GEOWALL- E

Mauricio Sunde Jiménez C111
Richard Alejandro Matos Arderí C111
Facultad de Matemática y Computación, Universidad de La Habana

2023

Resumen

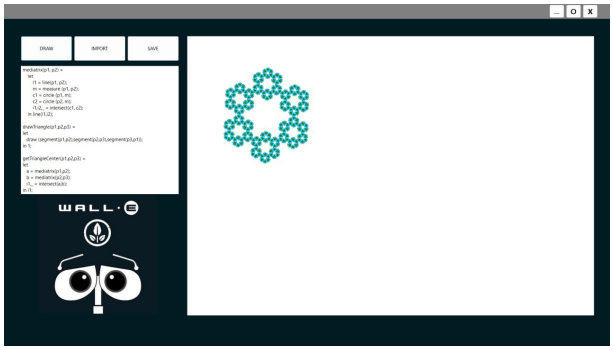
GEOWALL- E es una aplicación capaz de representar conceptos geométricos, como puntos, líneas o circunferencias, graficarlos y comprobar que relaciones básicas conocidas de la geometía se cumplen.

Índice

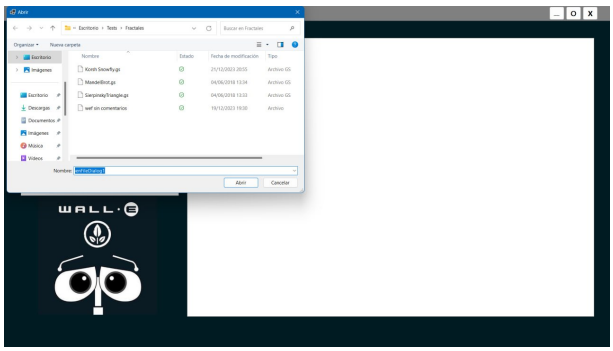
1. Introducción

2. ¿Cómo usarlo?

Una vez ejecutado el proyecto desde la carpeta raíz, el usuario será direccionado a la interfaz gráfica diseñada sobre la plataforma Windows Form. En la interfaz encontrará un cuadro de texto capaz de recepcionar el código de G # , y una vez presionado el botón ´DRAW´ aparecerá en pantalla, específicamente en la región del lienzo la representación geométrica (en caso de que el código la tuviese) , de lo introducido por el usuario. Ejemplo:



Otras funcionalidades están sobre los botones ´IMPORT´ y ´SAVE´. La ejecución del primero se basa en llamar al Administradorde archivos predeterminado del sistema , para seleccionar un archivo que contenga un código en G # que se desea representar. Una vez seleccionado el texto que contiene aparecerá automáticamente en el cuadro de texto de la interfaz , listo para ser procesado. El botón ´SAVE´ de forma análoga al anterior llama al Administradorde archivos predeterminado del sistema , pero con el objetivo opuesto, en lugar de importar información , se busca guardar en un archivo compatible, el código que se encuentra sobre el cuadro de texto en el momento de la ejecución de esta funcionalidad. Ejemplo:



De forma general solo serán representadas en el lienzo instrucciones que constituyan declaraciones de tipo draw válidas.

3. Estructura del proyecto

3.1. Interfaz Gráfica

La interfaz gráfica está implementada sobre Windows Form.

El formulario principal es construido cuando se ejecuta el proyecto (el flujo de ejcución comienza con el método Main de la clase Program, aquí se inicializa el formulario cuyo código funcional recae en la clase GEOWALL- E). Esta clase contiene los métodos que responden ante el uso de los botones implementados. El principal sería el método DRAWCLICK, quien dirige todos los procesos de procesamiento de la entrada del usuario (dígase parsing, chequeo de tipos y evaluación). A cotinuación el siguiente listado ilustra el código del método:

```
private void DRAWClick(object sender, EventArgs h)
{
            Papel.Clear(BackColor);
            Papel.Clear(ForeColor);
            string Entrada = PANELCOMANDOS.Text;
            if (Entrada != string.Empty)
            {
                try
                {
                    StandardLibrary.Initialize();
                    WallEColors.InitializeColor();
                    var statements = StatementsTree.Create(Entrada);
                    TypeChecker.CheckType(statements);
                    GSharpEvaluator evaluator = new(statements);
                    var result = evaluator.Evaluate();
                }
                catch (Exception e)
                {
                    MessageBox.Show(e.Message);
                }
            }
 }
```

3.1.1. Uso de System.Drawing: Clase graphics

La clase Graphics en C# es parte del espacio de nombres System.Drawing y proporciona funcionalidades para dibujar gráficos en una superficie, como un formulario o un control. Se utiliza para realizar operaciones de dibujo, como trazar líneas, dibujar formas, texto, imágenes, etc. Proporciona métodos para dibujar y rellenar formas geométricas, trabajar con colores, fuentes, transformaciones, entre otros que fueron empleados para cubrir estas funcionalidades en GEOWALLE-E. A continuación se listan los métodos principales que fueron empleados:

- DrawEllipse: Este método dibuja una elipse delimitada por un rectángulo especificado. Se utiliza para dibujar el contorno de una elipse. Fue adpatado para dibujar circunferencias, como caso particular de una elipse delimitada por un cuadrado cuya longitud de lado coincide con el diámetro de la circunferencia en cuestión.

- DrawArc: Dibuja un arco delimitado por un rectángulo especificado. Puede ser utilizado para dibujar una porción de una elipse o círculo. Fue utilizado para dibujar el objeto Arc ,propio de la biblioteca GSharpProject, que representa un arco de circunferencia.

- FillEllipse: Rellena el interior de una elipse delimitada por un rectángulo especificado con un color sólido. Fue empleado para representar en el lienzo al objeto Point, propio de la biblioteca GSharpProject, que representa un punto.

- DrawLine: Dibuja una línea entre dos puntos especificados. Este método fue empleado para representar los objetos de la biblioteca GSharpProject : Line, Segment y Ray que represenntan respectivamente una línea, un segmento y un rayo(entiéndase por esto una recta con un extremo).

3.1.2. Diseño

El diseño recae en el archivo archivo ´Form1.Designer.cs´ que es generado por Visual Studio para definir y configurar los controles y componentes de un formulario de Windows Forms.

Algunos aspectos clave son:

- *Configuración de Controles*: Establece las propiedades y disposiciones de los controles visuales en el formulario, como botones, etiquetas, paneles y cuadros de texto. Por ejemplo, se agregan controles como 'PANEL_DIBUJO', 'DRAW', 'IMPORT', 'CLOSE_BTN', 'MIN_BTN', 'MAX_BTN', entre otros, y se definen algunas de sus propiedades como color, ubicación y tamaño.

- *Configuración de Formulario*: Define las propiedades y comportamiento del formulario en sí, como su tamaño, color de fondo, nombre, eventos asociados, etc. Esto incluye el manejo de eventos como 'Load' y 'MouseDown'.

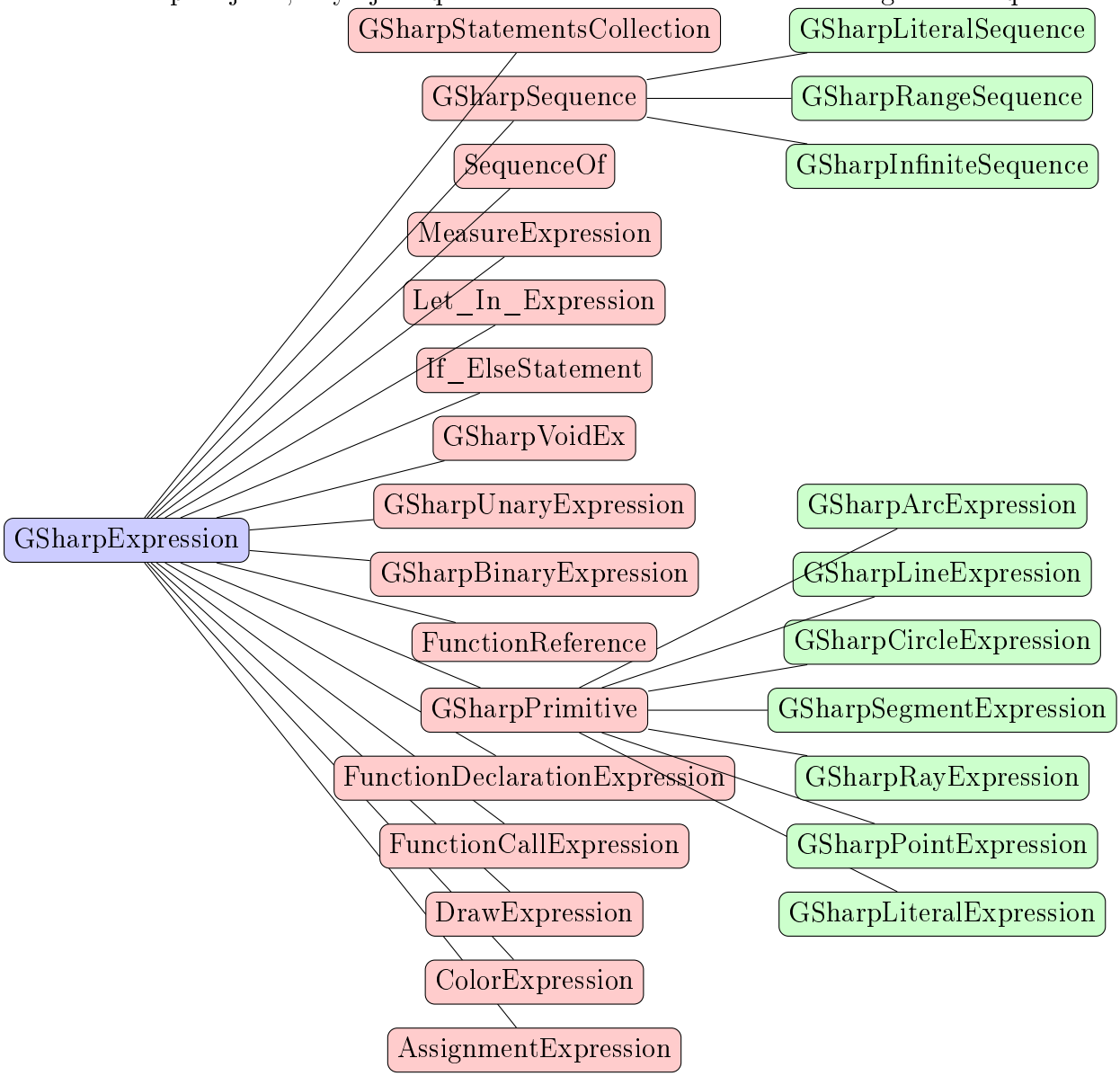
- *Ajustes de Visualización*: El código incluye ajustes de diseño y visualización, como el anclaje y el escalado de los controles para que se adapten y se muestren correctamente en el formulario.

3.2. Intérprete de G#

El intérprete de G# implementado responde al procesamiento de una cadena de texto apoyado en tres fases principales: parsing(análisis sintáctico) , chequeo de tipos y evaluación.

3.2.1. Parsing

El método Parse es llamado desde el método de tipo GharpStatementsCollection, Create perteneciente a la clase StatementsTree , invocado en DRAWClick. Con esta llamada se tokeniza la cadena de caracteres (string) introducida por el usuario, esta tokenización se realiza en el constructor de la clase Parser y se apoya en la clase Lexer y us métodos de reconocimiento de tokens. El parsing se basa en construir objetos del árbol de expresiones de GSharpProject , cuya jerarquía de herencia es ilustrada en el siguiente esquema:



3.2.2. Chequeo de tipos

Cada una de las clases del árbol de expresiones contiene el método void CheckType, pues la clase abstracta GSharpExpression lo contiene como método abstracta. De esto se deduce que cada nodo sepa c̈hequear su tipo ¨ En caso de alguna inconsistencia con las condiciones de validez de una expresión determinada , la ejecución de este método ,llamado desde el método DRAWClick y aplicado sobre cada una de las expresiones resultado del parsing , nos alertará del error y se detendrá el procesamiento de la entrada actual. Como asistente esencial de este proceso está la clase *static* TypeCheckerHelper que contiene un diccionario binaryTypesDic que contiene los patrones de tipo válidos para expresiones de G#.

3.2.3. Evaluación: Clase GSharpEvaluator

La evaluación en GEOWALL-E se realiza gracias a la clase GSharpEvaluator que recibe un nodo GSharpStatementsCollection ya una vez chequeado su tipo. Cuando en la clase GEOWALL_E se instancia este objeto , seguido se llama al método Evaluate que por cada declaración en la propiedad _Nodo la clasifica y de acuerdo a su tipo la redirecciona al método de evaluación correspondiente. De manera particular si se trata de una expresión DrawExpression son es llamado el método void Draw de la clase GEOWALL-E encargado de representar el objeto dibujable en el lienzo, de lo contrario los métodos se restringen a retornar el valor de retorno (en caso de existir) de la expresión evaluada.

3.2.4. Geometría

Los objetos geométricos implementados en la biblioteca GSharpProject implementan la interfaz IFigure, cuyo código se representa en el siguiente listado:

```
1  public interface IFigure
2  {
3          bool PointBelong(Point p1);
4          IEnumerable<Point> PointsOf();
5  }
```

Esta interfaz nos permite aprovechar las funcionalidades de saber si un punto pertenece o no a un objeto geométrico determinado y obtener un `IEnumerable<Point>` que nos brinda la posibilidad de acceder a inifinitos puntos de un objeto. Los objetos geométricos implementados son:

- Point: Punto determinado por un par ordenado de valores double.
- Line: Recta determinada por dos puntos.
- Segment: Segmento determinado por dos puntos.
- Ray: Línea determinada por un punto de inicio y otro punto que marca la dirección sobre la que la línea es infinita.
- Arc: Arco determinado por un punto centro, una medida de radio, y dos puntos (los rayos determinados por cada uno y el centro que determinan el ángulo que cubre el arco).

- Circle: Circunferencia determinada por un punto centro y una medida de su radio.

Sobre conceptos geométricos es esencial también la clase Utiles, que contiene métodos static que brindan funciones de la Geometría Analítica tales como la Distancia Euclidiana entre dos puntos y el conjunto intersección de dos lugares geométricos.

4. Conclusiones

La creación de este proyecto ha sido sumamente instructiva sobre temas como encapsulamiento, modularización, herencia, y polimorfismo por lo que ha contribuido sólidamente a fortalecer habilidades como la investigación y otras menos interactivas pero igualmente importantes como son la organización y la planificación, tan necesarias para la vida profesional.