Animación

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática

Curso 2019-2020

Contenidos

1. Introducción
2. Animación procedural
3. Animación mediante escenas clave
4. Animación mediante caminos

Objetivos

- Conocer los tipos de animación existentes
- Programar animaciones controlando la velocidad
- Programar animaciones mediante escenas clave
- Programar animaciones mediante caminos

Animación

Introducción

- **Animación:** Creación de la ilusión de que las cosas cambian.
- Se basa en el fenómeno de la persistencia de la visión.
- Percepción de movimiento: 24 imágenes por segundo.

Clasificación

- **Animación convencional**
  - Orientada principalmente a animación 2D con apariencia plana
  - *Ventajas:* Mayor flexibilidad y expresividad en los personajes
  - *Desventajas:* Creación de todos los dibujos a mano
- **Animación asistida por ordenador**
  - Se usa el ordenador en algunas fases del proceso: creación de dibujos, coloreado, ...
  - *Ventajas:* Permite automatizar ciertos procesos reiterativos
- **Animación por ordenador**
  - Orientada principalmente a animación 3D con entornos complejos
  - *Ventajas:* Automatismo y manejo de grandes cantidades de información
  - *Desventajas:* Falta de expresividad

Animación por ordenador

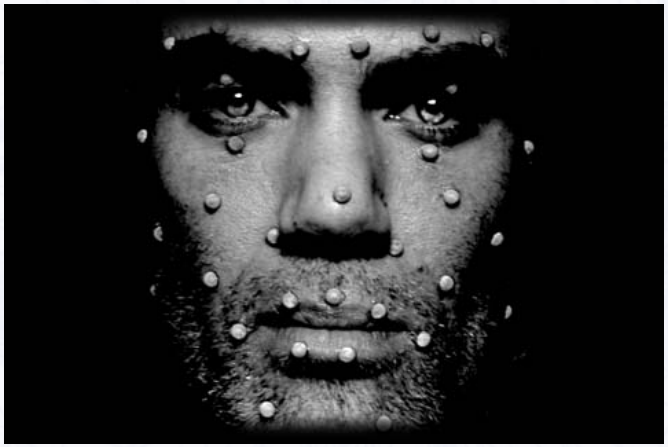
El problema de la falta de expresividad

- **Motion capture**
  - Se busca dotar de expresividad humana a los personajes ...

Animación por ordenador

Motion capture

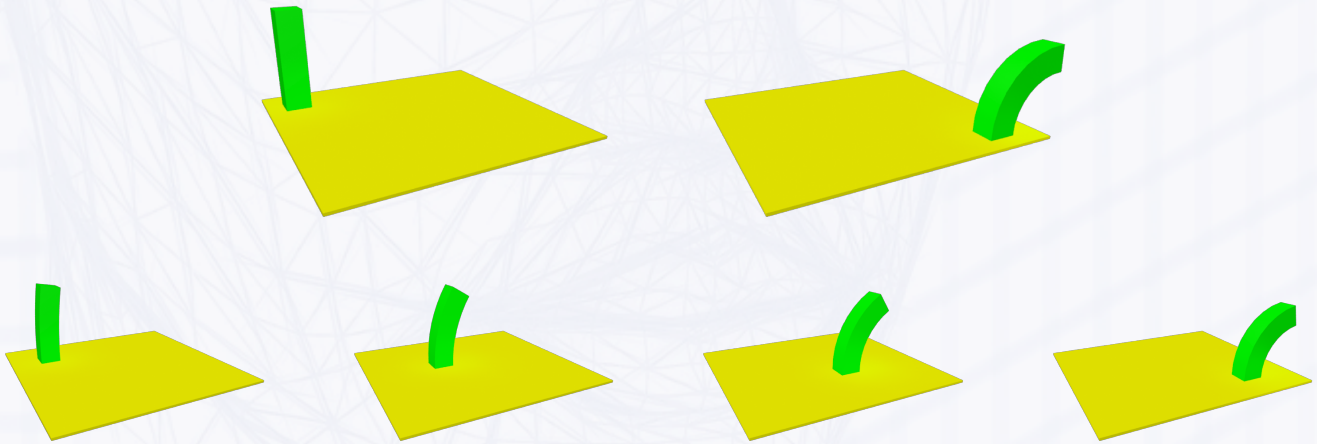
- ... también a nivel expresión facial



Animación Convencional vs. por Ordenador

- Animación Convencional (o Asistida)

- Animación por Ordenador



Etapas en una animación por ordenador

- Un corto de animación por ordenador requiere varias etapas
  - Guion, Storyboard, grabación de los diálogos, etc.

  (los detalles en la asignatura de 4^o)

- **Guion (Script)**

  Uno de los aspectos más importantes de la animación.

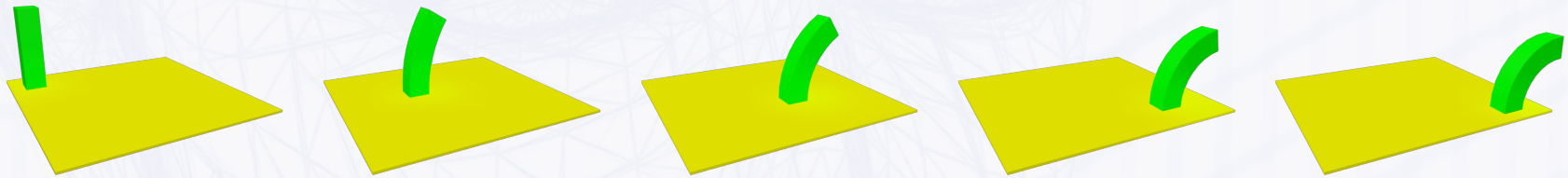
  ¿Qué se quiere contar?

- **Esquema de la historia (Storyboard)**

  Resumen gráfico (en viñetas) de la historia

Modos de implementar la animación

- Animación procedural
  - Modificando valores de parámetros
- Mediante escenas clave
  - Se indican valores concretos de parámetros en frames concretos
  - El ordenador calcula los valores en los frames intermedios



- Mediante caminos
  - La posición de un objeto viene determinada por una línea

Animación procedural Three.js

- Cada objeto animable dispone de un método que:
  - Lee el tiempo
  - Modifica los parámetros que correspondan
- Dicho método es llamado para cada frame
- Permite una animación muy personalizada

Algoritmo: Método `update` de la clase `Game`

```
Game.deltaTime = this.clock.getDelta();

this.gameObjects.forEach (function (gameObject) {
  gameObject.update();
});
```

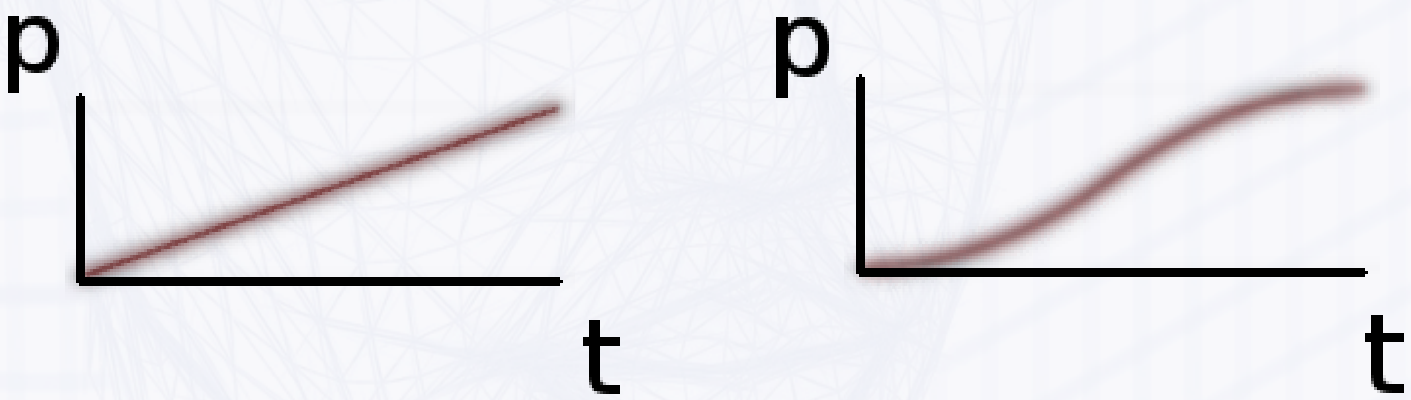
La clase Game sería una clase que puede controlar diversos aspectos del juego.

En el ejemplo, tiene la lista de objetos que hay que animar en cada frame.

Animación procedural

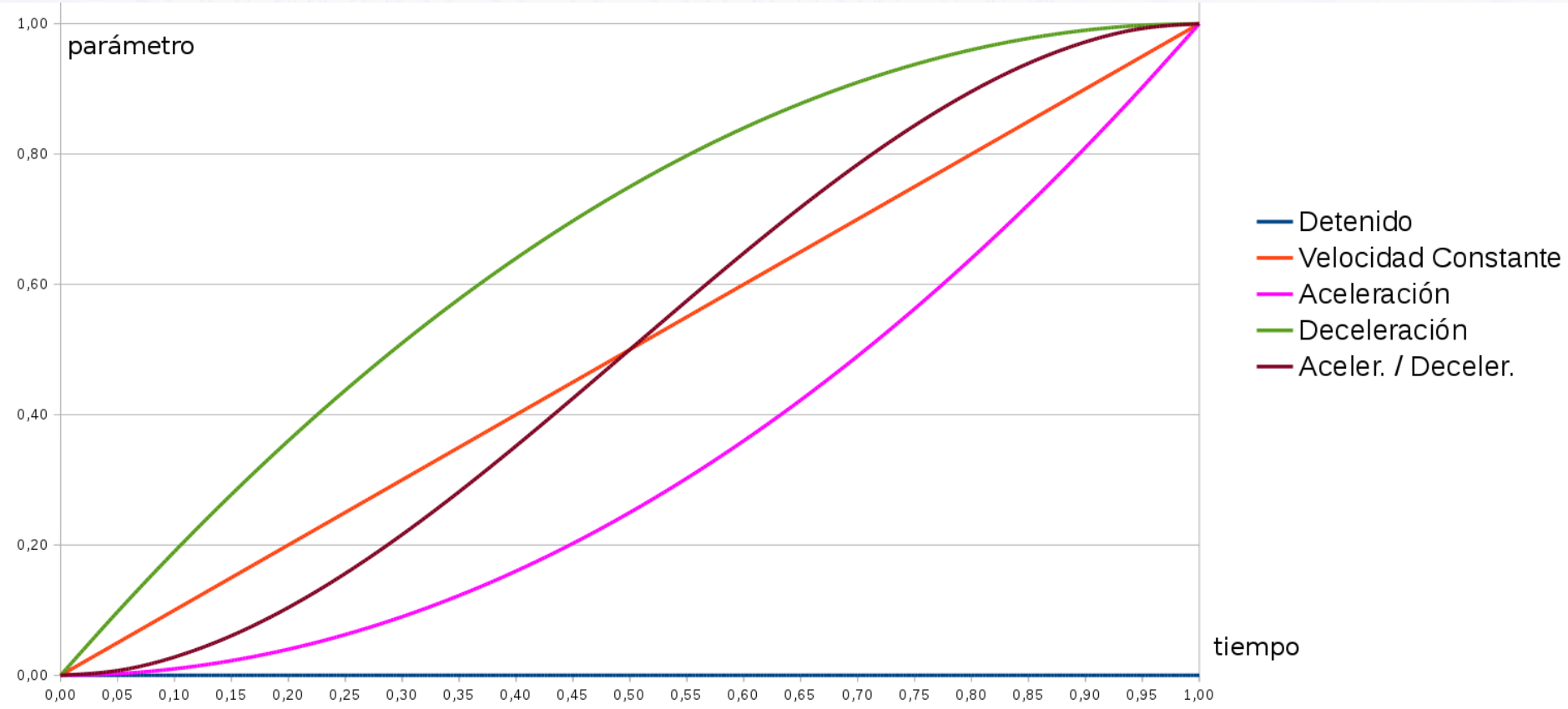
Método update

- Tiene toda la responsabilidad de la animación
- Debe conocer qué movimientos puede hacer la figura
- Saber en qué estado se encuentra cada movimiento
- Saber y controlar en qué momento ocurre cada cosa y a qué velocidad



Curvas de función: Controlan la velocidad de cambio de los parámetros

Curvas de función



Curvas de función

Expresiones aritméticas

- Premisas
  - El parámetro p varía entre v_0 y v_1
  - El tiempo t varía entre t_0 y t_1
- El valor actual de p se calcula como $p = v_0 + f(\lambda) \cdot (v_1 - v_0)$
  - Donde $\lambda = \frac{t-t_0}{t_1-t_0} \in [0,1]$
  - $f(0) = 0$
  - $f(1) = 1$
- $f(\lambda)$ determina la forma de la función (ejemplos)
  - Velocidad Constante: $f(\lambda) = \lambda$
  - Aceleración: $f(\lambda) = \lambda^2$
  - Deceleración: $f(\lambda) = -\lambda^2 + 2 \cdot \lambda$
  - Aceleración al comienzo, deceleración al final: $f(\lambda) = -2 \cdot \lambda^3 + 3 \cdot \lambda^2$

Velocidad independiente del ordenador

- En muchas ocasiones un objeto se mueve modificando su posición una determinada cantidad en cada frame
  - Por ejemplo, `objeto.position.x += 1;`
- Sin embargo, si un ordenador 'A' es capaz de renderizar el doble de frames/segundo que otro ordenador 'B',
  dicho objeto se moverá el doble de rápido en 'A' que en 'B'
- ¿Cómo conseguir que los objetos se muevan a la velocidad deseada en todos los ordenadores?
  - Recurrimos a la ecuación de la cinemática $e = v \cdot t$
    - ¿Cuánto debemos incrementar la posición del objeto en cada frame?
    - El resultado de multiplicar la velocidad deseada por el tiempo que transcurrió desde el último frame.

Velocidad independiente del ordenador

Ejemplo

Ejemplo: Velocidad independiente del ordenador

```
// Al crear el objeto medimos el tiempo actual
this.tiempoAnterior = Date.now();    // Medido en milisegundos
// Se tiene en un atributo la velocidad
//    (expresada como unidades / segundo)
this.velocidad = 10;


// En el método update(), que se ejecuta en cada frame
//    y actualiza el objeto
var tiempoActual = Date.now();
var segundosTranscurridos = (tiempoActual-this.tiempoAnterior)/1000;
objeto.position.x += this.velocidad * segundosTranscurridos;
this.tiempoAnterior = tiempoActual;
```

Animación mediante escenas clave

- La animación se analiza y descompone en movimientos sencillos
- En cada movimiento se eligen momentos importantes
  - El principio, el final, tal vez algún momento intermedio
- Esos momentos importantes son las Escenas clave
- Para cada escena clave se definen los valores concretos de los parámetros que determinan la posición de la figura.
- La animación queda configurada mediante:
  - La definición de cada escena clave
  - El tiempo que transcurre entre cada 2 escenas clave
  - La definición de la gráfica que controla el ritmo de ese movimiento
  - La definición de cómo se encadenan los diferentes movimientos

Animación mediante escenas clave Three.js

- Se usa la biblioteca **Tween.js**
  - `https://github.com/tweenjs/tween.js/`
- Cada animación Tween es un movimiento entre un origen y un destino
- Se definen 2 variables locales con los parámetros a usar en el movimiento
  - Cada variable local contiene los valores para los parámetros
  - Una variable con los valores para el origen y otra para el destino
- La animación se completa indicando:
  - El tiempo, en ms, que transcurre entre el origen y el destino
  - Cómo se modifican los parámetros de las figuras según los parámetros de las variables locales usadas en la animación

Animación con Tween Three.js

Ejemplo: Uso de Tween.js

```
// La figura que se quiere animar
this.figura = new THREE.Mesh ( . . . );

// Variables locales con los parámetros y valores a usar
var origen  = { x:   0, y: 300 };
var destino = { x: 400, y:  50 };

// Definición de la animación: Variables origen, destino y tiempo
var movimiento = new TWEEN.Tween(origen).to(destino, 2000); // 2 seg

// Qué hacer con esos parámetros
var that = this;
movimiento.onUpdate (function(){
  that.figura.position.x = origen.x;
  that.figura.position.y = origen.y;
});

// La animación comienza cuando se le indique
movimiento.start();

// Hay que actualizar los movimientos Tween en la función de render
TWEEN.update();
```

Biblioteca Tween.js

Control de la velocidad (1)

- Se realiza con el método `easing(param)`
- Donde `param` puede ser
  - Velocidad constante
    TWEEN.Easing.Linear.None
  - Aceleración al empezar y/o deceleración al acabar
    TWEEN.Easing.Quadratic.InOut
    - ★ Cambiando InOut por In o Out,
      hace que sea solo aceleración o deceleración
    - ★ Cambiando Quadratic por Cubic, Quartic, Quintic, Exponential
      se consigue una mayor aceleración/deceleración

Biblioteca Tween.js

Control de la velocidad (y 2)

- Con retroceso
  TWEEN.Easing.Back.InOut

- Elástico
  TWEEN.Easing.Elastic.InOut

- Rebote
  TWEEN.Easing.Bounce.InOut

  - ★ En los tres, cambiando InOut por In o Out, hace que el efecto se produzca solo al principio o al final

Biblioteca Tween.js

Ajuste de otros aspectos de la animación

- Número de repeticiones
  - Método `repeat (n)`
    Se puede indicar `Infinity` para repeticiones infinitas
- Movimiento de vaivén
  - Método `yoyo (true)`
- Acciones a realizar antes y después de la animación
  - Método `onStart (function () { ... })`
  - Método `onComplete (function () { ... })`
- Encadenamiento de animaciones
  - Método `chain (otraAnimacion)`
- Detención de una animación
  - Método `stop ()`

Biblioteca Tween.js Three.js

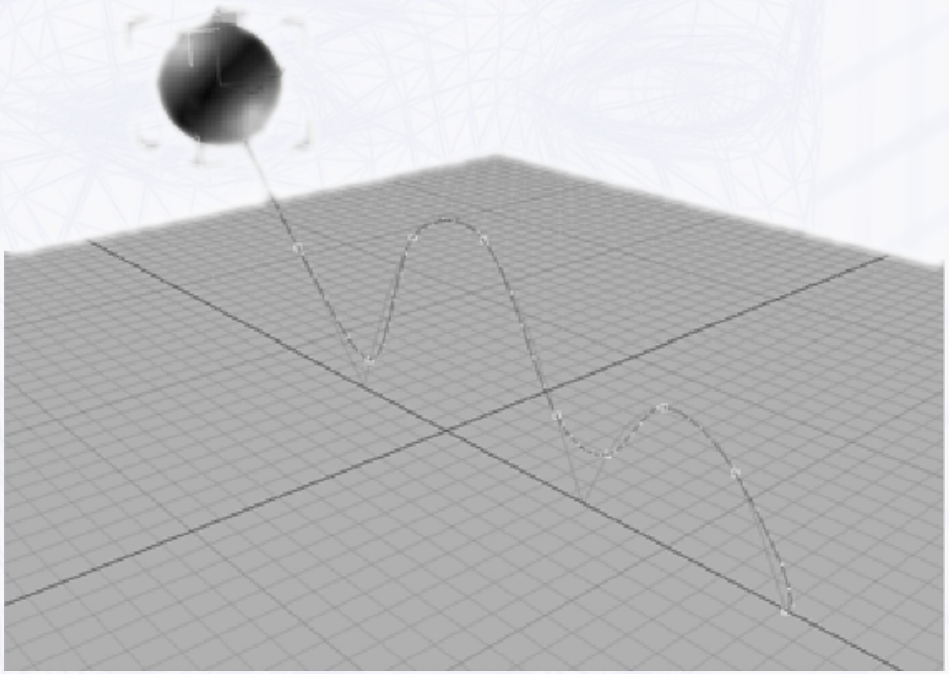
- Cada método devuelve el propio objeto
  - Se pueden encadenar los mensajes, definiendo la animación de una manera muy compacta

Ejemplo: Definición de una animación

```
var origen  = { p :   0 };
var destino = { p : 100 };
var that = this;
var movimiento = new TWEEN.Tween ( origen )
  .to (destino, 1000 )
  .easing ( TWEEN.Easing.Linear.None )
  .onUpdate (function () { that.figura.position.x = origen.p })
  .onComplete (function () { origen.p = 0; })
  .repeat (Infinity)
  .yoyo (true)
  .start();
```

Animación mediante caminos

- Se define una trayectoria
- El objeto a animar sigue dicha trayectoria



Animación mediante caminos

Procedimiento

- Se define el camino mediante un Spline

- La posición y orientación del objeto a animar se toman del spline

La cámara siguel el camino marcado por el spline

Animación mediante caminos

Three.js

Definición de Splines

- Se definen indicando sus puntos de paso

Ejemplo: Definición de Splines

```
var spline = new THREE.CatmullRomCurve3 ([
  new THREE.Vector3 (0, 0, 0), new THREE.Vector3 (0, 1, 0), ... ]);
```

- Si se desea dibujar la línea

Ejemplo: Dibujado de un Spline

```
// Se crea una geometría
var geometryLine = new THREE.Geometry();
// Se toman los vértices del spline, en este caso 100 muestras
geometryLine.vertices = spline.getPoints(100);
// Se crea una línea visible con un material
var material = new THREE.LineBasicMaterial ({color: 0xff0000});
var visibleSpline = new THREE.Line (geometryLine, material);
```

Animación mediante caminos

Three.js

Uso del Spline en la animación

- Se puede obtener una posición y una dirección tangente

Ejemplo: Uso de Splines para modificar parámetros

```
// Se necesita un parámetro entre 0 y 1
//    Representa la posición en el spline
//    0 es el principio
//    1 es el final
var time = Date.now();
var looptime = 20000;    // 20 segundos
var t = ( time % looptime ) / looptime;

// Se coloca y orienta el objeto a animar
var posicion = spline.getPointAt (t);
object.position.copy (posicion);
var tangente = spline.getTangentAt (t);
posicion.add (tangente); // Se mira a un punto en esa dirección
object.lookAt (posicion);
// Lo que se alinea con la tangente es la Z positiva del objeto
```

Animación combinando técnicas

- El método onUpdate de TWEEN es como un método update personalizado pero
  - Es llamado solo cuando esa animación está activa
- Se puede usar Tween para interpolar un parámetro entre 0 y 1 con una determinada curva de velocidades
  - Usar dicho parámetro para obtener la posición y orientación de una trayectoria (spline)
  - Y posicionar un objeto en dicho camino
- Se puede usar TWEEN para interpolar un parámetro entre 0 y 1
  - Nos indica que porcentaje ha transcurrido de un movimiento mayor
  - E iniciar y detener animaciones en función de eso

Animación

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática

Curso 2019-2020