Técnicas de animación

P. J. Martín, A. Gavilanes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Escenas animadas

- Si se muestran con rapidez varias escenas estáticas, relacionadas temporalmente de forma incremental, se percibe una escena animada.

```
while(……){              //Bucle update-render
   update scene;        //Actualizar la escena,
                        //avanzando un paso de tiempo
   render scene;        //Visualizar la escena
   glutSwapBuffers(); //Intercambiar buffers
}//while
```

- Se percibe un movimiento continuo a partir de un conjunto discreto de instantáneas estáticas.

Doble buffer

- *Viewable (**v**, front buffer)* y *drawable(**d**, back buffer)* bufferes
- Bucle: se muestra el frame actual ***v*** mientras se dibuja en ***d***. Cuando se termina de dibujar en ***d***, se intercambian los bufferes
- Implementación por hardware (ping-pong buffering, memorias VRAM) o por software (copia de ***d***, memoria RAM, en ***v***, memoria VRAM)
- Single buffering vs double buffering
  - Flickering, ghosting

Doble buffer en el esqueleto

- Activación del modo doble/single buffer en el `main()`

```
glutInitDisplayMode(● | GLUT_RGBA);
```

donde ●ϵ{GLUT_DOUBLE, GLUT_SINGLE}

- Activación del intercambio de bufferes en el `display()` con:

```
glutSwapBuffers();
```

o con:

```
glFlush();
```

Control de la animación

1. De forma **interactiva** en respuesta a **eventos de teclado**.
   - ✓ Cada vez que se pulsa una determinada tecla se actualiza la escena y se marca como pendiente de visualización.

```
// Keyboard input processing routine
void keyInput(unsigned char key, int x, int y){
    …
    switch(key) {
       case 'a': // Move the scene to the next state
            update scene;
            glutPostRedisplay();
            break;
       …
    }//switch
}
```

Control de la animación

2. De forma **automática** mediante un **temporizador**.
   - ✓ Cada vez que transcurre un intervalo de tiempo se actualiza la escena y se marca como pendiente de visualización.
   - ✓ En freeglut, la función **`glutTimerFunc`** registra el método que se ejecutará (`animate`) cuando pase el tiempo dado (`animatePeriod`).

```
// Timer function
void animate(int value){
      update scene;
      glutTimerFunc(animationPeriod, animate, 1);
      glutPostRedisplay();
}
```

   - ✓ El reloj se activa o detiene en respuesta a eventos de teclado!

Control de la animación

3. De forma **automática** mediante la función **idle**.
   - ✓ En freeglut, la función idle se ejecuta cuando ningún otro evento de OpenGL está pendiente de ser resuelto.
   - ✓ Se registra con la función `glutIdleFunc` registra el método que se ejecutará cuando él programa esté en espera:
     - ✓ `glutIdleFunc(NULL)` elimina la función idle actualmente registrada.
     - ✓ `glutIdleFunc(my_idleFunc)` registra la función `my_idleFunc`.

```
// Idle function
void my_idleFunc(void){
        update scene;
        glutPostRedisplay();
}
```

   - ✓ La función idle registrada se modifica en respuesta a eventos de teclado!

Algunas técnicas de animación

1. Basadas en transformaciones

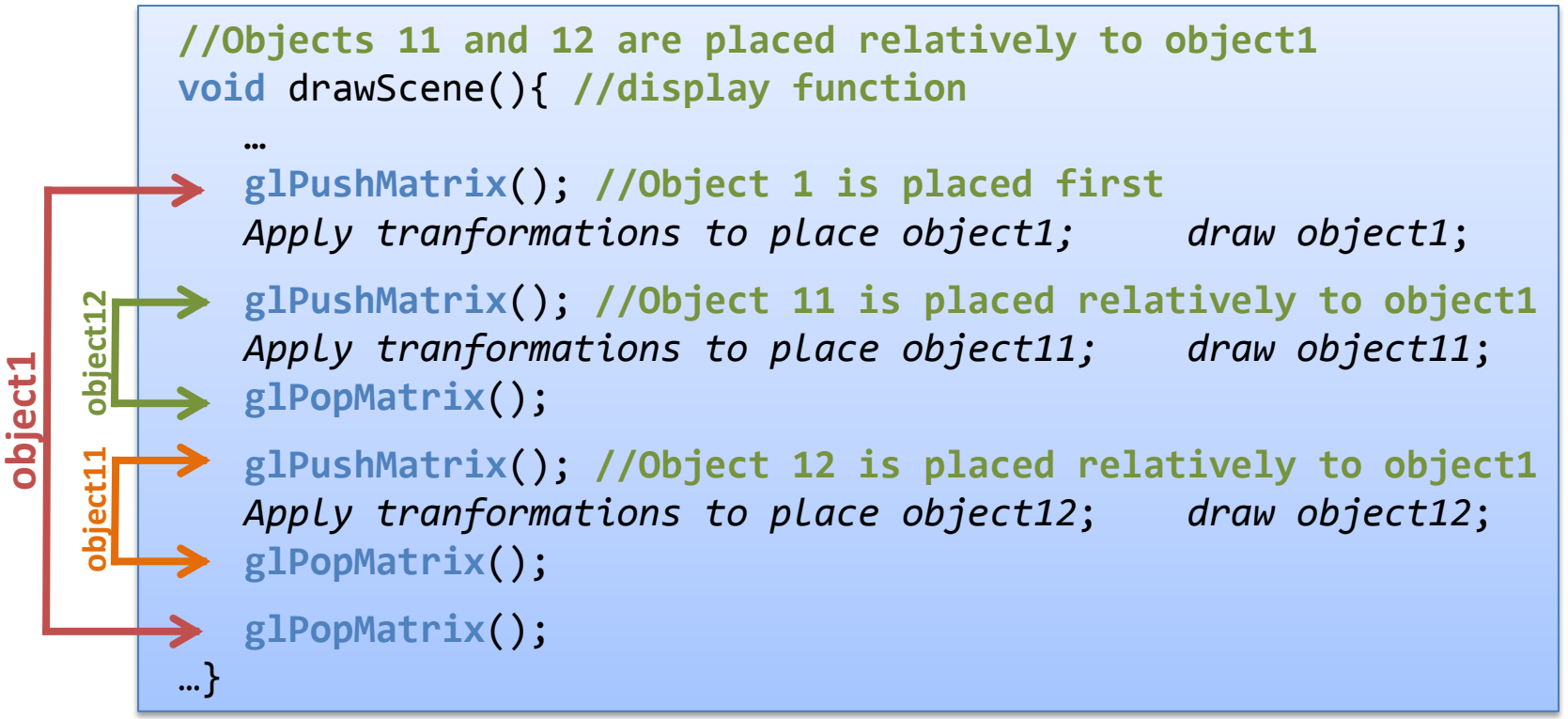
- ✓ Cada objeto de la escena actualiza su ubicación aplicando una transformación.
- ✓ Las transformaciones se van acumulando según avanza la animación.

```
//Variable angle controls the rotation of the object
void increaseAngle(){
   angle += 5.0;
   if (angle > 360.0) angle -= 360.0;
   glutPostRedisplay(); }

void drawScene(){  //display function
   glPushMatrix();
   glRotatef(angle, 0.0, 1.0, 0.0);
   draw the object;
   glPopMatrix();
   glutSwapBuffers(); }
```

Algunas técnicas de animación

- ✓ Los objetos de pueden organizar usando el *modelo jerárquico:* un objeto se coloca en función de la colocación de otro
  - Las ruedas de un coche en función de la carrocería…
  - La cabeza de un robot en función del tronco…



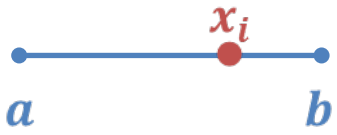

```
//Objects 11 and 12 are placed relatively to object1
void drawScene(){ //display function
   …
   glPushMatrix(); //Object 1 is placed first
   Apply tranformations to place object1;     draw object1;
   glPushMatrix(); //Object 11 is placed relatively to object1
   Apply tranformations to place object11;    draw object11;
   glPopMatrix();
   glPushMatrix(); //Object 12 is placed relatively to object1
   Apply tranformations to place object12;    draw object12;
   glPopMatrix();
   glPopMatrix();
…}
```

Algunas técnicas de animación

2. Basadas en ecuaciones paramétricas

- ✓ El estado del objeto (posición, orientación, tamaño…) se calcula en función de un parámetro $\boldsymbol{t}$ que representaría el tiempo.
- ✓ La **interpolación lineal** se usa para calcular los estados intermedios entre el estado inicial y el final.
- ✓ En el caso 1D, si interpolamos desde $\boldsymbol{a}$ hasta $\boldsymbol{b}$ en $\boldsymbol{n}$ pasos, los valores intermedios que resultan son:

$$x_i = a + i\frac{b-a}{n} = \left(1 - \frac{i}{n}\right)a + \frac{i}{n}b, \quad 0 \le i \le n$$



Si usamos $t_i = \frac{i}{n}$ (la familia de instantes en que se divide el intervalo [0,1] de forma equilibrada), resulta:

$$\boldsymbol{x_i = (1 - t_i)a + t_i b}$$

Combinación afín convexa de a y b (en 1D)

Algunas técnicas de animación

- ✓ Objetivos de la interpolación lineal
  - La posición desde el punto P hasta el punto Q (2D/3D)

$$\boldsymbol{x_i = (1 - t_i)P + t_iQ, \qquad 0 \leq i \leq n}$$

  - La orientación desde el ángulo α hasta β (2D)

$$\boldsymbol{\theta_i = (1 - t_i)\alpha + t_i\beta, \qquad 0 \leq i \leq n}$$

  - El tamaño desde el factor g hasta h (2D/3D)

$$\boldsymbol{f_i = (1 - t_i)g + t_ih, \qquad 0 \leq i \leq n}$$

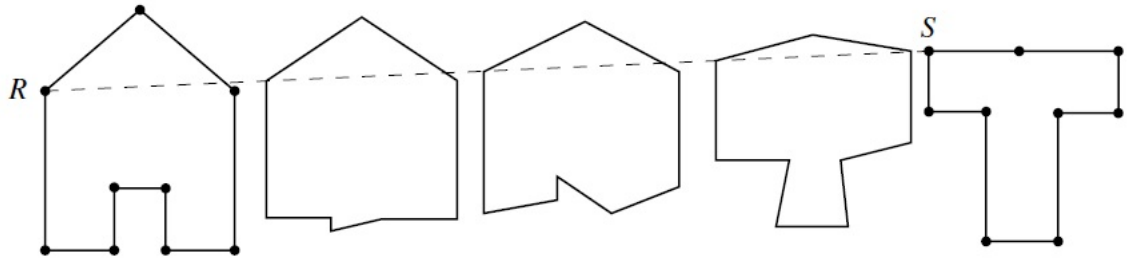
  - El color desde $c_0 = (r_0, g_0, b_0)$ hasta $c_1 = (r_1, g_1, b_1)$ (2D/3D)

$$\boldsymbol{c_i = (1 - t_i)c_0 + t_ic_1, \qquad 0 \leq i \leq n}$$

  - La propia geometría del objeto: ***morphing***
- ✓ Aplicación de la interpolación:
  - Sobre el modelo del objeto: se modifican sus coordenadas
  - Sobre la representación del objeto: se aplican transformaciones

Morphing

- ✓ El ***morphing*** es una técnica que sirve para transformar una figura en otra de forma animada.
- ✓ El diseñador sólo modela estas dos figuras, y utiliza el computador para calcular las figuras intermedias mediante interpolaciones.



- ✓ Las figuras de control se llaman ***key frames***.

Morphing

1. Se establece un relación uno a uno entre los vértices de la figura inicial y los de la final. Típicamente tienen el mismo número de vértices.
2. Entre para cada pareja de vértices se aplica una interpolación lineal, pero usando en todas ellas el mismo valor de t.

```
//Morphing between the figures figureIn and figureOut
PV2D tween( PV2D in, PV2D out, double t){
    return (1-t)*in + t*out;
}
void buildTween( int n, PV2D figureIn[], PV2D figureOut[],
                 double t, PV2D figureInBetween){
    PV2d p;
    for (int i=0; i<n; i++){
      figureInBetween[i]= tween( figureIn[i], figureOut[i],t);
    }//for
}
```

Algunas técnicas de animación

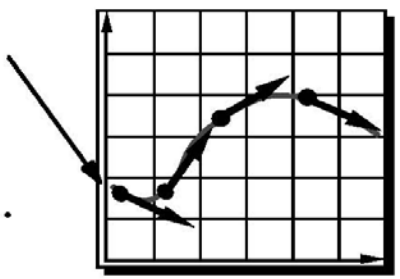
3. Basadas en principios físicos

- Ecuaciones diferenciales ordinarias

$$\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}, t)$$

Expresa cómo varía la variable $\boldsymbol{x}$ en función del tiempo t.
Típicamente $\boldsymbol{x}$ y f son vectores.
La función f es conocida.
$\boldsymbol{x}$ depende de t, podría ser la posición de una partícula en 2D/3D.

- Integración numérica
  - ✓ Se conoce el valor inicial $\boldsymbol{x}(t_0) = \boldsymbol{x_0}$
  - ✓ Se trata seguir la evolución de $\boldsymbol{x}$ según avanza t



Initial Value Problem