GeoWall-E Proyecto de

Pogramación III

Ernesto González Vargas C-112

Ronal Prieto Vázquez C-112

Resumen

¿Qué es GeoWall-E?

Se trata de un software que permite al usuario representar conceptos geométricos (como puntos, líneas o círculos), dibujarlos y verificar propiedades que se cumplen (por ejemplo, que las bisectrices de un triángulo se intersectan en un punto que es el centro del círculo que lo circunscribe). Para esto, dispone de un plano, una regla para trazar líneas, segmentos y semirrectas y un compás para trazar círculos 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:

El **Lexer** se encarga de leer la expresión que introduce el usuario. Este proceso consiste en analizar la entrada y generar una lista de **tokens**.

El **Parser** se ocupa de crear el Árbol de Sintaxis Abstracta, **AST** por sus siglas en inglés. Este componente devuelve una lista que categoriza las distintas expresiones.

El **Interpreter** se responsabiliza de interpretar la expresión. Este elemento produce la evaluación de la expresión que introduce el usuario.

1 Lectura de la expresión (Análisis Léxico)

La clase **Lexer** se encarga de leer la expresión que introduce el usuario. Este proceso consiste en analizar la entrada y generar una lista de tokens. Un **token** es una secuencia de caracteres que tiene un valor y un tipo específico. Hay varios tipos de tokens, como las palabras clave (circle, let, if), los nombres de las variables o las funciones (x, y, fib), los operadores aritméticos, relacionales y lógicos

(*, ==, &), las funciones matemáticas (cos, sqrt, log) y los literales numéricos o cadenas de caracteres. Estos tipos se agrupan en un enum llamado **SYNTAXKIND**. Después de escanear toda la entrada, se devuelve una lista con todos los tokens en el orden en que se encontraron para su análisis posterior.

2 Creación del AST (Análisis Sintáctico)

Se crea el Árbol Sintáctico a partir de la lista de Tokens devuelta. Este árbol es una forma de definir los pasos que debe seguir el Compilador para evaluar las expresiones de forma recursiva. Se usó el método de análisis sintáctico recursivo descendente, que consiste en construir el árbol según las reglas de gramática de cada lenguaje. Esta gramática está implementada en la clase **Parser**. Con este

método de análisis se empieza desde el símbolo inicial (raíz) y se va bajando en el AST hasta las hojas. En cada caso se escoge la regla gramatical adecuada y se llama a la función correspondiente para seguir el análisis.

Para generar el árbol se crearon las clases abstractas **ExpressionSyntax** y **Statements** para identificar los diferentes tipos de entradas que el usuario podía introducir y añadirlas a una lista. Un statement es una declaración que modifica al programa, por ejemplo, la declaración de una función, la asignación de variables o la declaración de una nueva figura a dibujar. Por otro lado, las expresiones son valores o elementos que acompañan a estas declaraciones.

3 Interpretación de la expresión (Análisis Semántico)

Se usa la clase **Evaluator** para interpretar, evaluar y devolver el resultado de la entrada del usuario. El propósito es comprobar que las operaciones se hagan con los operandos adecuados.

En algunas expresiones como la declaración de variables y funciones se creó un **Scope global**. Esto es el conjunto de variables o funciones que se pueden usar desde cualquier parte del programa. Es muy importante para prevenir conflictos y ambigüedades en el código, ya que no se puede declarar una función o variable con el mismo nombre en el mismo ámbito y con las mismas características.

Para interpretar las expresiones se llama al método **Evaluate**, que hace la evaluación de los diferentes tipos de expresiones guardados en la lista. Como resultado de esta evaluación se obtendrán las distintas figuras a dibujar en la interfaz gráfica con el color que el usuario quiera. Después de evaluar

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

4 ERRORES

Durante el proceso de compilación pueden ocurrir tres errores diferentes:

Error Léxico: se presenta cuando la entrada contiene caracteres no validos 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 sobrescribir 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.

5 PARTE GRÁFICA

Se crea un formulario que contiene:

Dos labels, uno para mostrar los errores y otro encima de inputTextBox, que indica al usuario que escriba su código ahí.

Un botón **Run**, que al pulsarlo le da valor de true a isDrawing (isDrawing es una propiedad de tipo bool que se inicializa en false, para poder hacer el dibujo tiene que cambiarse a true).

Un textBox inputTextBox, donde el usuario puede escribir su código.

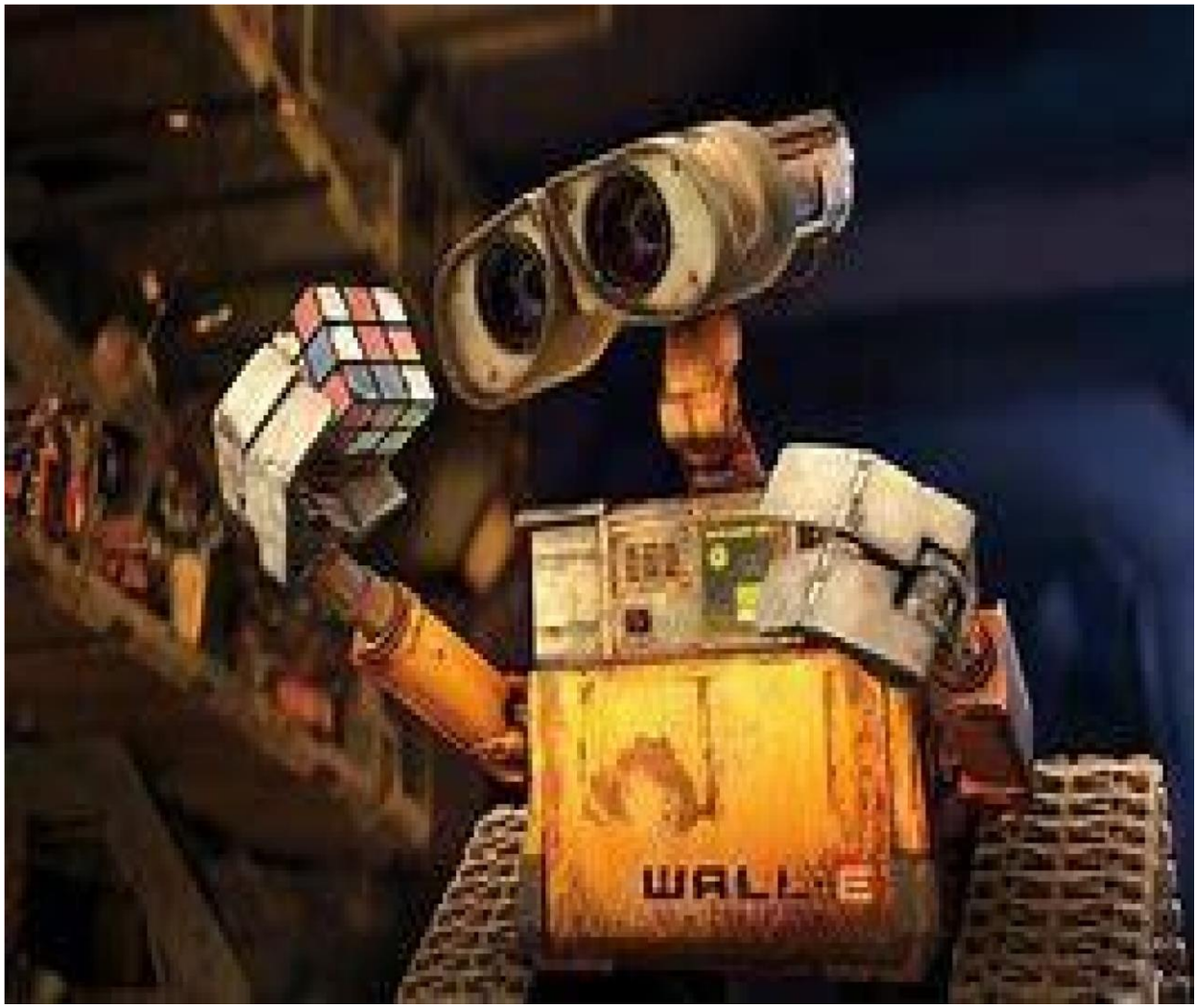
Un botón **Compile**, que al pulsarlo compila el texto del usuario, muestra los errores en la pantalla si los hay y si está todo correcto activa el botón Draw y borra los errores anteriores si los hubiera. Crea una nueva instancia de Compile y la lista que se obtiene como resultado se asigna a la lista de IDrawer que se tiene como propiedad.

Después activa el botón **Stop**, empieza el método para dibujar y vuelve a ponerse en falso lo que se había cambiado antes. Un botón **Stop**, que le da valor false a isDrawing y desactiva el botón **Stop**.

Un PictureBox **Painting**, 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:

- Convierte el color que tiene como propiedad el Idrawer a tipo Color con la función SearchColor.
- Si la propiedad Phrase no es nula se imprime.
- Hace un Switch case que dependiendo del tipo del IDrawer, llama al método que lo dibuja. Con el correcto funcionamiento de este programa el usuario podrá representar gráficamente conceptos geométricos mediante los comandos necesarios para su realización.


WALL·E