Animación

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática

Curso 2023-2024

Contenidos

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 ...

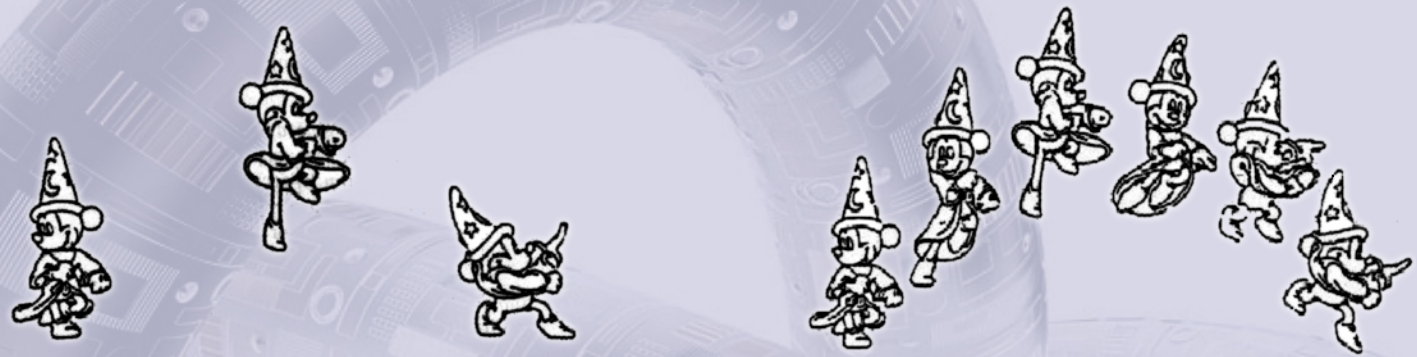
Animación por ordenador

Motion capture

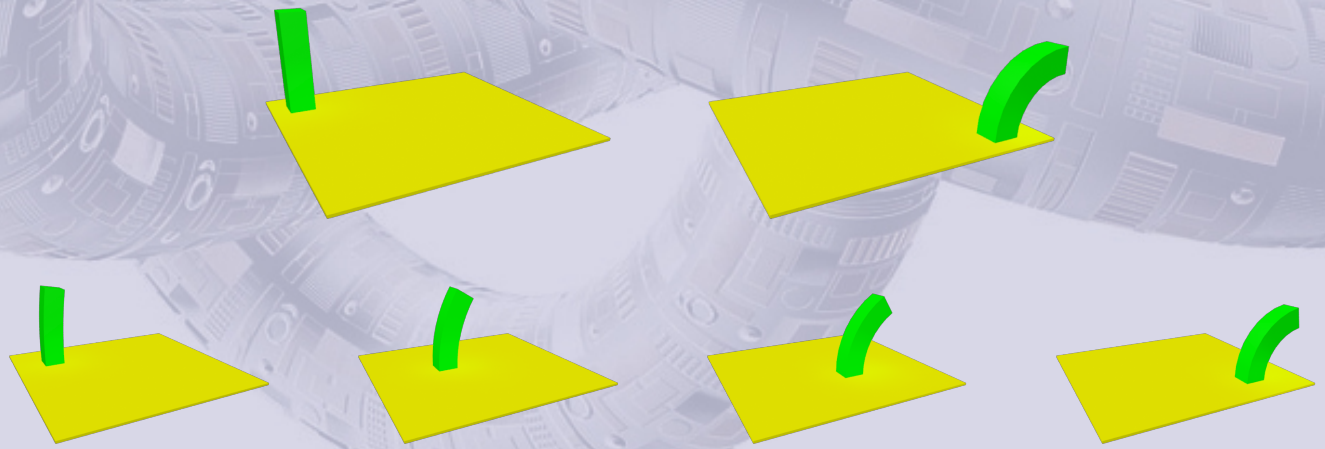
- ... también a nivel expresión facial

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.

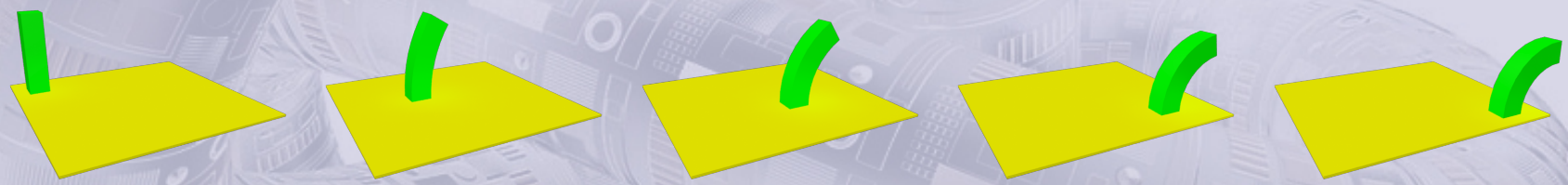
  ¿Qué se quiere contar?

- **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 `MyScene`

```
this.objetosAanimar.forEach ((unObjeto) => {
  unObjeto.update();
});
```

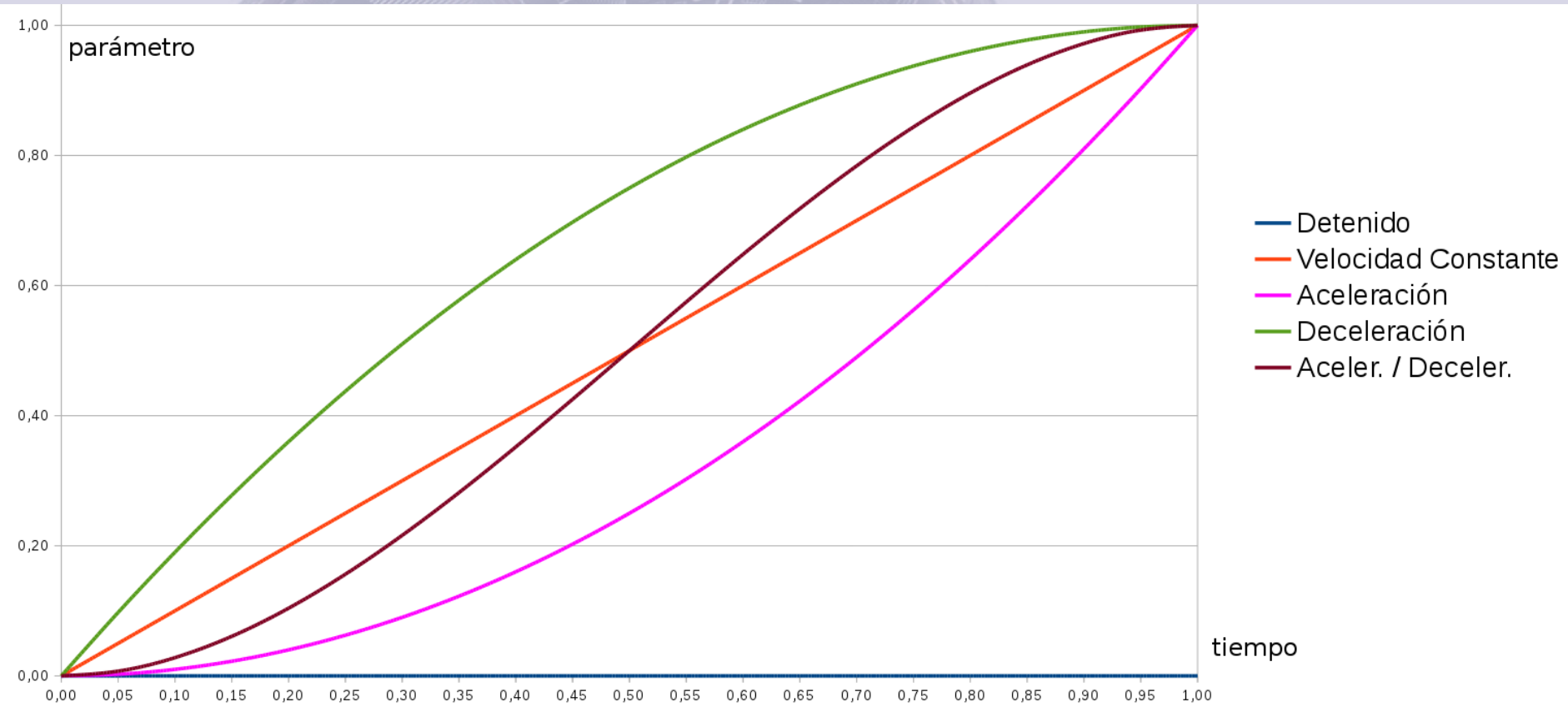
Animación procedural

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 algebraicas

- 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?
      $e += \Delta e$ donde $\Delta e = v \cdot \Delta t$
    - El incremento depende del tiempo transcurrido desde el último frame

Velocidad independiente del ordenador

Ejemplo

Ejemplo: Velocidad independiente del ordenador

```
// Al crear el objeto, creamos y referenciamos un reloj
this.reloj = new THREE.Clock();

// 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 segundosTranscurridos = this.reloj.getDelta();  // segundos desde la última llamada
objeto.position.x += this.velocidad * segundosTranscurridos;
```

- Dado que `getDelta()` devuelve los segundos transcurridos desde la última llamada

!! Cada elemento necesita su propio reloj (no se puede compartir reloj)

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/`
  - `import * as TWEEN from '../libs/tween.esm.js'`
- Tween es un interpolador de variables
- Esa interpolación se usará para cambiar la figura o escena a animar entre 2 escenas clave
- Se definen 2 variables locales con los parámetros a interpolar (parámetros de interpolación)
  - Cada variable local contiene los valores de interpolación
  - Una variable con los valores de inicio y otra con los de fin
- 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 o escena (parámetros de animación) según los parámetros de interpolación

Animación con Tween Three.js

Ejemplo: Uso de Tween.js

```
import * as TWEEN from '../libs/tween.esm.js'

...

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

// Variables locales con los parámetros de interpolación 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 de interpolación?
//    Modificar los parámetros de animación de la figura
movimiento.onUpdate (() =>{
  this.figura.position.x = origen.x;
  this.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

Ajuste de otros aspectos de la animación

- Número de repeticiones
  - Método `repeat (n)`
    Se puede indicar `Infinity` para repeticiones infinitas
  - Método `onRepeat (() => { ... })`
    Acciones a realizar tras completar cada repetición
- Movimiento de vaivén
  - Método `yoyo (true)`
- Acciones a realizar antes y después de la animación
  - Método `onStart (() => { ... })`
  - Método `onComplete (() => { ... })`
- Encadenamiento de animaciones
  - Método `chain (otrasAnimaciones, ...)`
- Pausado y continuación de una animación
  - Métodos `pause ()` y `resume ()`
- Detención de una animación
  - Método `stop ()` (se volvería a iniciar con `start()`)

Biblioteca Tween.js Three.js

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

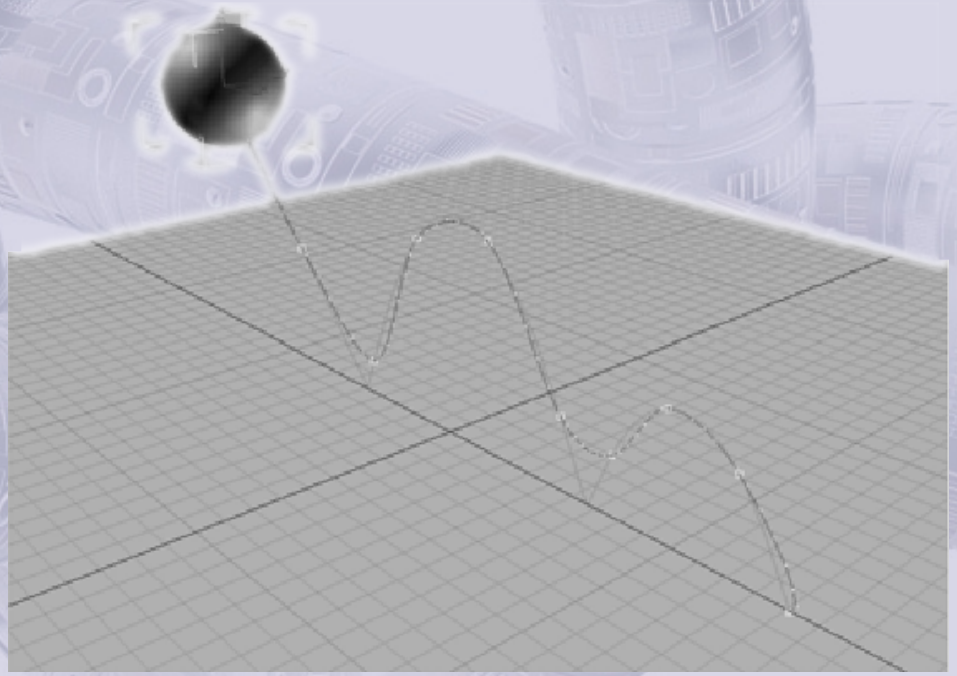
Ejemplo: Definición de una animación

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

- Es imprescindible programar `onComplete` para dejar los parámetros de interpolación en sus valores iniciales

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), ... ]);
  // Además del array de puntos
  //    se puede añadir un segundo parámetro (true) para obtener un spline cerrado.
  //    En ese caso, se asume que el spline acaba en el primer punto,
  //    pero NO hay que repetirlo
```

- Si se desea dibujar la línea

Ejemplo: Dibujado de un Spline

```
// Se crea una geometría
var geometryLine = new THREE.BufferGeometry();
// Se toman los vértices del spline, en este caso 100 muestras
geometryLine.setFromPoints (spline.getPoints(100));
// Se crea una línea visible con un material
var material = new THREE.LineBasicMaterial ({color: 0xff0000, linewidth: 2});
var visibleSpline = new THREE.Line (geometryLine, material);
```

Animación mediante caminos Three.js

Uso del Spline en la animación

- Se obtiene una posición, una binormal y una tangente del spline que se usan para posicionar y orientar la figura

Ejemplo: Uso de Splines para animar una figura

```
// Se necesita un parámetro entre 0 y 1 que indica la posición en el spline
//   0 es el principio y 1 es el final. Se puede usar Tween para ello
// También se necesita haber extraído del spline un vector de binormales
this.segmentos = 100;
this.binormales = this.spline.computeFrenetFrames (this.segmentos, true).binormals;
  // Se pone el parámetro true solo si la curva es cerrada

// Entonces la animación sería
var origen = {t : 0};
var fin = {t : 1};
var tiempoDeRecorrido = 4000;  // 4000 ms = 4 segundos

var animacion = new TWEEN.Tween (origen).to (fin, tiempoDeRecorrido)
  .onUpdate (() => {
    // Se coloca y orienta el objeto a animar
    var posicion = this.spline.getPointAt (origen.t);
    this.figura.position.copy (posicion);
    var tangente = this.spline.getTangentAt (origen.t);
    posicion.add (tangente); // Se mira a un punto en esa dirección
    this.figura.up = this.binormales[Math.floor(origen.t * this.segmentos)];
    this.figura.lookAt (posicion);
    // Lo que se alinea con la tangente es la Z positiva del objeto
  });
```

Animación

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática

Curso 2023-2024