MU2 - G Key OpenGL Component

M. Peña G., J. J. Negrete R.

Acoustics Laboratory, Superior School of Mechanical and Electric, National Polytechnic Institute ESIME-IPN, Av. Instituto Politécnico Nacional S/N. 07300 Zacatenco D.F., México Phone 5 729 6000-54652 and 54616 - E-mail: mpenag@ipn.mx, josejnegerter@yahoo.com.mx
MU2 - G Key OpenGL Component

M. Peña G., J. J. Negrete R.

Acoustics Laboratory, Superior School of Mechanical and Electric, National Polytechnic Institute ESIME-IPN, Av. Instituto Politécnico Nacional S/N. 07300 Zacatenco D.F., México Phone 5 729 6000-54652 and 54616 - E-mail: mpenag@ipn.mx, josejnegreter@yahoo.com.mx

ABSTRACT
One of the problems with which the designers face of audio systems, it is the programming of standard graphic components, because each operating system (UNIX, Win32, Mac) has different ways to make the things. In this document we present the graphic programming of the G clef as example of a component that has a form complex topology, and drawing it on a music staff. To achieve it, we program in C several polynomials (splines) linked using OpenGL functions as much in Linux as in the Win32 system.

Keywords: staff graphics, OpenGL, music, splines curves, software component.

RESUMEN
Uno de los problemas con que se enfrentan los diseñadores de sistemas de audio, es la programación de componentes gráficos estándar, pues cada sistema operativo (UNIX, Win32, Mac) tiene distintas maneras de hacer las cosas. En este documento presentamos la programación gráfica de la clave de sol como ejemplo de un componente que tiene una forma topológica compleja, y dibujado éste en un pentagrama de música. Para lograrlo, programamos en C varios polinomios (splines) concatenados utilizando funciones de OpenGL tanto en Linux como en el sistema Win32.

Palabras Clave: Pentagrama, OpenGL, música, curvas splines, componente software.

1 Introduction
In our laboratory of acoustics it is desirable to have scores that you make big or tiny without it gets lost the information of the staff. However, those graphic objects that they are defined with a fixed points group (f.e. the text fonts), they have a problem: the zoom. It is not possible to make approaches to estrangements without some type of graphic distortion is introduced in the figure.

To solve this problem, we program the G key with the following mathematical methods: the Bezier splines, and the Bernshtein equation. For the curves continuous we uses the concatenation of several of these splines. On the other hand, we use the Bernshtein algorithm to fill the figure with right lines, which are limited by those splines that outline the one mentioned figure [1].

We can say that a music staff contains graphic components that belong to a certain language type: the music language. This is similar to the computer languages as such as FORTRAN and C++, which follow rules and definitions that describe their behavior.

The symbols of the musical language are written in called objects scores. A score is formed by many components that they represent a melody: the same staff, keys, notes, and many other special symbols. It is evident that this symbols are used by the artists to communicate their musical compositions.

However, the programming of the staff visual components depends on those graphic primitive ones and basic functions (point, line, circle, etc) that provides us the one operating system that use to program our audio application [4].

In order to solve this platform dependence problem, we have programmed our key of G (and other software components) using the OpenGL library graphic functions, the one which it is independent of the operating system one where our is executed application [5].

Finally, it is necessary to mention that a software component is an independent object that can be used in diverse way ways similar to their hardware contraposition. This way, a software component can also to be used in diverse agreement applications with their application context.
2 How it work

A third order polynomial, Bezier curve, or simply Bezier spline, it is defined by the following two equations [3].

\[
\begin{align*}
  x(t) &= (1-t)^3 x_0 + 3t(1-t)^2 x_1 + 3t^2(1-t) x_2 + t^3 x_3 \\
  y(t) &= (1-t)^3 y_0 + 3t(1-t)^2 y_1 + 3t^2(1-t) y_2 + t^3 y_3
\end{align*}
\]

where \( x(t) \) and \( y(t) \) they are the coordinates of each point that it belongs to the curve. In the Figure 3 it is presented the code source that programs this equations .

In consequence, to form a Bezier figure they are only needed four points in the cartesian plane: \((x_0, y_0), (x_1, y_1), (x_2, y_2), \) and \((x_3, y_3)\). The point \((x_0, y_0)\) it is the curve start point; while the one point \((x_3, y_3)\) it is the end of the curve. On the other hand, the points \((x_1, y_1)\) and \((x_2, y_2)\) they are known as control points, those which to the one to move these, the curve it distort. This allows to draw a curve adjustable to the contour of a section of another much more complex figure [2].

The Figures 6 and 7 we view source program of the line and point primitive functions.

In the Figure 5 \(\text{DrawCuerno}()\) source program is shown that it contains the programming of two splines; the first one provides the coordinates of beginning for the function that one draws lines straight line; the second spline provides the final points of these same lines. Notice you that the function \(\text{plotPoint}(x,y)\) it draws the points of the splines, and the \(\text{Recta}(x1,y1,x11,y11)\) function it draws this line. The \(z\) variable it's the zoom value (between 0 and 1). This way, the G key is formed by many curved linked Bezier one with other and filled with right lines.

3 As a Result

In the Figure 1 a sample of the results is presented that we obtained as consequence of the programming (code source Figure 4) of the key G component. Here you can appreciate the form of the lines and curved that constitutes said elements graph. Optionally, in the Figure 2 we present one score that contains the other graphic components that were as consequence of of the present work.

Figure 1: Clef of G graphic component.  
Figure 2: Staf with some of their musical components.
Conclusion

The graphic objects as the Bezier splines, and the Bernshtein algorithm, they are formalisms of algorithms that allowed us drawing the musical clef G (or key G) symbol. We program several splines linked for to draw the contour of the figure. You straight lines they were programmed of agreement with the Bernshtein algorithm to cover with these lines the body of the figure, and whose limits are the continuous points of the splines that they form the key G contour.

References


Gratefulness

We thank the support that they have provided us the authorities of the Polytechnic Institute National (IPN), the Commission of Operation and Development of Academic Activities (COFAAA), and the Section of Studies of Posgrado and Investigation (SEPI).

Appendix

```c
void DrawSpline(GLfloat x1,GLfloat y1,
   GLfloat x2,GLfloat y2,
   GLfloat x3,GLfloat y3,
   GLfloat x4,GLfloat y4,
   GLfloat z)
{
    GLint i;
    GLint j;
    GLfloat t;
    GLfloat x;
    GLfloat y;

    for(t=0.0;t<1;t+=0.001)
    {
        x = pow((1.0-t),3.0)*(x1*z)  + 3.0*t*pow((1.0-t),2.0)*(x2*z)
         + 3*pow(t,2.0)*(1.0-t)*(x3*z)
         + pow(t,3.0)*(x4*z) ;

        y = pow((1.0-t),3.0)*(y1*z)  + 3.0*t*pow((1.0-t),2.0)*(y2*z)
         + 3*pow(t,2.0)*(1.0-t)*(y3*z)
         + pow(t,3.0)*(y4*z);

        plotPoint(x,y);
    }
}
```

Figure 3: DrawSpline() function that only draws a spline.
Figure 4: DrawCsol() function that draws the segmented body of the key of G with two splines for each segment fillers and linked.
The function `plotPoint(x11, y11);` and `Recta(x, y, x11, y11);` have been programmed to draw two splines that form and limit a part of the body of the G Key.

The `plotPoint(x, y)` function only draws a pixel.

The `Recta(x, y, x11, y11)` function iterates through a series of calculations to determine the correct pixels to draw a line from `(x, y)` to `(x11, y11).`


```
fraccion -= dy;
} 
y += stepy;
fraccion += dx;
plotPoint(x,y);
}
}
Figure 7: Recta(x0,y0,x1,y1 primitive function that draws a direct line of agreement with the Bernshtein algorithm.
```

About the authors

José de Jesús Negrete Redondo: Was born in México D.F. in 1947. Graduate as Comunications & Electronic Enineer (ESIME-IPN, 1981). He has been professor and researcher in Acoustics Academy of the National Polytechnic Institute, México, since 1980.