Microprogetto 2 – piano
Vedi file: cgLab01 (.html e .js)

Proositi:

1. Passiamo da una mesh “mono-triangolo” ad una mesh “mono-quad” (diagonal split) e poi un pentagono
2. Settiamo una matrice di modellazione “a mano”
3. Eseguiamo una mini animazione per far ruotare il nostro pentagono

26

Disegnamo un quad

- ✓ Modifichiamo la geometria e connettività per produrre un quad costitutito da due triangoli(con un diagonal split):

```
var punto0 = { x:-0.5, y:-0.5, z:0 };
var punto1 = { x:+0.5, y:-0.5, z:0 };
var punto2 = { x:-0.0, y: 0.4, z:0 };
var punto3 = { x:+0.6, y: 0.8, z:0 };

// una mesh INDICIZZATA
var listaVertici = [
        punto0,
        punto1,
        punto2,
        punto3
];
var listaFacce = [
        0, 1, 2,
        3, 2, 1, // diagonal split della nostra unica faccia quad!
];
```

- ✓ Nota che l’edge condiviso, 1,2, è percorso in sensi opposti, quindi la nostra minuscola mesh è “ben orientata”
- ✓ Testare cosa succede altrimenti! (back-face culling elimina una delle due facce

27

Disegnamo un pentagono (tre triangoli)

```
var punto0 = { x:-0.5, y:-0.5, z:0 };
var punto1 = { x:+0.5, y:-0.5, z:0 };
var punto2 = { x:-0.0, y: 0.4, z:0 };
var punto3 = { x:+0.6, y: 0.8, z:0 };
var punto4 = { x:+0.9, y: 0.0, z:0 };

// una mesh INDICIZZATA
var listaVertici = [
        punto0,
        punto1,
        punto2,
        punto3,
        punto4,
];
var listaFacce = [
        0, 1, 2, // 1ma faccia
        3, 2, 1, // 2da faccia
        4 ,3, 1, // 3za faccia
];
```

- ✓ Nota che l'edge condiviso dalle prime due facce, 1,2, è percorso in sensi opposti, quindi la nostra mesh è "ben orientata"
- ✓ Idem per l'edge 1,3, condiviso dalla 2nda e 3za faccia

28

Esercizio: costruiamo manualmente una metrice di traslazione

La classe `Matrix4` di `three.js` è preposta a rappresentare **matrici di trasformazione**.
Come esercizio, costruiamo manualmente una matrice di traslazione:
(avremmo potuto usare funzioni esistenti di `three.js`)

```
function matriceDiTraslazione( dx, dy, dz ) {
       var matrTrasl = new THREE.Matrix4();
       matrTrasl.set(
              1,0,0,dx, // 1ma RIGA della matrice
              0,1,0,dy, // 2da RIGA della matrice
              0,0,1,dz, // 3za RIGA della matrice
              0,0,0,1   // 4ta RIGA della matrice
       );
       return matrTrasl;
}
```

29

Esercizio: costruiamo manualmente una metrice di rotazione

Come esercizio, costruiamo manualmente anche una matrice di rotazione attorno all'asse delle Z
(di nuovo, avremmo potuto usare funzioni esistenti di `three.js`)

```
function matriceDiRotazioneZ( angoloInGradi ) {
	var matrRot = new THREE.Matrix4();
	var angoloInRadianti = angoloInGradi/180*Math.PI;
	var c = Math.cos( angoloInRadianti );
	var s = Math.sin( angoloInRadianti );
	matrRot.set(
		c,-s,0,0, // 1ma RIGA della matrice
		+s, c,0,0, // 2da RIGA della matrice
		 0, 0,1,0, // 3za RIGA della matrice
		 0, 0,0,1  // 4ta RIGA della matrice
	);
	return matrRot;
}
```

30

Campo matrix della mesh: assegnamento automatico da parte di three.js

- ✓ Il campo `matrix` della mesh contiene la model-matrix
  - ⇒ altri campi utili presenti: matrice `modelViewMatrix` che viene automaticamente aggiornato
- ✓ Per manipolare questa matrice in modo intuitivo, Three.js consente di specificare alcuni campi di mesh che determinano la posizione, orientamento e la scala dell'oggetto:
  - ⇒ Scalatura (`scale`)
  - ⇒ Rotazione (`rotation`)
  - ⇒ Traslazione (`position`)
- ✓ Di default, `matrix` viene automaticamente settata in modo da posizionare e orientare l'oggetto nella posizione voluta
- ✓ Cioè, viene settata al prodotto di tre metrici: di traslazione, rotazione, e scaling $M = T \cdot R \cdot S$
- ✓ Come esercizio, a noi ora interessa settare questa la matrice di modellazione "a mano", decidendo noi il suo valore

31

Assegnamo la matrice di modellazione a mano

Assegnamo la nostra matrice come ModelView dell'oggetto miaMesh.

Per far questo, dobbiamo prima disabilitare il meccanismo che costruisce automaticamente la matrice in funzione dei campi "posizione, rotazione, e scala" dell'oggetto.

```
miaMesh.matrixAutoUpdate = false;
      // altrimenti Three.js la sovrascrive (come descritto sopra)
miaMesh.matrix = matriceDiRotazioneZ( 30 );
```

Nota: ruotare attorno all'asse Z in spazio vista (o clip) significa ruotare il disegno in 2D.
Per noi, lo spazio vista è anche lo spazio oggetto e mondo.

32

Animazione nella pagina web in JavaScript

E' molto semplice, in JavaScript, produrre un'animazione in una pagina web.

1. Scriviamo la funzione che intendiamo eseguire in ogni frame.
   (qui: incrementa una variabile "`angolo`", assegna la matrice di modellazione della mesh come una rotazione attorno all'asse delle Z dell'`angolo` corrente, e ripete il rendering)
2. In fondo alla funzione, chiediamo alla macchina virtuale JavaScript di eseguire la funzione stessa nuovamente, una volta appena sia necessario un nuovo frame
3. Invocare la funzione una prima volta (che ne richiederà una 2da invocazione, che ne richiederà una 3za, etc)

```
var angolo = 0; // una variabile globale

function eseguimiOgniFrame() {
        angolo += 1; // un grado a frame

        miaMesh.matrix = matriceDiRotazioneZ( angolo );
        renderizzatore.render(miaScena, miaCamera );

        // "per cortesia, esegui questa stessa funzione al prossimo frame"
        requestAnimationFrame( eseguimiOgniFrame );
}

eseguimiOgniFrame();
```

Nota JavaScript: occhio alla differenza fra *invocare* una funzione (con "`()`"), come nell'ultima riga, e *riferirsi* ad una funzione (senza "`()`"), come l'argomento che passiamo a `requestAnimationFrame` per indicare cosa essere invocato nel prossimo frame

33