International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

Implementation of the Sierpinsky Triangle with Pic Microcontroller

M. A. Sandoval-Hernandez[1,2], U. A. Filobello-Nino[3], H. Vazquez-Leal[3,4], O. Alvarez-Gasca[3], A. D. Contreras-Hernandez[3], B. E. Palma-Grayeb[3], L. Cuellar-Hernandez[3], N. Carrillo-Ramón[3], M. Garcia Lozano[1], S.E. Torreblanca-Bouchan[1]

[1] Centro de Bachillerato Tecnológico industrial y de servicios No. 190, Av. 15 Col. Venustiano Carranza 2da Sección, Boca del Río, 94297, Veracruz, México.

[2] Escuela de Ingeniería, Universidad de Xalapa, Carretera Xalapa-Veracruz Km 2 No.341, 91190, Xalapa Veracruz, México.

[3] Facultad de Instrumentación Electrónica, Universidad Veracruzana, Circuito Gonzalo Aguirre Beltrán S/N, Xalapa, 91000, Veracruz, México.

[4] Consejo Veracruzano de Investigación Científica y Desarrollo, Av. Rafael Murillo Vidal No. 1735, Cuauhtémoc, Xalapa, 91069, Veracruz, México.

***Abstract*— The objective of this paper is demonstrative and educational since it proposes the implementation of the Sierpinski Triangle fractal in a development system with the Pic18F45k22 microcontroller and GLCD screen. The Sierpinski Triangle algorithm is simple and easy to implement and can be proposed as a didactic programming practice for students of the technological high school studying a technical degree in programming and computer science. Plotting times between 1.2 minutes and 4 minutes were obtained by varying the different settings in the internal oscillator of the Microcontroller as well as by varying the number of iterations within the algorithm.**

***Keywords*— Fractal, microcontroller, recursive algorithm, school math speech, development system.**

I. INTRODUCTION

A fractal is a geometric object in which the same pattern is repeated at different scales. In other words, a fractal is an object that shows self-similarity, on all scales. The term was proposed by Benoit Mandelbrot in 1977 [1]. Among the best known nonlinear fractals are the Julia and Mandelbrot sets, which are obtained from the behavior of complex numbers when they are iterated by a function. Julia sets are obtained from quadratic functions [2].

$$f_c(z) = z^2 + c, \tag{1}$$

where c is a complex number. The Julia set obtained from (1) is denoted J_c . The process to obtain this Julia set from is as follows: Any complex number z is chosen and a sequence of numbers is built iterating as follows

$$z_{n+1} = f_c(z_n) = {z_n}^2 + c. \tag{2}$$

The Mandelbrot set denoted by M is defined by the set of parameters $c \in \mathbf{C}$ for (1) connected. The Mandelbrot set is obtained by modifying Julia's iterative process in (2) by making the point c variable and setting the point $z_0 = 0$ in (1). This set is contained in a circle of radius 2 in the complex plane of the numbers c [2].

Linear fractals are constructed from a change of scale with elementary figures such triangles and squares. Examples of these fractals are the Cantor set, the Von Koch Curve and the Sierpinski Triangle [3]. For more details the fractals see [4].

Among the applications that fractals have is in image compression. This technique consists of compressing images using the fact of self-similarity at different scales of the same figure. One of the fundamental differences of this type of technique is that the compression code does not save pixels, so it is free of scales. Thus it can be decompressed at any scale without having resolution problems [5].

To graph a fractal it is required to program the recursive algorithms in programming languages such as C/C++ [6],Visual basic [7], Matlab [8], among others.

This paper is organized as follows. In Section II, we introduce the basic concept Sierpinski Triangle. Section III present basics of microntroller Pic. Implementation and simulations are provided in Section IV. Finally, a concluding remark is given in Section V.

II. SOME BASICS OF CONCEPT SIERPINSKI TRIANGLE

Initially you start from an equilateral triangle. First, it must be divided into four internal equilateral triangles by joining the midpoints of the sides, eliminating the central triangle. For each of the triangles obtained, four triangles must be generated again

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

by joining the midpoints and once again eliminating the central triangle. The process is repeated indefinitely. Figure 1 shows the procedure.

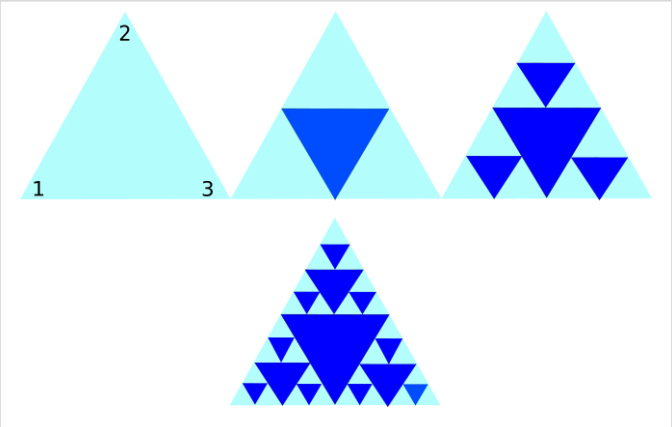

Figure 1. Construction of the Sierpinski Triangle.

Thus, the Sierpinski Triangle is built by several triangles where each of them is made up of segments that join the midpoints of each side.

To build the triangle on a Cartesian plane, we start by defining the vertices of the first triangle and it is done as follows.

1. Define 3 points with coordinates: $a(ax, ay), b(bx, by), c(cx, cy)$. These will become the corners of the outer triangle. We mark three points 1, 2, 3 in each of the vertices to identify them, see figure 1.
2. Define an arbitrary starting point $p(px, py)$. This can be in one of the vertices, in this way we will remain halfway between the selected point and the initial starting point.
3. Now change the coordinates of point p, depending on the number you have taken. The point at which we have chosen is selected randomly by rolling a dice or using a random function and with the following rules: if 1 or 2 comes out we go to point 1, if it comes out 3 or 4 we go to point 2 and if it comes out 5 or 6 we go to point 3.

$$\text{if you rolled 0, } p = (p + a)/2.$$
$$\text{if you rolled 1, } p = (p + b)/2.$$
$$\text{if you rolled 2, } p = (p + c)/2. \quad (3)$$

4. Go back to step 2, there place the point in the new position of p, and so on.

In recent years it has been proposed to introduce the basic concepts of fractal geometry including the Sierpinski Triangle in high school in order to recognize numerical and geometric patterns [9]. In the same way, it has been proposed as an academic activity in secondary education [10]. Likewise, at these levels the use of GeoGebra has been used to clarify its characteristics in an interactive way [11] and to ensure that students obtain significant knowledge [12].

III. MICROCONTROLLER PIC

A PIC is a programmable integrated circuit that contains the architecture to perform different tasks, which is why it is called a microcontroller. In order to classify the PICs, they are put into three families according to their capacity, these families are: 8 bits, 16 bits and 32 bits. These in turn have subdivisions which are already by the ranges of microcontrollers that we can use. The PIC18F family offers many advantages in its performance.

They offer numerous advantages, such as low cost efficient solutions for general purpose applications written in C that use a real-time operating system (RTOS) and require a complex communication protocol stack such as, CAN, USB, among others. PIC18F devices provide flash program memory in sizes from 8 to 128Kbytes and data memory from 256 to 4Kbytes, at speeds from DC to 64MHz [13]. For example, some characteristics of the Pic18F45K22 are [14]: 1024 Bytes Data EEPROM, 64 Kbytes Linear Program Memory Addressing, 3896 Bytes Linear Data Memory Addressing, precision 16 MHz Internal Oscillator Block with - Factory calibrated to ± 1%, selectable frequencies of 31.5 kHz, 250 kHz, 500 kHz, 1 MHz, 2 Mhz, 4 MHz, 8 Mhz, and 16 MHz; 64 MHz performance available using phase locked loop (PLL) no external components required, four Crystal modes up to 64 MHz, seven timers, etc.

IV. NUMERICAL SIMULATION AND DISCUSSION

The school mathematical discourse (DME in spanish) is identified because it is everything that remains unchanged in the classroom, it does not modify what is being taught but how it is being taught [15, 16]. To avoid DME the Sierpinski Triangle algorithm implementation was implemented in the EasyPic V7 Conectivity development system using a graphical Liquid Crystal Display (GLCD) of 128 x 64 pixels. Figure 2 shows the development system employed in this work. The algorithm formulas (3) were programmed in language using the Mikcro C compiler of MikroElektronika [13].

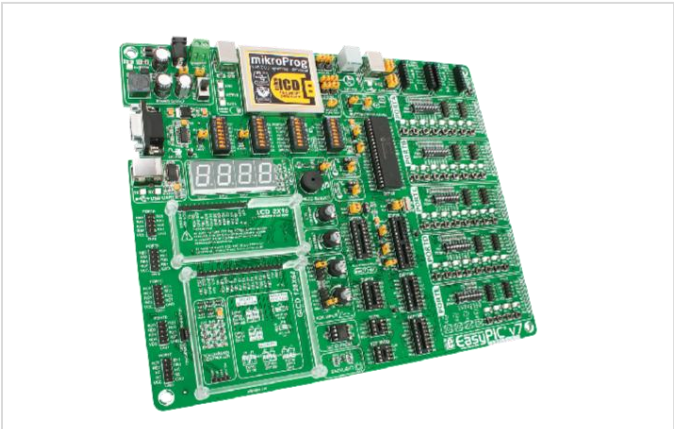

Figure 2. MikroElektronika EasyPic v7 Development system.

To implement (3) the resolution of the GLCD has been considered. For the lower left vertex we consider the coordinate (0, 64), for the second (64, 0) and for the lower right (128, 64).

The configuration for the Pic was to select the internal oscillator with the PLL enabled at 64 MHz. A function was included in the program to indicate with sound at 800Hz. The

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

program used is presented in the appendix and the executi time was approximately 1 minute.

The random selection for each of the vertices was throug the rand() command and dividing by 10000. The modf command was employed to obtain its whole number compone to obtain the numbers 1, 2 and 3 in the if statement. It is important to note that the functions rand() and modf are contained in a set standard ANSI C library functions.

The number of iterations used was 200,000. In figure 3 we show an implementation of the Sierpinski Triangle by means of a right rectangle. For this, the upper vertex has the coordina (128,0) in GLCD.

Figure 3. Variant right triangle used in the simulation

Figure 4 shows the classic simulation figure of the Sierpinski equilateral triangle in the GLCD. The coordinate of the top vertex is (64,0).

Figure 4 Sierpinski triangle

Years before, the Mandelbrot set has been programmed in in Borland C [6], Visual basic [7], Java [17], among others. If we program the Mandelbrot set algorithm in Pic microcontroller using Mikro C, we obtain its graph in the GLCD as see in figure. 5. The Mandelbrot set is built iterating equation (2).

Figure 5 Mandelbrot set.

V1. CONCLUDING REMARKS

In this work, as an example, the implementation of the Sierpinski triangle algorithm was presented in a system based on the Pic18f45k22 microcontroller. It is possible to implement other fractals such as Bransley's fern, and Julia sets, among others. However, it is possible to optimize the procedures using fixed-point in programming [18], a random function declared by the user [19] with some probability distribution [20].

In this paper we chose to implement the Sierpinski Triangle Algorithm because it is easy to implement since the formulas are arithmetic for basic programming class and it is illustrative for students who are beginning to program using iterative and conditional statements. We hope that students will be interested in the study of science and engineering.

ACKNOWLEDGMENTS

Authors would like to thank Roberto Ruiz Gomez for his contribution to this project. The authors are grateful to the anonymous referee for a careful checking of details and helpful comments that improved this paper.

APPENDIX

Code in MikroC of the Sierpinski Triangle

```
// Glcd module connections
char GLCD_DataPort at PORTD;

sbit GLCD_CS1 at RB0_bit;
sbit GLCD_CS2 at RB1_bit;
sbit GLCD_RS at RB2_bit;
sbit GLCD_RW at RB3_bit;
sbit GLCD_EN at RB4_bit;
sbit GLCD_RST at RB5_bit;

sbit GLCD_CS1_Direction at TRISB0_bit;
sbit GLCD_ CS2_Direction at TRISB1_bit;
sbit GLCD_RS_Direction at TRISB2_bit;
sbit GLCD_RW_Direction at TRISB3_bit;
sbit GLCD_EN_Direction at TRISB4_bit;
sbit GLCD_RST_Direction at TRISB5_bit;
// End Glcd module connections
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 01, January-2022

```
void Tone1()
  {
    LATE.F0=0;
    //  Frec = 800Hz, time = 2000ms
    Sound_Play(800, 2000);
  }

unsigned long i,ii, x=0,y=0;
double dd, doub;

void main()
  {
    //16MHz Internal Oscillator Frequency Select bits
    OSCCON.IRCF0=1;
    OSCCON.IRCF1=1;
    OSCCON.IRCF2=1;

    /* IRCF<  2,1,0>: Internal RC Oscillator Frequency Select
       bits:
       111 = HFINTOSC          ± (16MHz)
       110 = HFINTOSC/2        ± (8MHz)
       101 = HFINTOSC/4        ± (4MHz)
       100 = HFINTOSC/8        ± (2MHz)
       011 = HFINTOSC/16       ± (1MHz default output frequency of
                                 HFINTOSC on Reset)   */
    //PLL Enable or Disable
    // 0   --  Off   , 1 ²On,   16MHz x 4=64  MHz
    PLLEN_bit = 0;

    //    Set up a digital ports              in pic18f45k22
    // Configure AN pins as digital I/O
    ANSELA =0;
    ANSELB =0;
    ANSELC =0;
    ANSELD =0;
    ANSELE =0;

    TRISE.F0=0;
    TRISC.F2=0;
    Sound_Init(&PORTC, 2);

    //  Initialize GLCD
    Glcd_Init();
    Glcd_Fill(0x00);

  while(1)
    {
      Tone1();
      x=64;
      y=32;
      ii=0;
      dd=0;
      doub=0;
      delay_ms(200);

      for(ii=0;ii<200000;ii++)
        {
          LATE.F0=1;
          doub = modf(rand()/10000, &dd);
          if(dd==0)                //Top Vertex
            {
              x=(x+64)/2;
              y=y/2;
            }
          else if(dd==1)           //  Lower vertex right
            {
              x=(x+128)/2;
              y=(y+64)/2;
            }
          else if(dd==2)           //Lower vertex left
            {
              x=x/2;
              y=(y+64)/2;
            }
          Glcd_Dot( x, y, 1);
          if(i%1000==0)
          delay_ms  (40);
        }
      } // End    while
} //end main
```

REFERENCES

[1] Benoit B. Mandelbrot, The fractal Geometry of natur, W. H. Freeman and Co., San Francisco, 1977.

[2] A. David Wunsch, P. González-Sancho, and S. Réguley, Variable compleja con aplicaciones, Pearson educación, 1997.

[3] C. Monroy-Olivares. Curvas fractales, AlfaOmega, 2002.

[4] M. F Barnsley, Fractals everywhere. Academic press, 2014.

[5] S. Pérez-Becker, . "Compresión Fractal de Imágenes." Foro-Red-Mat: Revista electrónica de contenido matemático 28.2 (2012):

[6] H. Shildt, Turbo C/C++ 3.1 Manual de referencia primera edicion, Mcgraw Hill, 1994.

[7] Microsoft Press, Visual basic 6.0 Manual del programador, McGraw Hill, 1999.

[8] S. Nakamura, Análisis numérico y visualización con MATLAB, 1997.

[9] W. Estrada-García, Una propuesta didáctica para introducir los conceptos básicos de geometría fractal en niveles de preparatoria, Instituto Tecnologico y de Estudios Superiores de Monterrey Campus Virtual, Mexico D.F., 1999.

[10] P. C. Moreno, Un Sierpinski en la fachada. Revista Épsilon, 96, 45-60. (2017).

[11] D. A. Giraldo-Duque, Uso de la herramienta Geoebra y su influencia en la construccion del Triangulo de Sierpinski en estudiantes 8° del Instituto Tecnico Industrial Pascual Bravo, Medellin 2016, 2017.

[12] F. Díaz-Barriga, and G. Hernández-Rojas "Estrategias docentes para un aprendizaje significativo. Una interpretación constructivista, McGraw Hill, 2002.

[13] I, Dogan. Advanced PIC microcontroller projects in C: from USB to RTOS with the PIC 18F Series. Newnes, 2011.

[14] Microchip, PIC18(L)F2X/4XK22 Data Sheet, Microchip Technology Inc., 2010.

[15] D. Soto, K. Gomez, H. Silva, and F. Cordero, Exclusión, cotidiano e identidad: una problemática fundamental del aprendizaje de la matemática, Comité Latinoamericano de Matematica educativa, 2012.

[16] M.A. Sandoval-Hernandez, S. Hernandez-Mendez, S.E. Torreblanca-Bouchan, G. U. Diaz-zarango, Actualización de contenidos en el campo disciplinar de matemáticas del componente propedéutico del bachillerato tecnológico: el caso de las funciones especiales, RIDE Revista Iberoamericana para la Investigación y el Desarrollo Educativo, 12(23),34, 2021.

[17] A. Ojeda-Garcia, Programacion en Visual J++, Anaya Multimedia, 1997.

[18] O. Schlösser, Oliver. "Implementing a C++ Fixed-Point Class for Embedded Systems."

[19] L.G. Astaiza, Los números aleatorios y la ingeniería. Ingeniería e Investigación, (7), 55-60, 1983.

[20] M.A. Sandoval-Hernandez, H. Vazquez-Leal, U. A. Filobello-Nino, L. Hernandez-Martinez. New handy and accurate approximation for the Gaussian integrals with applications to science and engineering. Open Mathematics, 17(1), 1774-1793, 2019.