Topic 3: (Part B)
Advance Graphics in Ext. Rotoscale
Affine Matrix Manipulation

Systèmes Embarqués Microprogrammés

Prof. David Atienza Alonso, SEL-STI

Affine transformation matrix

- Graphical engines calculate the inverse operation of an affine transformation: for a given pair of coordinates, the corresponding pixel in the original image is obtained
  - They avoid making calculations for areas out of the screen
- The screen can be sequentially refreshed (pixel by pixel) from left to right horizontally and line by line from top to bottom
  - The engines calculate the coordinates of the pixel in the original image that corresponds to a specific pair of coordinates ***(x,y)***

$$(x, y) = (x', y', 1) * \begin{pmatrix} xdx, xdy \\ ydx, ydy \\ dx, dy \end{pmatrix}$$

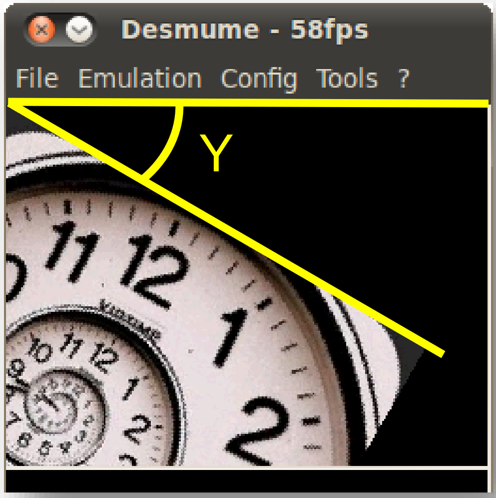
    – ***(x', y')*** are the coordinates of the pixel to render on the screen
    – ***(x, y)*** are the coordinates of the pixel value en the original image

EPFL

Image rotation and translation

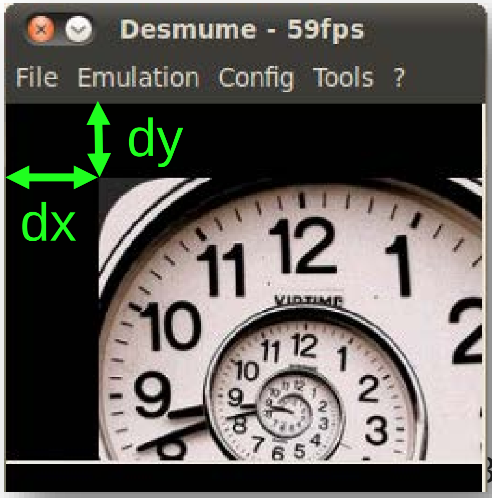
- The image to be rendered in the screen can be rotated clockwise Y radians with respect to the upper-left corner of the screen

$$\begin{pmatrix} xdx, xdy \\ ydx, ydy \\ dx, dy \end{pmatrix} = \begin{pmatrix} \cos(Y), -\sin(Y) \\ \sin(Y), \cos(Y) \\ 0, 0 \end{pmatrix}$$



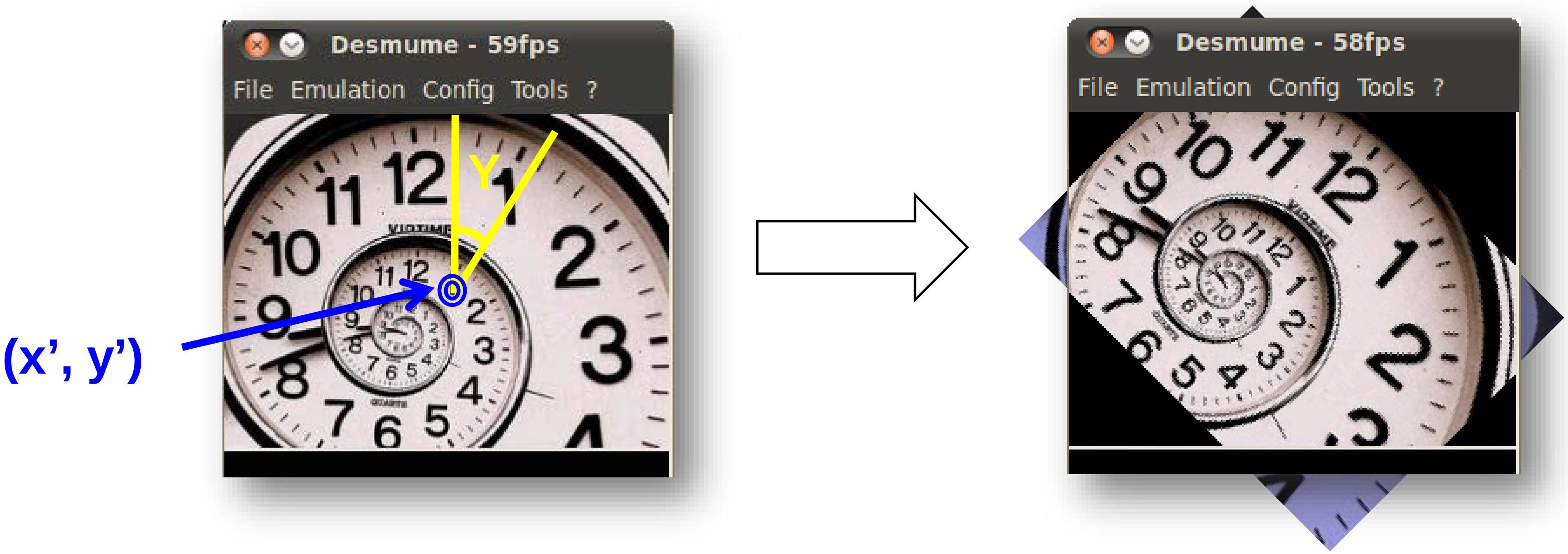
- It can also be translated with respect to the coordinates of the system (upper-left corner)

$$\begin{pmatrix} xdx, xdy \\ ydx, ydy \\ dx, dy \end{pmatrix} = \begin{pmatrix} 1, & 0 \\ 0, & 1 \\ -50, & -40 \end{pmatrix}$$



Rotation with respect to another point

- Can the the image be rotated Y radians with respect to another pair of coordinates (x', y')?



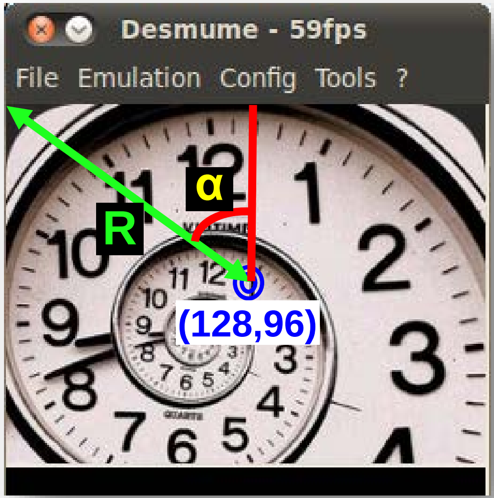
$$\begin{pmatrix} xdx, xdy \\ ydx, ydy \\ dx, dy \end{pmatrix} = \begin{pmatrix} \cos(Y), -\sin(Y) \\ \sin(Y), \cos(Y) \\ ?, \quad\quad ? \end{pmatrix}$$

Rotation with respect to the center of the screen (0 rads)

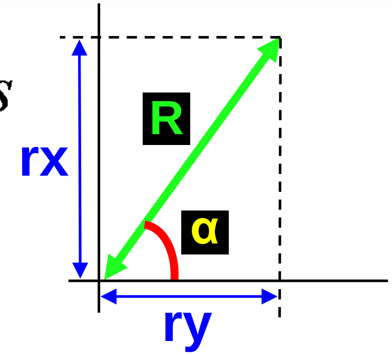
- Let's assume **no rotation**, i.e. the displacement in x and y (dx and dy) **must be 0**
  - R is the distance from the rotation point (x', y') to the system origin and to the upper left image corner

$$R = \sqrt{(x')^2 + (y')^2} = \sqrt{128^2 + 96^2} = 160$$

  - α is the angle defined by the coordinates of the rotation point and the rotation angle β (0 rads in this case)



$$\alpha = \arctan\left(\frac{x'}{y'}\right) + \beta = \arctan\left(\frac{128}{96}\right) + 0 = 0,9273\,rads$$



  - dx and dy are the difference between the coordinates of the rotation point (x', y') and the projections of R in the x and y axis (rx and ry)

$$dx = x' - rx = x' - R * \sin(\alpha) = 128 - 160 * 0,8 = 0$$

$$dy = y' - ry = y' - R * \cos(\alpha) = 96 - 160 * 0,6 = 0$$

Rotation with respect to the center of the screen (0.5 rads)

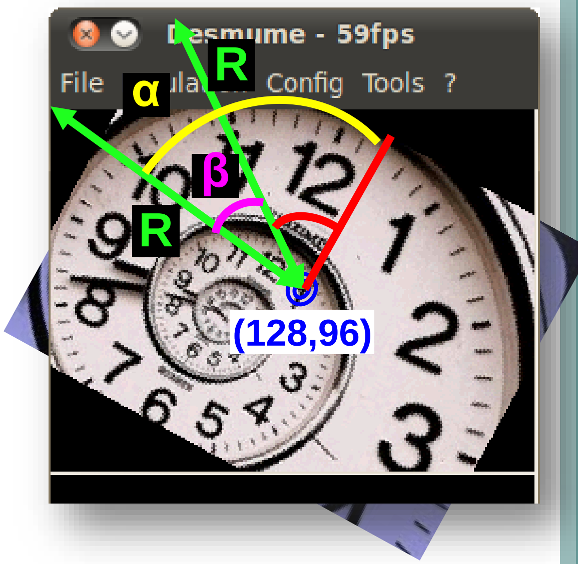
- Let’s rotate the image 0.5 radians (~28,7°) with respect to the center of the screen
  - R is still the distance from the rotation point (x’, y’) to the system origin **and** to the upper left image corner

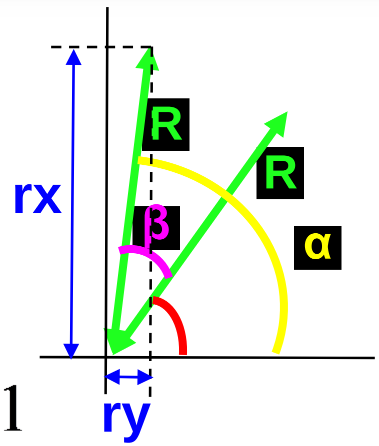
$$R=\sqrt{(x')^2+(y')^2}=\sqrt{128^2+96^2}=160$$

  - α is the angle defined by the coordinates of the rotation point and the rotation angle β(0.5 radians)

$$\alpha=\arctan\left(\frac{x'}{y'}\right)+\beta=0,9273+0.5=1.4273\,rads$$

  - dx and dy are the difference between the coordinates of the rotation point (x’, y’) and the projections of R in the x and y axis (rx and ry)





$$dx=x'-rx=x'-R*\sin(\alpha)=128-160*0,99=-31$$

$$dy=y'-ry=y'-R*\cos(\alpha)=96-160*0,14=73$$

C implementation (Parameter calculation)

- The function **rotateImage_BG2**(**int** *x*, **int** *y*, **float** *angle_rads*) modifies the affine transform matrix of BG2 of the MAIN graphical engin to rotate the displayed image clockwise an angle of “angle_rads” radians with respect to the rotation point with coordinates “x” and “y”

1. Calculate R:

```c
//Distance from rotation point to system origin
float r = sqrt(x*x + y*y);
```

2. Calculate angle α

```c
//Angle alpha
float alpha = atan((float)x/(float)y) + angle_rads;
```

C implementation (Configuration of affine matrix)

3. Configure image rotation (with input argument)

```
//Image rotation
REG_BG2PA = cos(angle_rads) * 256;  //xdx
REG_BG2PB = sin(angle_rads) * 256;  //xdy
REG_BG2PC = -sin(angle_rads) * 256; //ydx
REG_BG2PD = cos(angle_rads) * 256;  //ydy
```

4. Configure image translation (with angle α and coordinates of rotation point)

```
//Image translation
REG_BG2X = (x - r*sin(alpha)) * 256;  //dx
REG_BG2Y = (y - r*cos(alpha)) * 256;  //dy
```

Final C implementation

```c
#include "math.h"

void rotateImage_BG2(int x, int y, float angle_rads) {

    //Distance from rotation point to system origin
    float r = sqrt(x*x + y*y);

    //Angle alpha
    float alpha = atan((float)x/(float)y) + angle_rads;

    //Image rotation
    REG_BG2PA = cos(angle_rads) * 256;  //xdx
    REG_BG2PB = sin(angle_rads) * 256;  //xdy
    REG_BG2PC = -sin(angle_rads) * 256; //ydx
    REG_BG2PD = cos(angle_rads) * 256;  //ydy

    //Image translation
    REG_BG2X = (x - r*sin(alpha)) * 256;  //dx
    REG_BG2Y = (y - r*cos(alpha)) * 256;  //dy
}
```

Implementation remarks

- In order to write this implementation, trigonometric functios for the sine (sin), cosine (cos) and actangent (atan) have been used.
  - These functions are included in the standard C math library (see math.h)
  - These functions receive arguments of type ***float*** representing an angle in radians (in the case of sine and cosine) or the tangent (in the arctangent function)
  - These functions are usually computing-intensive, therefore the improper usage can lead to large execution delays.
- 1 radian = 180 / π degrees
- The values of the affine transform matrix are stored in Fixed-point arithmetic using 8 bits for the decimal part of the number
  - Therefore, the values are shifted 8 bits (or multiplied by 256)
- The used registers (REG_BG2PA, REG_BG2PB, etc) correspond to the background affine matrix of BG2 of the main engine
  - Similar registers can be used for the BG3 and for the SUB engine