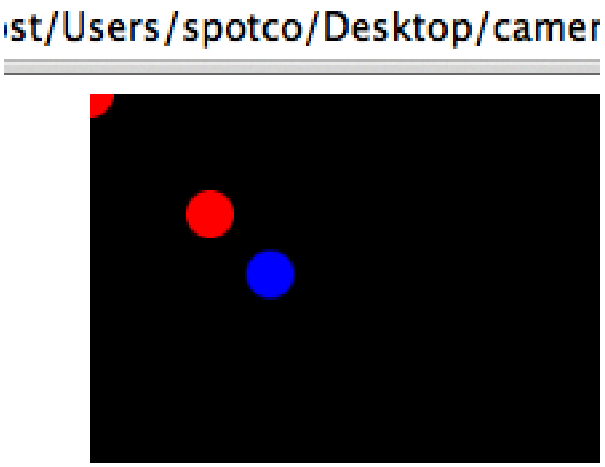
UW GAME DEV CLUB

Context save restore stack

```
_g.save();
     _g.translate(50,50);
     GLIB.draw_circle(0,0,10,COLOR.RED);

     _g.save();
          _g.translate(25,25);
          GLIB.draw_circle(0,0,10,COLOR.BLUE);
     _g.restore();

_g.restore();

GLIB.draw_circle(0,0,10,COLOR.RED);
```

Think of save as pushing a "current transform" onto the stack. (Does not modify the current transform though!)

Think of restore as popping the top "transform" from the stack.

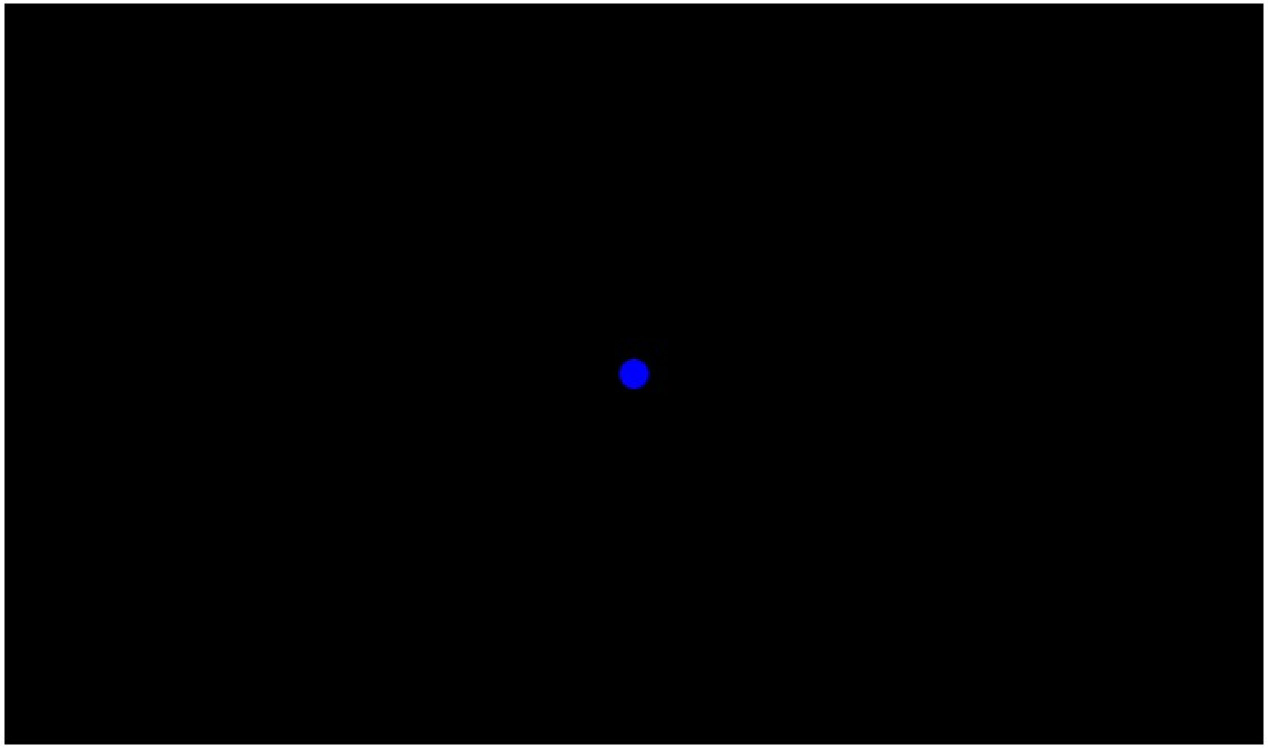


So how is this useful at all?

UW GAME DEV CLUB

Making a camera centered on the player

```
_g.save();
     _g.translate(WID/2-_player_x,HEI/2-_player_y);
     GLIB.draw_circle(_player_x,_player_y,10,COLOR.BLUE);
_g.restore();
```

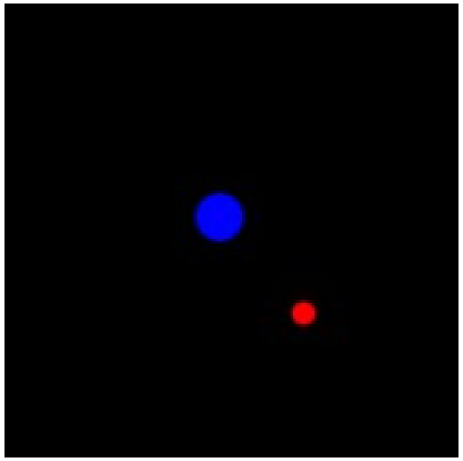


If you have nothing else in your game, it'll appear like the player's not moving at all.

UW GAME DEV CLUB

Drawing other objects

```
_g.save();
    _g.translate(WID/2-_player_x,HEI/2-_player_y);
    GLIB.draw_circle(_player_x,_player_y,10,COLOR.BLUE);

    for (var i_obj = 0; i_obj < _gameobjs.length; i_obj++) {
        var itr_obj = _gameobjs[i_obj];
        GLIB.draw_circle(itr_obj.x,itr_obj.y,itr_obj.radius, itr_obj.color);
    }
_g.restore();
```



Where in the init, I'm doing:

```
_gameobjs.push(create_powerup(50,50))
```

Let's make a more interesting world to walk around.

Level files

```
var LEVEL = {
     objects:[
          {x:0, y:0},
          {x:50, y:50},
          {x:100, y:100},
          {x:150, y:150}
     ]
};
```

← Store your data in “JSON” notation somewhere

And then somewhere in your init, do this:

```
for (var i_obj = 0; i_obj < LEVEL.objects.length; i_obj++) {
     var itr_obj = LEVEL.objects[i_obj];
     _gameobjs.push(create_powerup(itr_obj.x,-itr_obj.y));
}
```

UW GAME DEV CLUB

My shitty visual level editor for a game

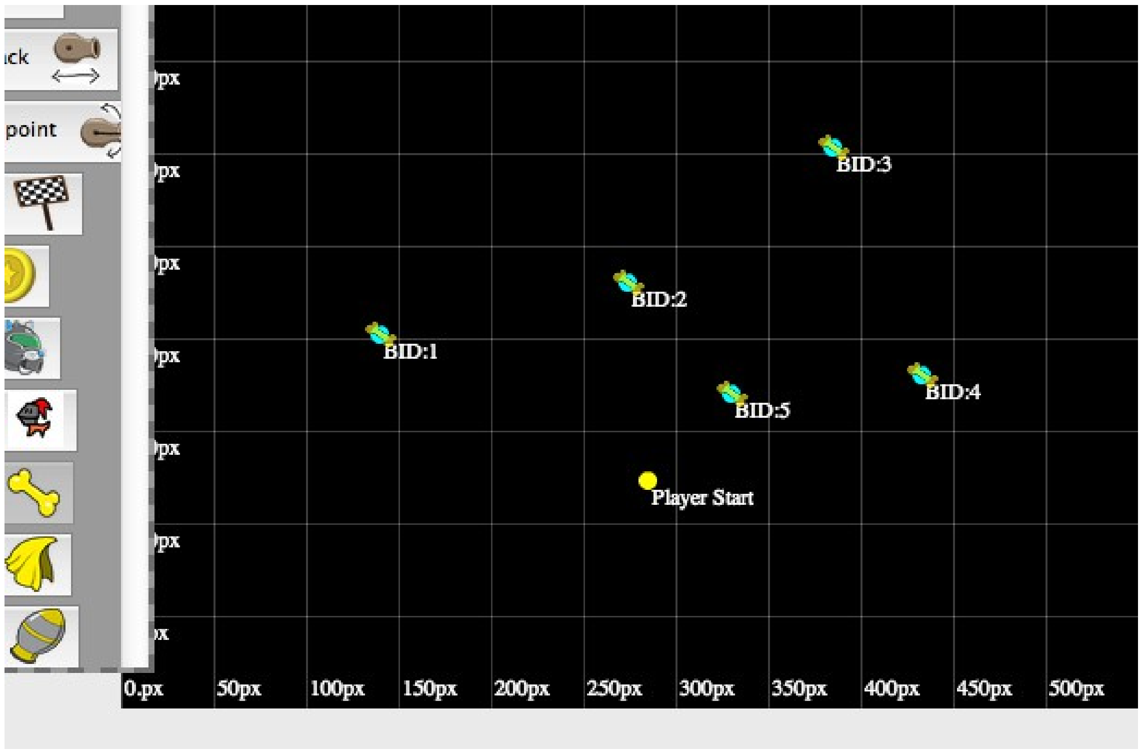
http://spotcos.com/gogodoggy/editor.html

Add a “point” object by holding down “Q” and clicking.

Change player start by holding down “W” and clicking.

Output the JSON by pressing “P” or clicking “Output JSON” (It'll be in the text area below)

It outputs some extraneous information as well.



```
{
    "islands": [],
    "start_x": 285,
    "connect_pts": {},
    "start_y": 123,
    "objects": [{"x": 140, "label": "BID:1", "y": 202, "bid": 1, "type": "dogbone"}],
    "assert_links": 0
}
```

UW GAME DEV CLUB

Ideas for save/restore state practice

Implement a “green ball” that orbits the player and follows the player when it moves. (Doable without using save/restore)

Implement a “zoom” functionality for the game. Press a button to zoom in/out.

Make the camera always rotate behind the player facing.