Arquitecturas Reconfigurables

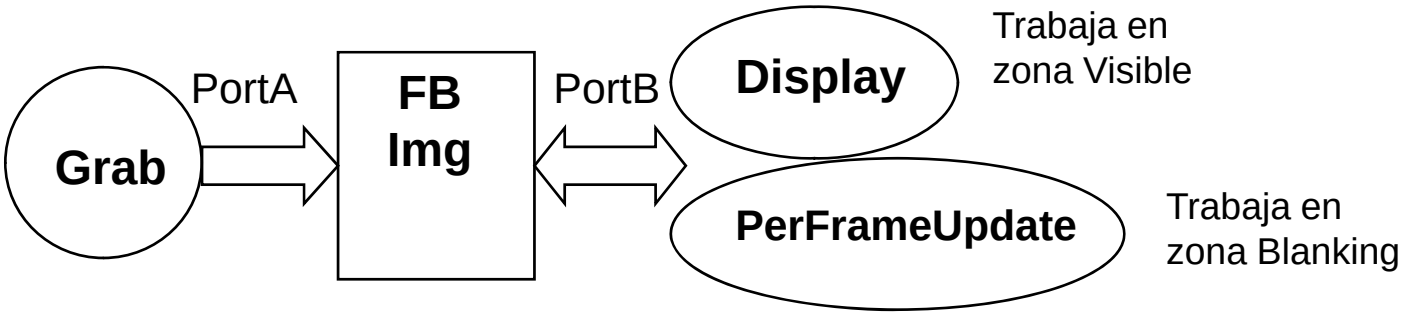
Tutorial 5: EyeToy

Profesores: Sergio Cuenca y Antonio Martínez
sergio@dtic.ua.es
Dept. Tecnología Informática y Computación
Universidad de Alicante

Estructura básica

```
mpram FB  //memoria de imagen doble puerto
{
    ram unsigned 16 A[128*128]; // Read/write port
    rom unsigned 16 B[128*128]; // Read only port
} M with {block = "BlockRAM"};

void main (void){
  par{
    while (1){
        Grab(); //captura y escribe en framebuffer
        Display(Video,BallX,BallY); // mezcla imagen y juego
        PerFrameUpdate(Video,BallX,BallY); // actualiza juego
    }
  }
}
```



Captura y segmentación de imágenes

```
macro proc Grab() {
  unsigned X, Y, Pixel;
  macro expr Red      = Pixel[15:11];
  macro expr Green    = Pixel[10:5];
  macro expr Blue     = Pixel[4:0];

  while(1) {
    RC10CameraReadRGB565(&X, &Y, &Pixel);
    if(X<128 & Y<128) // se guardan 128*128 pixels
      M.A[(Y<-7)@(X<-7)]= (Red>16&&Green>16)? 0xffff : Pixel;
    else
      delay; //se descartan el resto
  }
}
```

Detección de objetos en imagen

- **Varios métodos de Segmentación de los objetos de interés**

Umbralización

Si los niveles de color de los pixels superan ciertos umbrales => pertenecen al objeto.
Se puede caracterizar el color de la piel mediante ciertos intervalos de RGB para segmentar la manos o la cara.

Movimiento

Detección y marcado de los pixels que cambian en la escena.
Es necesario almacenar la imagen de la escena en instantes anteriores para compararla con la nueva imagen.

Flujo óptico

Detección de movimiento y su dirección.

En que fase segmentar? Opciones

- durante la captura (se guardan las coordenadas del objeto)
- durante el PerFrameUpdate (se segmentan solo los pixels afectados por el juego)

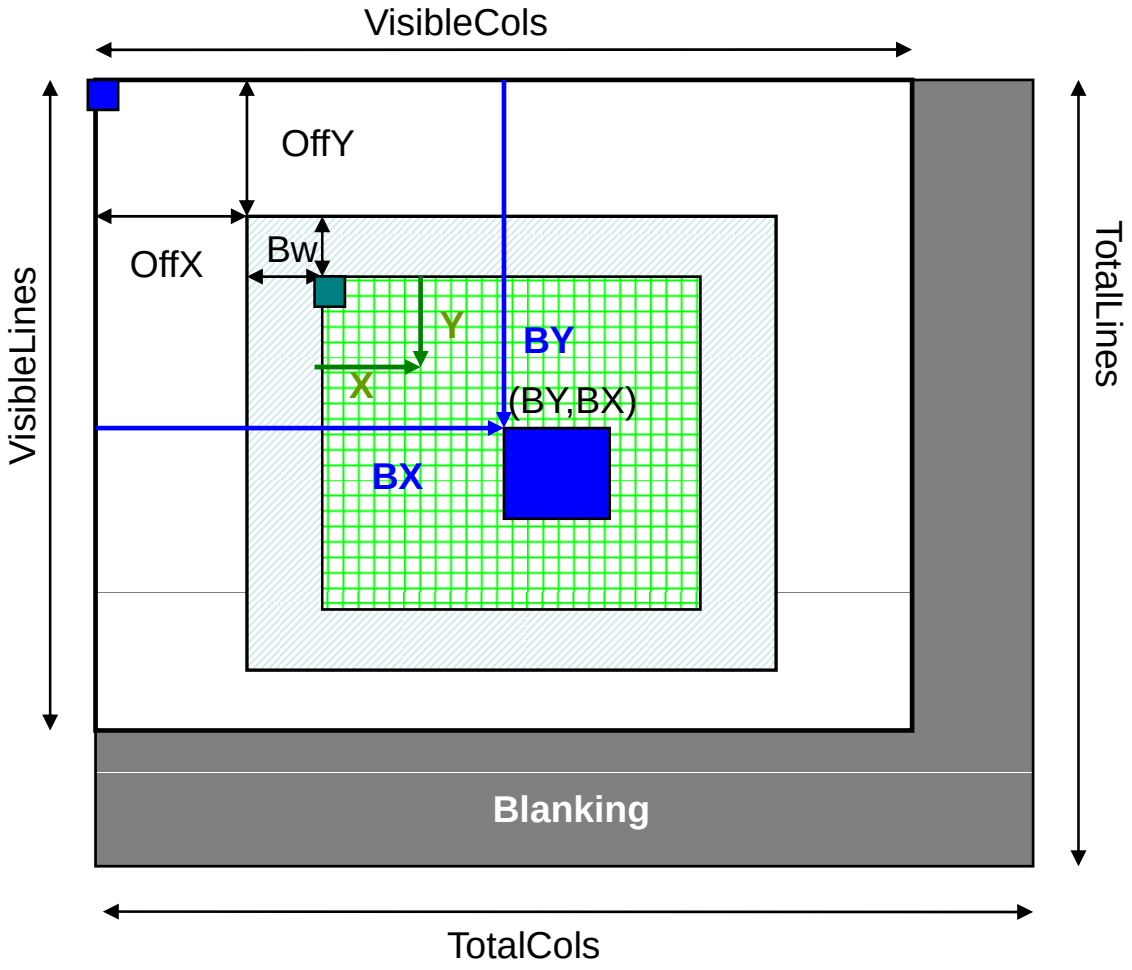
Generación juego (imagen+juego)

```
macro proc Display (Video,BallX,BallY){
while (1) {
   par {
    InGameX ; InGameY; //verifico si estamos en tablero de juego
    InBallX; InBallY ; //verifico si estamos en bola
    if (sx<VisibleCols & sy<VisibleLines ){
      if (InGameX==1 && InGameY==1 ) {
        par{
            Pixel= M.B[(Fil[8:2])@(Col[8:2])]; // escalado *4
            if (InBallX==1 && InBallY==1)
              PalVideoOutWrite (Video,BlueColor); //bola
            else
              PalVideoOutWrite (Video,Pixel);  // imagen
        } //par
      } //if
      else
        PalVideoOutWrite (Video,RedColor);
    }
    else
        PalVideoOutWrite (Video,BlackColor);
   }//par
}//while
```

PerFrameUpdate

```
while (1) {
  while(sx!=VisibleCols || sy!=(VisibleLines-1)){
    delay; // espera zona blanking
  }
  par { //detecta colisiones con bordes
    if (BallX > (OffX+ANCHO*4+BORDER_WIDTH-BALL_SIZE-SPEED) ||
                BallX < (OffX+BORDER_WIDTH+SPEED))
       ColV=1; //Colisión bordes verticales
    else ColV=0;
    if (BallY > (OffY+ALTO*4+BORDER_WIDTH-BALL_SIZE-SPEED) ||
                BallY < (OffY+BORDER_WIDTH+SPEED))
       ColH=1; //Colisión bordes horizontales
    else ColH=0;
  }
 // detecta colisiones con objetos de imagen
 par{ // actualiza posición bola
   BallX = dx ? BallX+SPEED : BallX-SPEED;
   BallY = dy ? BallY+SPEED : BallY-SPEED;
 }
}
```

Detección de colisiones con objetos de la imagen. Coordenadas



Coordenadas bola (BY,BX) relativas a origen de monitor

Coordenadas imagen (Y,X) Relativas a origen imagen

Para que imagen y bola interaccionen deben conocerse sus coordenadas respecto de la misma referencia

Dado una posición de la bola (BY,BX) ¿a que posición de la imagen corresponde?

Xb=BX-OffX-BW
Yb=BY-OffY-BW

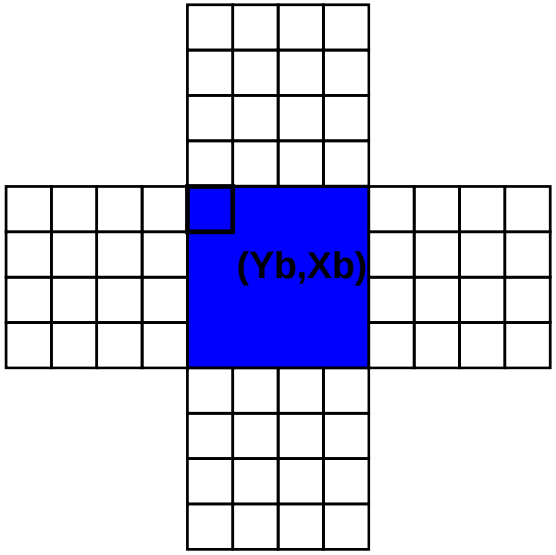
Interacción imagen-juego

¿Cómo saber si la bola en (BY,BX) chocará con el objeto de la imagen (Yobj,Xobj)?

1) Calculamos la posición de la bola dentro de la imagen

Xb=BX-OffX-BW
Yb=BY-OffY-BW

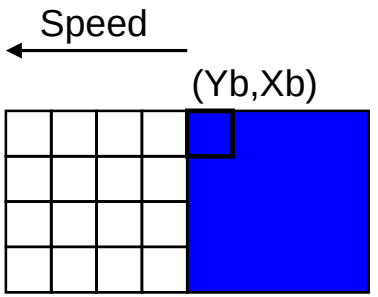
2) Miramos si la siguiente posición de la bola coincidirá con alguno de los pixels del objeto en la imagen



Interacción imagen-juego

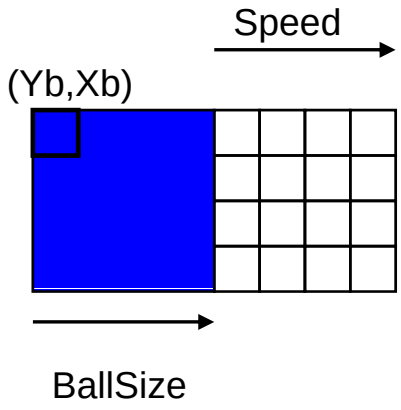
Si se mueve hacia la izquierda (dx=0)
con velocidad 4 leemos de la imagen los pixels que ocuparía

```
For (f=0; f<4; f++)
  For(c=1; c<5; c++)
    Pix= Img(Xb-c,Yb+f);
    If (Pix∈Objeto)
      invertir_dirección;
    else
      delay;
  Endfor;
Endfor;
```



Si se mueve hacia la derecha (dx=1)

```
For (f=0; f<4; f++)
  For(c=1; c<5; c++)
    Pix= Img(Xb+BallSize+c,Yb+f);
    If (Pix∈Objeto)
      invertir_dirección;
    else
      delay;
  Endfor;
Endfor;
```



Interacción imagen-juego: efecto del escalado

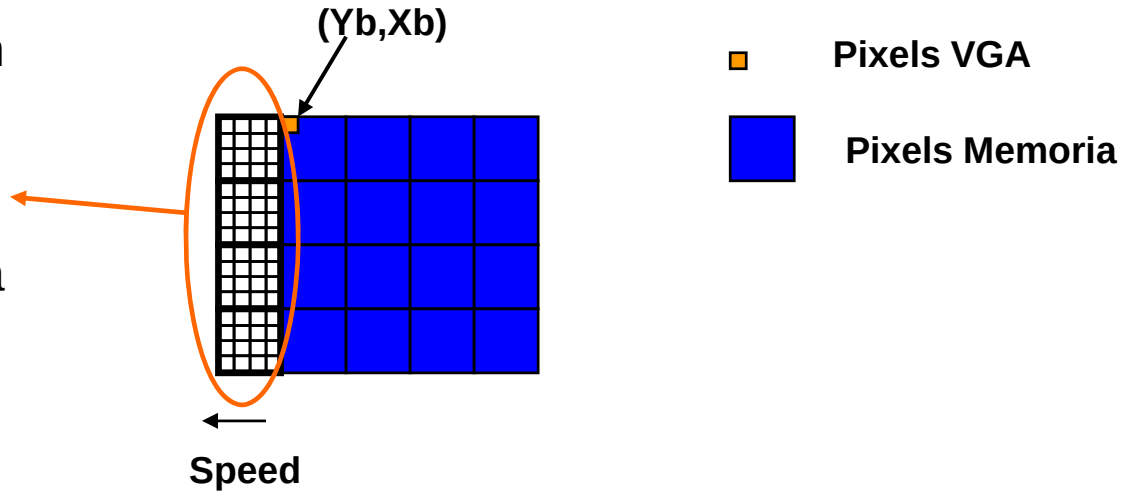
Si existe escalado en la visualización de la imagen => un pixel en memoria corresponde a varios pixels visualizados (VGA)

Ejemplo: escalado x4 en ambas direcciones

tamaño bola = 16x16 pixels VGA, Speed=4 pixels VGA

- 1 pixel en memoria corresponde a 16 pixels en VGA
- la bola ocupa 16 posiciones de memoria

Para buscar interacción hay que mirar en las cuatro posiciones de memoria anteriores a la de la bola



X=Ceil[Xb/SCALA] ⇔ X=Xb>>LOG2SCALA; Ej:Si Xb= 9 =>X=Ceil[9/4]= 2

Limitaciones arquitectura

- Capacidad FPGA

  Spartan3 1500
  Memoria en BlockRAM: 576Kbits
  Nº BlockRam= 32

  Tamaño BlockRaM= 18Kb => 16Kb+2Kb
  Configuraciones posibles: 16Kx1, 1Kx16.

  Memoria imagen 128x128x16b= 262.144bits
  NºBloques ocupados = 128x128 pixels= 16K pixels x 16b/pixel
  16 bloques configurados como 16Kx1 = 50% de los bloques

Tamaño máximo imagen ?

Limitaciones arquitectura

- Camara CMOS

code	*resolution*
CIF	352 x 288
QCIF	176 x 144
QQCIF	88 x 72
SXGA	1280 x 1024
VGA	640 x 480
QVGA	320 x 240
QQVGA	160 x 120

```
macro proc RC10CameraReadRaw     (XPtr, YPtr, ValuePtr);
macro proc RC10CameraReadRGB565  (XPtr, YPtr, ValuePtr);
macro proc RC10CameraReadYUV     (XPtr, YPtr, ValuePtr);
```

Propuestas

- Modificar el sistema de segmentación por otro invariante a iluminación:

- Si utilizamos un señuelo ROJO, realizar una calibración de los umbrales calculando el máximo nivel de R en la imagen sin el señuelo => umbral = Rmax+c

Modificar el sistema de segmentación para que segmente los niveles pertenecientes a piel (manos)

Interacción con partes fijas del juego (cambio de color, aceleración de la música, etc...)