GeoWall-E (Proyecto de Pogramación III)

Laura Alonso Rivero C-113
Salma Fonseca Curbelo C-113

Abstract

¿Qué es GeoWall-E?

Es un programa con el cual el usuario será capaz de representar conceptos geométricos (como puntos, líneas o circunferencias), graficarlos y comprobar relaciones que se cumplen (por ejemplo, que las mediatrices en un triángulo se cortan en el mismo punto y que es centro de la circunferencia que lo circunscribe). Para ello se cuenta con un lienzo (plano), una regla (para trazar líneas, rectas y semirrectas) y un compás (para trazar circunferencias y arcos).

El proyecto se divide en dos partes fundamentales:

- El compilador o parte lógica.
- La interfaz gráfica.

El compilador a su vez se divide en tres fases fundamentales:

- Lectura de la expresión introducida por el usuario. De este proceso se encarga el **Lexer** el cual escanea la entrada y devuelve una lista de **Tokens**.
- Creación del Árbol de Sintaxis Abstracta, AST por sus siglas en inglés. Esta parte corre por el **Parser** que devuelve una lista donde se clasifican las diferentes expresiones.
- La interpretación de la expresión. Llevada a cabo por el **Interpreter** que va a devolver la evaluación de la expresión entrada por el usuario.

Lectura de la expresión (Ánalisis Léxico)

Cuando el usuario introduce la línea comienza la lectura de la expresión, con este propósito se creó la clase **Lexer**. Esta clase consta de un método principal llamado **Scan** el cual recorre cada caracter de la entrada del usuario y dependiendo de sus valores se irán clasificando. Esta clasificación está dada debido al tipo de **Token** que sea.

Un **Token** es una secuencia de caracteres que posee un valor y tipo específico. Existen diversos tipos como son: las palabras claves (circle, let, if), los nombres de las variables o las funciones (x, y, fib), los operadores aritméticos, de relación y lógicos (*, ==, &), las funciones matemáticas (cos, sqrt, log) y finalmente los literales numéricos o cadenas de strings. Estos están agrupados en un enum nombrado **TypesOfToken**.

Una vez escaneada toda la entrada se devolverá una lista con todos estos tokens en la posición en que fueron encontrados para su posterior procesamiento.

Creación del AST (Ánalisis Sintáctico)

A partir de la lista de Tokens retornada se procede a crear el Árbol Sintáctico el cual no es más que una forma de definir los pasos a seguir por el Compilador a la hora de evaluar las expresiones de una forma recursiva. En este caso se uso el método de parsing recursivo descendente, el cual consiste en construir el árbol a partir de las reglas de gramática de cada lenguaje. Esta grámatica se encuentra implementada en la clase **Parser**. Con este método de parsing se comienza a partir del símbolo inicial (raíz) y se va descendiendo en el AST hacia las hojas. En cada caso se elige la regla gramatical adecuada y se invoca la función correspondiente para continuar el análisis.

Para lograr la generación del árbol fue necesario crear las clases abstractas **Expressions** y **Stmt** para poder identificar los distintos tipos de entradas que el usuario era capaz de implementar y agregarlas a una lista. Un statement sería la declaración que modificará al programa por ejemplo la declaración de una función, la asiganación de variables o la declaración de una nueva figura a dibujar. Por otro lado las expesiones son valores o elemnetos que acompañaran a estas declaraciones.

Interpretación de la expresión (Ánalisis Semántico)

En esta etapa se interpreta, evalúa y finalmente se devuelve el resultado de la entrada del usuario a través de la clase **Interpreter**. El objetivo es verificar que las operaciones sean realizadas con los operandos correctos.

En algunas expresiones como la declaración de variables y funciones fue necesaria la creación de un **Scope global**. Esto no es más que el conjunto de variables o funciones que son accesibles desde cualquier parte del programa. Es muy importante para evitar conflictos y ambigüedades en el código pues una vez declarada una función o variable no es posible declarar una con el mismo nombre en el mismo ámbito y con las mismas características

Para interpretar las expresiones se invoca el método **Evaluate** el cual como su nombre indica realiza la evaluación de los distintos tipos de expresiones guardados en la lista. Como resultado de esta evaluación se tendrán las distintas figuras a dibujar en la interfaz gráfica con el color requerido por el usuario.

Una vez evaluada la entrada el resultado se imprimirá en la consola.

Errores

Durante el proceso de compilación pueden ocurrir tres errores diferentes:

Error Léxico: se presenta cuando la entrada contiene caracteres no válidos en el lenguaje.

Error Sintáctico: se presenta cuando la secuencia de tokens no es válida atendiendo a la gramática del lenguaje o cuando se trata de sobreescribir una variable o función en un mismo contexto.

Error Semántico: se presenta cuando no se puede evaluar la expresión debido a incoherencias entre la operación y los tipos de expresiones que se encuentra en ella.

Parte Gráfica:

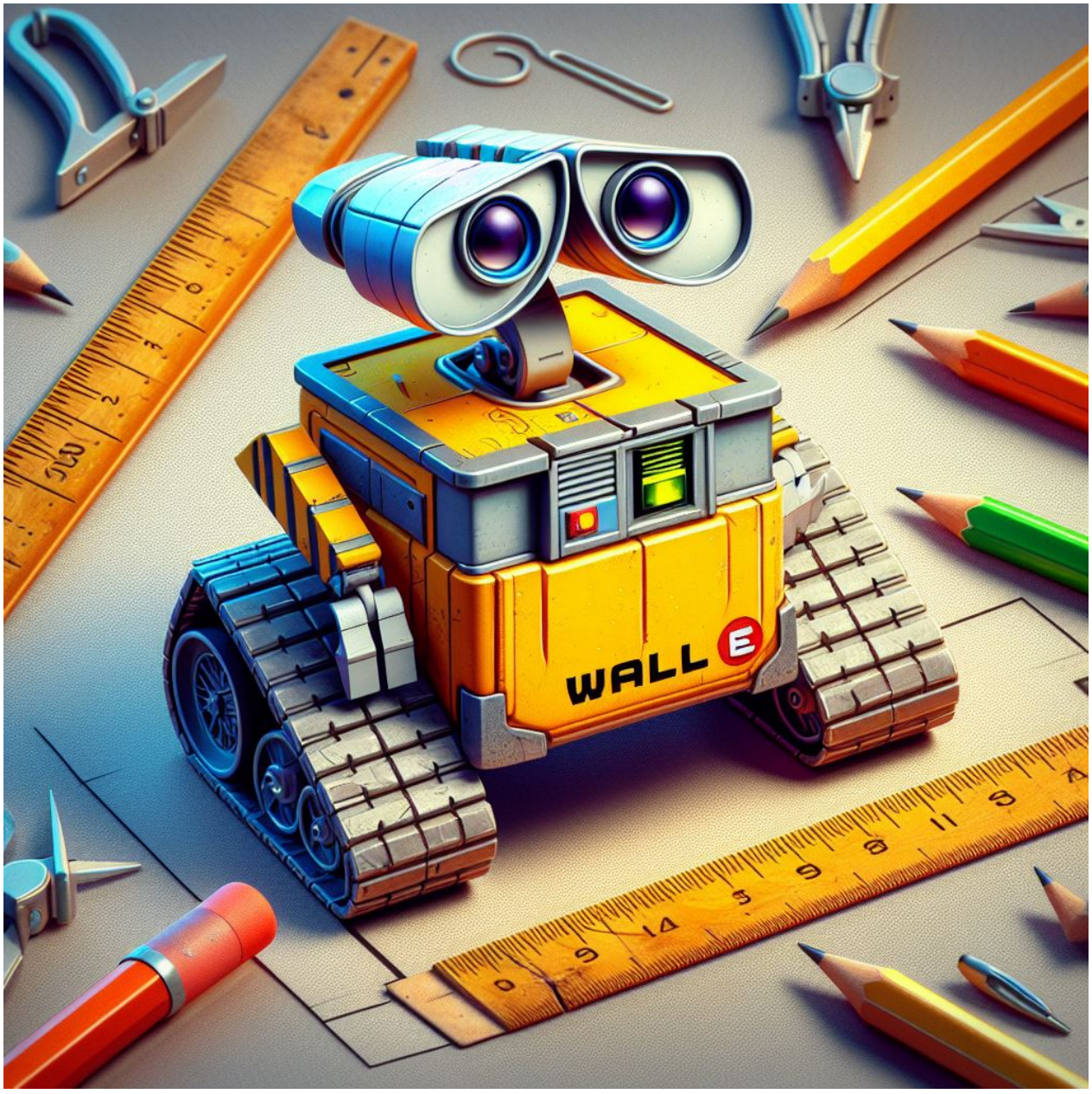
Se inicia un formulario compuesto por:

- Un textBox inputTextBox, donde el usuario puede introducir su código.
- Un botón Compilebutton, que al darle click compila el texto del usuario, de tener algún error se muestra en la pantalla y si está todo correcto activa el botón Draw y borra si había mostrado algún error anteriormente. Crea una nueva instancia de Compile y la lista que se obtiene como resultado, se iguale a la lista de IDrawer que se tiene como propiedad.
- Un botón runButton, que al darle click le da valor de true a isDrawing (isDrawing es una propiedad tipo bool que se inicializa en false, para poder realizar el dibujo tiene que cambiarse a true). Luego activa el botón StopDrawing, se comienza el método para dibujar y vuelve a ponerse en falso lo que se había cambiado anteriormente.
- Un botón stopDrawing, que le da valor false a isDrawing y desactiva el botón stopDrawing.
- Dos labels, uno para mostrar los errores y otro encima de inputTextBox, que indica al usuario que escriba su código ahí.
- Un PictureBox drawingBox, donde se muestra el dibujo.

El método de dibujar recorre la lista de Idrawers con un foreach, que mientras que isDrawing esté en true hace lo siguiente:

1. Convierte el color que tiene como propiedad el Idrawer a tipo Color con la función SearchColor.
2. Si la propiedad Phrase no es nula se imprime.
3. Hace un Switch case que en dependencia del tipo del IDrawer, se llama al método que lo dibuja.

Con el correcto funcionamiento de este programa el usuario será capaz de representar gráficamente conceptos geométricos a través de los comandos necesarios para su realización.


WALL E