Reporte de Practica
Generación de diagramas de maquina de estados finita

Raul Alejandro Lopez Balleza*
*Ingeniería en Tecnologías de la Información
Universidad Politécnica de Victoria

***Resumen*—Una máquina de estado finito, es una abstracción matemática que nos permite modelar la computación[1]. El producto del desarrollo de este proyecto nos permitira generar los diagramas de MEF automaticamente mediante la entrada de datos desde un archivo.**

I. Introducción

El desarrollo de esta practica tiene como proposito el poder generar diagramas de Maquina de estados finitos a partir de datos proporcionados, que ayudaran a modelar los cambios en el sistema

II. Desarrollo Experimental

En esta practica se desarrolló un programa para modelar un diagrama de Maquina de Estados Finito. Para esto se investigaron fuentes de datos que permitieran dar una idea de como realizar tal tarea. Tales como la libreria de PyOpenGL[2], ya que la practica se realizó en el lenguaje de programación Python, la implementación de curvas de bezier en OpenGL y la lectura de archivos csv para extraer los datos a modelar. Las curvas de bezier fueron lo que mas se acoplaba para el dibujado que se queria visualizar, al principio se trató de hacer las relaciones del diagrama mediante circulo, pero no siempre se adaptaba correctamente. En la figura 1, se muestra una curva de bezier, la curva de bezier se grafica gracias a un punto de inicio, un punto de fin y al menos un punto intermedio entre esos dos que es donde se tomara la forma de curva.[3] Para la lectura de archivos csv se uso una libreria de python llamada CSV que nos permitira leer el archivo para poder extraer los datos a modelar, el nombre del archivo se recibe como primer parametro a la hora de ejecutar el script de python, en el momento de compilacion se verifica quee se haya insertado el parametro y que tenga extensión ".csv", de ésta forma el programa puede continuar de forma correcta, de lo contrario el programa termina.[4] El archivo se lee linea por linea y se parsea cada vez que encuentra una coma, asi cada valor se guarda en un indice de fila del arreglo de datos en el programa.

Para el desarrollo , utilizamos una computadora con un procesador AMD-FX7500, a 2.1 GHz, con 2 GB de memoria RAM. La PC donde se hizo la prueba contiene una tarjeta GPU Integrada Radeon R7 de 4 núcleos/6 Hilos y 1 GB de memoria de video.

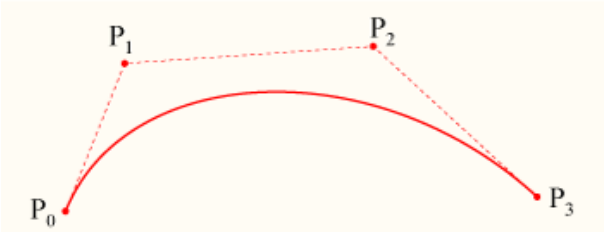

Figura 1: Curva de Bezier

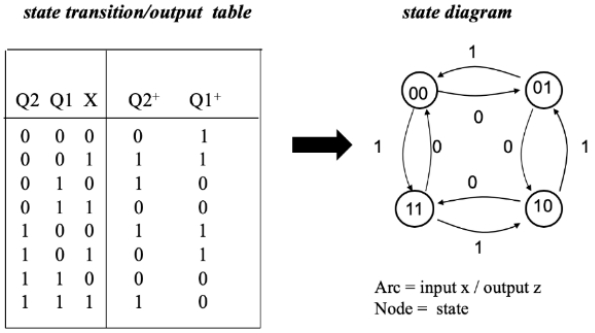

Figura 2: Problema a resolver

III. Resultados

Lo que se esperaba del resultado del diagrama de Maquina de estados es algo parecido a la Figura 2, en base a los datos de entrada desde un archivo .csv se generarian las relaciiones y los nodos del diagrama MFE. El proceso del programa es simple, primero valida el archivo ingresado, que sea un archivo .csv, de esa forma el programa sigue ejecutandose, sino, el programa termina. Despues va leyendo el archivo csv linea por linea, para cada linea obtenida se hace un parseo, se divide la linea en 3 partes, la parte de estado inicial, la parte de dato de entrada y la parte de estado final. Una vez obtenidos estos datos, se guardan en un arreglo y se repite hasta que se lea la ultima linea. Nota: Es importante que el archio proporcionado cumpla con el formato requerido (Vease tabla en la Figura 2)para su correcta lectura, de otra forma el diagrama generado no cumplira con su proposito. Posteriormente se crean los nodos de estado, obteniendo los datos del arreglo donde tenemos la informacion extraida del

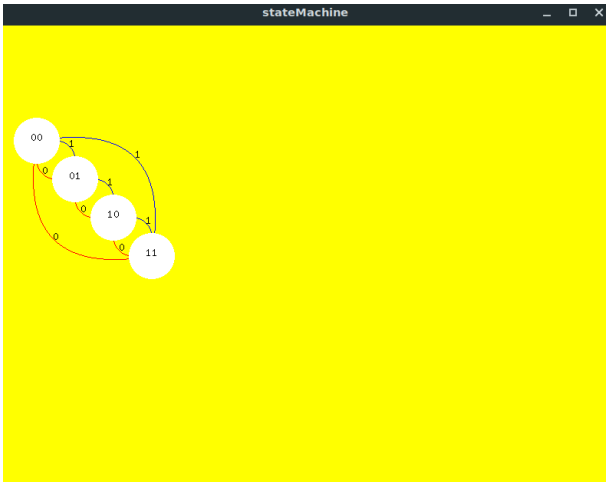

Figura 3: Salida

archivo csv. Para saber cuales son los nodos se hace un parseo de los datos, obteniendo las n posiciones antes de la mitad del arreglo, estos serian los nodos iniciales, ya que la mitad del arreglo se referiria al indice del dato de entrada y el resto a los nodos de salida. Se crean los nodos dandoles los valores de entrada y su posicion en (x,y).

Una vez generados todos los nodos se pasa a generar las relaciones. Para esto necesitamos que todos los nodos hayan sido generados con anterioridad, recorreremos el arreglo de datos nuevamente para obtener que nodos estan relacionados entre si y que dato de entrada se le ah dado a esa transicion, se comparan los datos de los nodos inicial y final con cada fila del arreglo de datos para saber si existe relacion, si la hay, se crea la relacion que recibira como parametros el par de nodos relacionados y el dato de entrada correspondiente, asi hasta acabar con todas las filas en el arreglo de datos. Se calcula el la distancia entre los nodos datos y el punto medio, la distancia nos servira de referencia para sacar el radio que tendra nuestra curva y el punto medio es donde estara nuestro punto de referencia para la curva. Segun el valor sea 1 o 0 se dibujara la linea correspondiente, si es 1 la linea se pintara de azul e ira hacia arriba(del nodo mas abajo hacia el nodo mas arriba) y si es 0 la linea se pintara de rojo e ira hacia abajo(del nodo mas arriba a el nodo mas abajo).

Despues de generar los nodos y las relaciones, pasamos a dibujar ambos, primero los nodos. Los nodos seran una circunferencia rellenada de color blanco con los datos correspondientes a cada nodo en medio, se dibuja el nodo blanco y se le ponen los datos como texto. Una vez dibujados los nodos pasamos con las relaciones que seran una curva de bezier entre 3 puntos, el nodo inicial, el nodo final y el punto itermedio, junto con su dato de entrada como texto. Estas se dibujan con la funcion glMap de OpenGL que recibe el arreglo de puntos por donde debe pasar En la Figura 3 se ve el resultado que se obtuvo, con los nodos generados con los datos proporcionados y las relaciones entre cada uno de ellos.

IV. Conclusión

Conclusion, el proposito de la practica fue cumplido, sin embargo aun veo oportunidades de mejora en cuanto al diseño del MEF, la solucion implemento lo visto en clases e investigacion de otros temas como las curvas de bezier, lectura de archivos e investigacion de como se debe modelar una MEF, se busco en los libros proporcionados por el profesor, se apoyo en articulos de internet y videos explicativos, fue probada en computadoras de mis compañeros de clase y fue ejectuado sin errores.

Referencias

[1] *Finite State Machine*. hftp://ftp.unicauca.edu.co/Facultades/FIET/DEIC/Materias/SEDS/Material%20Auxiliar/FSM.pdf. Consultado el 03-10-2019.

[2] *Computer graphics trough OpenGL - From Theory to Experiments*. http://www.sumantaguha.com/. Consultado el 03-10-2019.

[3] Unknown. *Curves in OpenGL - Evaluators*. http://www.cs.mun.ca/av/old/teaching/cg/notes/glcurves.pdf. Consultado el 03-10-2019.

[4] *CSV Library Python*. https://docs.python.org/3/library/csv.html. Consultado el 03-10-2019.