

Event Listeners

2019-2020

Fernando Birra

Objetivos

- Aprender a criar programas interativos usando *event listeners (callbacks)*
 - Botões
 - Menus
 - Mouse
 - Teclado
 - Reshape (janela)

Exemplo: Botão para controlar direção de rotação

- Começamos por adicionar um botão ao exemplo do quadrado, para controlar a direção de rotação do mesmo
- Na função **render** podemos usar uma variável booleana *direction* que controlará o incremento ou decremento do ângulo

```
var direction = true; // global initialisation
```

```
render() {  
    ...  
    if(direction) theta += 0.1;  
    else theta -= 0.1;  
    ...  
}
```

O botão

- No ficheiro HTML:

id para ser usado no .js

Change Rotation Direction

```
<button id="DirectionButton">Change Rotation Direction</button>
```

tag HTML
button

Texto a ser mostrado no botão

- Fazendo um click no botão gera um evento `click`

Event Listener do botão

- Sem nenhum *event listener* definido, o evento ocorre mas é ignorado...
- Distintas formas para adicionar o tratamento do evento:

```
document.getElementById("DirectionButton").onclick =  
function() {  
    direction = !direction;  
};
```

```
document.getElementById("DirectionButton").addEventListener("cl  
ick", function() {  
    direction = !direction;  
});
```

A evitar! Código no HTML!

```
<button ... click="direction=!direction">...</button>
```

Controlo da velocidade

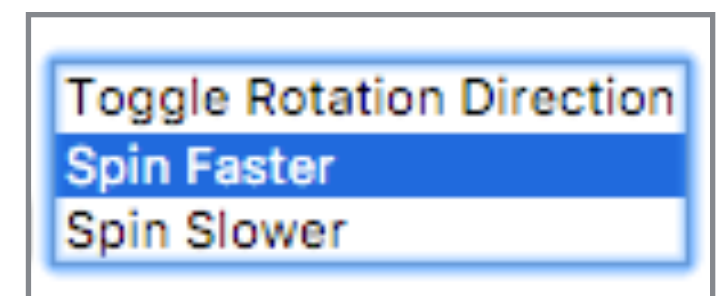
```
var delay = 100;

function render()
{
    setTimeout(function() {
        requestAnimationFrame(render);
        gl.clear(gl.COLOR_BUFFER_BIT);
        theta += (direction ? 0.1 : -0.1);
        gl.uniform1f(thetaLoc, theta);
        gl.drawArrays(gl.TRIANGLE_STRIP, 0, 4);
    }, delay);
}
```

Menu para controlo da velocidade

- O elemento HTML `select` oferece um leque de escolhas (menu)
- Cada entrada no menu é um elemento `option` com um inteiro `value` retornado no evento click

```
<select id="mymenu" size="3">  
  <option value="0">Toggle Rotation Direction</option>  
  <option value="1">Spin Faster</option>  
  <option value="2">Spin Slower</option>  
</select>
```



Event Listener do menu

```
var m = document.getElementById("mymenu");  
m.addEventListener("click", function() {  
    switch (m.selectedIndex) {  
        case 0:  
            direction = !direction;  
            break;  
        case 1:  
            delay /= 2.0;  
            break;  
        case 2:  
            delay *= 2.0;  
            break;  
    }  
});
```

Controlo através do teclado

```
window.addEventListener("keydown", function() {  
    switch (event.keyCode) {  
        case 49: // '1' key  
            direction = !direction;  
            break;  
        case 50: // '2' key  
            delay /= 2.0;  
            break;  
        case 51: // '3' key  
            delay *= 2.0;  
            break;  
    }  
});
```

objeto
window
representa o
conteúdo
todo do
browser

Variante (sem keycodes)

```
window.onkeydown = function(event) {  
    var key = String.fromCharCode(event.keyCode);  
    switch (key) {  
        case '1':  
            direction = !direction;  
            break;  
        case '2':  
            delay /= 2.0;  
            break;  
        case '3':  
            delay *= 2.0;  
            break;  
    }  
};
```

Elemento Slider

- Elemento `input` de tipo `range` permite fornecer à aplicação um valor num intervalo

```
<div>  
  speed 0% <input id="slide" type="range" min="0" max="100"  
  step="10" value = "50"/> 100%  
</div>
```

valor mínimo

valor máximo

deslocamento
necessário para
gerar evento

valor inicial

speed 0%  100%

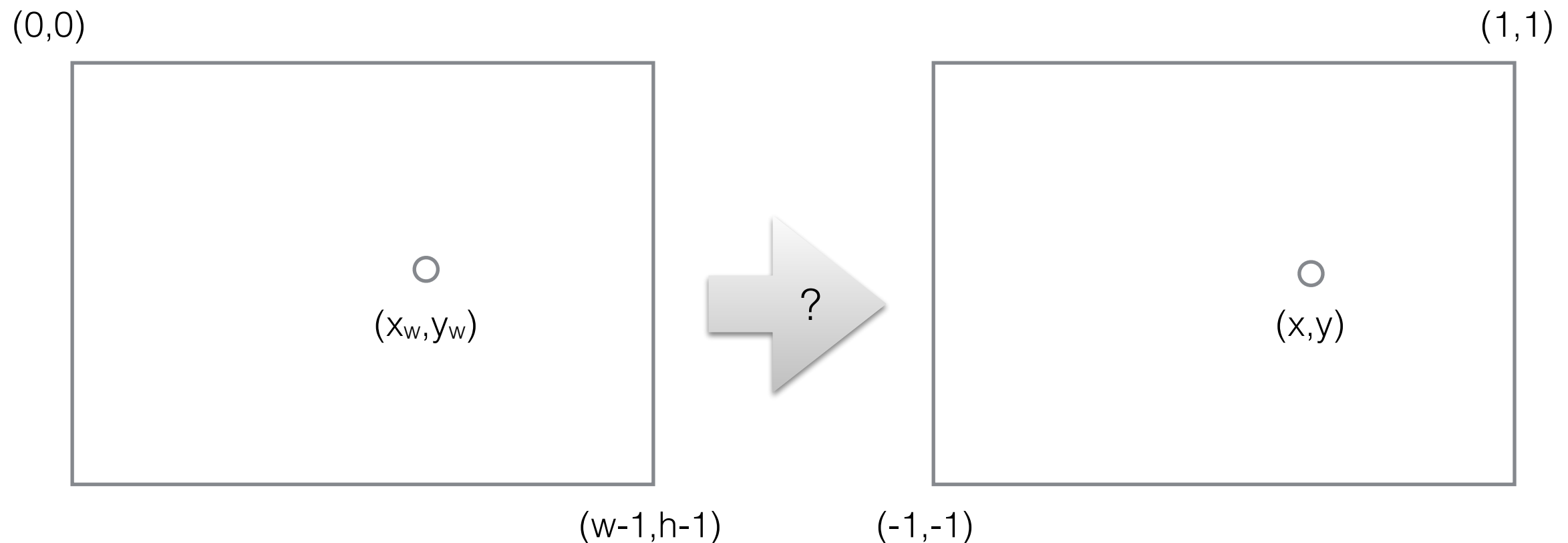
Event Listener do slider

id do slider

```
document.getElementById("slide").onchange =  
    function() {  
        delay = event.srcElement.value;  
    };
```

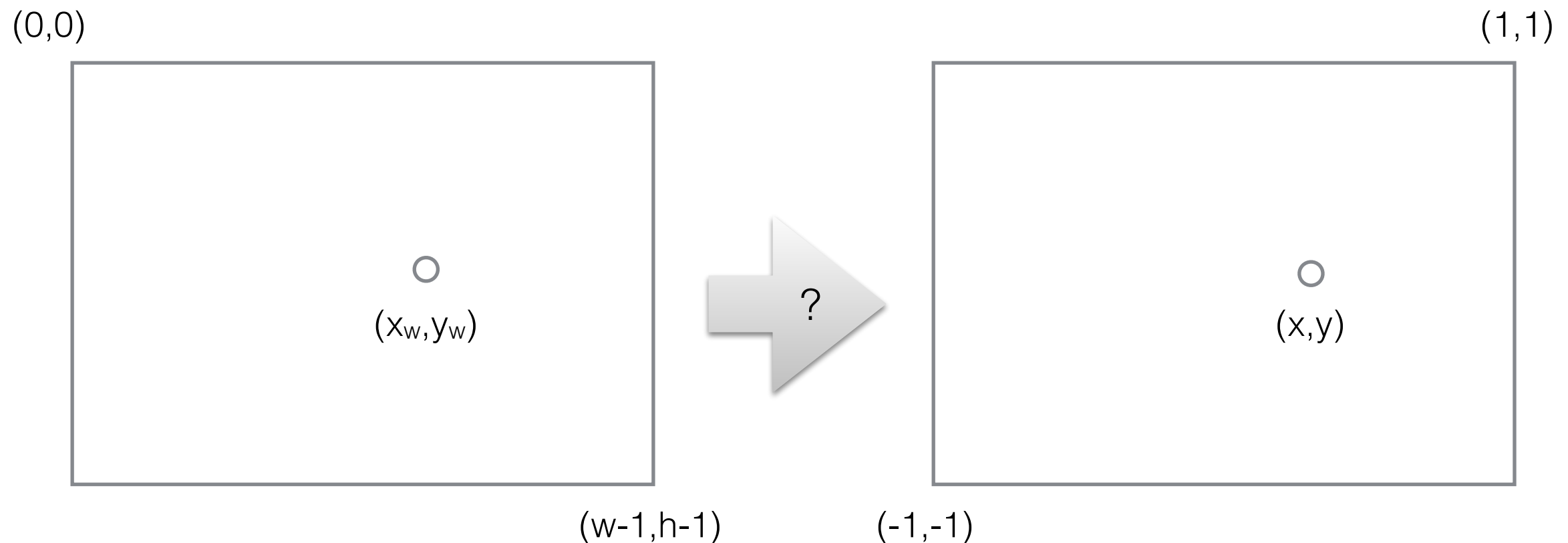
Também se poderia ter usado **oninput** e, nesse caso, os eventos são gerados continuamente.

Aquisição de posições



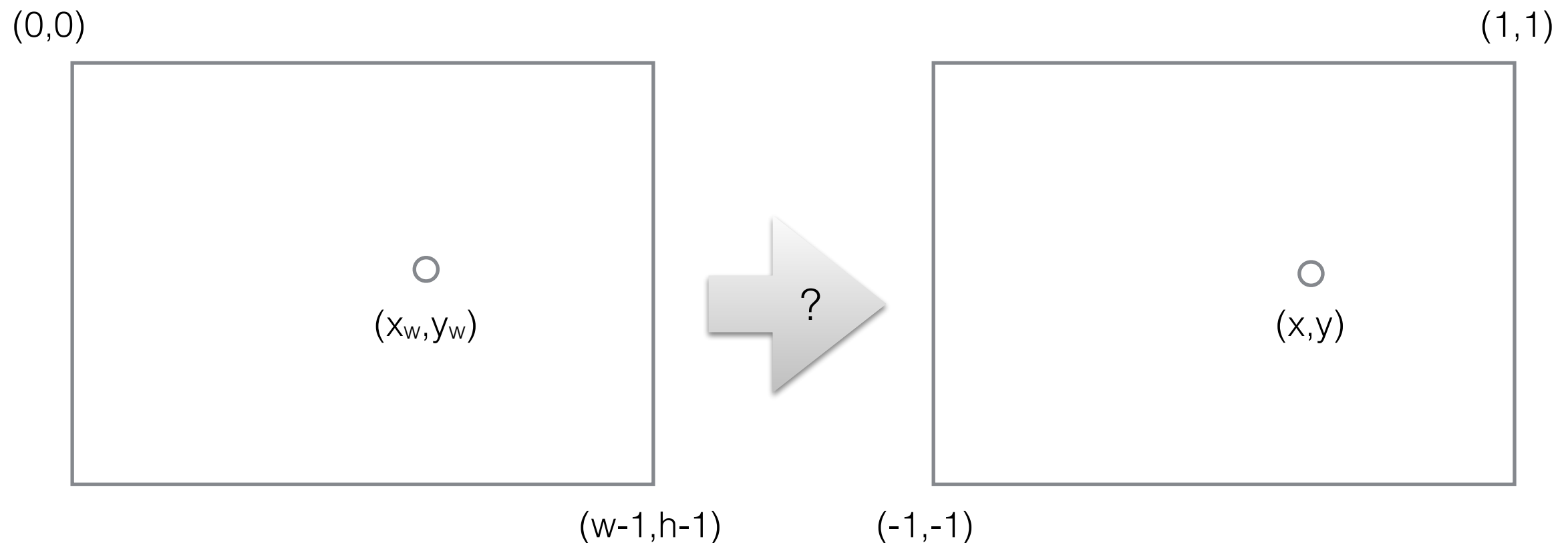
- É necessário converter das coordenadas do ecrã, fornecidas no evento, para coordenadas do modelo!

Aquisição de posições



- Esta transformação pode ser importante para:
 - reponder a eventos de mudança das dimensões da janela
 - desenhar novos elementos no ecrã, na posição fornecida pelo utilizador
 - seleccionar um objeto (pick)

Aquisição de posições



$$(0,0) \rightarrow (-1,1)$$

$$(w,h) \rightarrow (1,-1)$$

$$x = -1 + 2 \cdot x_w / w$$
$$y = -1 + 2 \cdot (h - y_w) / h$$

Conversão de coordenadas

- O canvas, especificado no HTML, tem dimensões `canvas.width` x `canvas.height`
- As coordenadas retornadas referentes ao campo superior esquerdo do canvas são: `event.offsetX` e `event.offsetY`

```
// add a vertex to GPU for each click
```

```
canvas.addEventListener("click", function() {  
    gl.bindBuffer(gl.ARRAY_BUFFER, vBuffer);  
    var t = vec2(-1 + 2*event.offsetX/canvas.width,  
                -1 + 2*(canvas.height-event.offsetY)/canvas.height);  
    gl.bufferSubData(gl.ARRAY_BUFFER, sizeof['vec2']*index, t);  
    index++;  
});
```

Exemplos tipo CAD

www.cs.unm.edu/~angel/WebGL/7E/03

Proposta de trabalho:

- Colocar os exemplos a funcionar corretamente usando `offsetX` e `offsetY`
- Modificar os exemplos para que durante a fase de desenho se possa ir vendo o resultado que seria obtido caso se terminasse a edição

Eventos de janelas

- Os eventos podem ser gerados por ações que afetam a janela do canvas
 - mover ou expor uma janela
 - redimensionar uma janela
 - abrir uma janela
 - minimizar/restaurar uma janela
- Há callbacks por omissão para cada um destes eventos

Evento *onresize*

- Um evento *onresize* ocorre quando se dá a alteração das dimensões duma janela
- Em resposta é necessário:
 - atualizar o conteúdo da mesma
 - Opções:
 - Mostar os mesmos objetos mas com diferente tamanho
 - Mostrar mais ou menos objetos mantendo o tamanho
 - Na maior parte das vezes queremos manter as proporções

Evento *onresize*

- As dimensões da nova janela podem ser acesadas através de `window.innerWidth` e `window.innerHeight`
- Usa-se `window.innerWidth` e `window.innerHeight` para alterar `canvas.width` e `canvas.height`
- Exemplo: Manter um canvas quadrado

Exemplo: Mantendo o canvas quadrado

```
window.onresize = function() {  
    var height = window.innerHeight;  
    var width = window.innerWidth;  
    var s = Math.min(width, height);  
    canvas.width = s;  
    canvas.height = s;  
    gl.viewport(0,0,s,s);  
};
```

Assume-se que o canvas é inicialmente quadrado

a função viewport permite definir a origem e a dimensão da área do canvas que é efetivamente usada para visualização dos gráficos (visor)

Viewport

- Uma aplicação não é obrigada a usar toda a área do canvas para a image: `gl.viewport(x,y,w,h)`
- Os valores são dados em pixels (coordenadas da janela)

