

Computer Graphics Labs

Abel J. P. Gomes

LAB. 3

Department of Computer Science and Engineering
University of Beira Interior
Portugal
2011

Copyright © 2009-2011 All rights reserved.

LAB. 3

BASIC ANIMATION AND COLLISIONS

1. Learning goals
2. Timing and redisplay callbacks
3. Example
4. Programming exercises

Lab. 3

BASIC ANIMATION AND COLLISIONS

In this lecture we intend to learn the basic of animation and collision detection when objects move around a 2D scene.

are going to deal with geometric transformations in 2D as their generalization in 3D is straightforward.

These geometric transformations are also called affine transformations.

1. Learning Goals

At the end of this chapter **you should be able to:**

1. Use geometric transformations with variable parameters to animate objects within a 2D scene.
2. Use timing callbacks to refresh a dynamic scene that changes over time.
3. Detect collisions of moving 2D objects against the scene borders.
4. Detect collisions between objects in a 2D scene.

2. Timing and Redisplay Callbacks

To animate an object within a scene, we need to move the object and redisplay the scene repeatedly within a short period of time. For that purpose, we need to register a timer callback that then calls the GLUT redisplay command called `glutPostRedisplay()`. See Example below for further details (lines 55-60). The `glutPostRedisplay()` command simply calls the display function called `DESENHAR` every 33 milliseconds.

Registering a timer callback is carried out through the following command:

`glutTimerFunc` — registers a timer callback to be triggered in a specified number of milliseconds.

Prototype:

```
void glutTimerFunc(unsigned int msec, void(*func)(int value), value);
```

where:

- msec: number of milliseconds to pass before calling the callback;
- func: the timer callback function;
- value: integer value to pass to the timer callback.

The number of milliseconds is a lower bound on the time before the callback is generated. GLUT attempts

to deliver the timer callback as soon as possible after the expiration of the callback's time interval.

3. Example

The following program displays a moving square within a sub-domain with a basic collision detection of the bounding edges of such a sub-domain.

```
1 #include <GLUT/glut.h>           // Header File For The GLut Library
2
3 // Initial square position and size
4 GLfloat x1 = 10.0;
5 GLfloat y1 = 0.0;
6 GLfloat size = 2.5;
7 // Step values for coordinates
8 GLfloat xstep = 0.25;
9 GLfloat ystep = 0.25;
10
11
12 void quad()
13 {
14     glBegin(GL_QUADS);
15         glVertex2f( 0.0, 0.0);
16         glVertex2f( 2.5, 0.0);
17         glVertex2f( 2.5, 2.5);
18         glVertex2f( 0.0, 2.5);
19     glEnd();
20 }
21
22
23
24 GLvoid DESENHAR(GLvoid)
25 {
26     glClearColor(0.0f,0.0f,1.0f,1.0f);
27     glClear(GL_COLOR_BUFFER_BIT);
28
29     glMatrixMode(GL_PROJECTION);
30     glLoadIdentity();
31     gluOrtho2D(0.0,40.0,0.0,40.0);
32
33     glMatrixMode(GL_MODELVIEW);
34     glLoadIdentity();
35
36     // BEGIN collision detection for a square
37     if (x1 + size > 40.0 || x1 < 0.0)
38         xstep = -xstep; // reverse direction on left or right edge
39     if (y1 + size > 40 || y1 < 0)
40         ystep = -ystep; // reverse direction on top or bottom edge
```

```

41     x1 += xstep; // update x-coordinate for square origin
42     y1 += ystep; // update y-coordinate for square origin
43     // END collision detection for a square
44
45     glTranslatef(x1,y1,0);
46     // set current drawing color to red
47     glColor3f(1.0f,0.0f,0.0f);
48     quad();
49     // flush drawing commands and swap
50     glutSwapBuffers();
51 }
52
53
54
55 GLvoid TEMPORIZADOR()
56 {
57     // Redraw the scene by calling the display function DESENHAR
58     glutPostRedisplay();
59     glutTimerFunc(33,TEMPORIZADOR, 1);
60 }
61
62
63 void ESCAPE(unsigned char key)
64 {
65     if(key==27) exit(0);
66 }
67
68
69 int main (int argc, char** argv)
70 {
71     // initialization for GLUT/OpenGL
72     glutInit(&argc,argv);
73     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
74     // create window with resolution 500x500
75     glutInitWindowSize( 500,500 );
76     glutCreateWindow("Bounce");
77     // register callbacks
78     glutDisplayFunc(DESENHAR);
79     glutKeyboardFunc(ESCAPE);
80     glutTimerFunc(33,TEMPORIZADOR,1);
81     // process events
82     glutMainLoop();
83     return 0;
84 }

```

4. Programming Exercises

1. Write a program that moves a circle inside a square box. The circle is reflected when it hits any side of the box. Let us assume that the reflection angle is 45 degrees in relation to the line perpendicular to any box side.
HINT: Use the timing callback called TEMPORIZADOR to update the movement of the ball. For that purpose, you must register such a callback by means of the statement `glutTimerFunc(33,TEMPORIZADOR,1)` in the main function. Within the TEMPORIZADOR callback, you must call the `glutPostRedisplay()` function to redraw the scene again and again every 33 milliseconds.
2. Write a program to roll a wheel on a horizontal line.
3. Write a program to roll a wheel on a bias line.
4. Write a breakout game with a ball, a paddle and a set of bricks.
5. Write a program that moves a 2.5x2.5 square inside a 40x40 domain in \mathbf{R}^2 . The corresponding viewport has 500x500 pixels. The square movement is carried out in steps of 0.025 in the x-direction and in the y-direction. The initial position of the square is at (0.25,0.00).
HINT: Use the `GL_QUADS` primitive for drawing the square, and the `glTranslatef` function to move the square.
6. Change the previous program in order to animate two squares that cross in a perpendicular manner inside the box.
7. Change the program in 5. in order to not only move it but also rotate it of 2.5 degrees at every single step.