Manual Técnico

Proyecto: Animaciones
Lenguaje :Java
IDE: NetBeans8.2
Herramientas Principales: jflex, cup

Para desarrollo del programa se utilizaron varias herramientas que serán explicadas de manera mas profunda con la ayuda de este manual Tecnico.
Primero debemos de enfocarnos en las gramáticas que se utilizaron, que son cuatro cada uno con el objetivo de analizar un archivo de entrada en específico.

Gramatica Lienzo(.lnz)

N: *{Inicio, Cabecera,Lienzos,Lienzo,Componentes, Componente,Id,Nombre,Tipo,Color ,CompRGBs,RGB,NumRGB, NumHex,CompsTamaño,CompsTam,HEX:#,Red,Blue,Green}*
T: *{{,},,,:,",png,gif,LIENZOS,nombre,tipo,Fondo,tamaño}*
S: *{Inicio}*
P:

```
Inicio          ::= {Cabecera}
Cabecera        ::=  LIENZOS:{Lienzos}
Lienzos         ::=  Lienzo , Lienzos
                        | Lienzo
Lienzo          ::=  Id :{Componentes}
Componentes     ::=  Componente, Componentes
                        | Componente
Componente      ::= nombre: "Nombre"
                        |tipo :"Tipo"
                        |Fondo:{Color}
                        |tamaño:{CompsTamaño}
Id              ::=Letra|_(Letra|Numero|_)*
Nombre          ::= (Letra|Numero|Simbolo|" ")+
Tipo            ::= png
                        |gif
Color           ::= CompRGBs
                        | HEX:#NumHEX
CompRGBs        ::=RGB,CompRGBs
                        |RGB
RGB             ::= Red: NumRGB
                  | Blue: NumRGB
                  | Green: NumRGB
NumRGB          ::= Digito(Digito|"")^2
NumHEX          ::= (DigitoHEX)^6
CompsTamaño:::= CompTam,CompsTamaño
                        |CompsTam
CompsTam        ::= cuadro: Entero
```

| dimension_x: Entero
| dimension_y: Entero
DigitoHEX ::= Digito |A |B |C |D |E |F
Entero ::=(Digito)+
Simbolo ::= _|$|-
Letra ::= [Aa-Zz]
Digito ::= [0-9]

Gramática Color (,clrs)

N: {Inicio,Cabecera,Lienzos,Lienzo,Id_Colores,Id_Color,Id,Nombre,Color,CompRGBs, RGB,NumRGB,NumHex}
T: {{,},,,:,COLORES,HEX:#,Red,Blue,Green}
S: {Inicio}
P:

Inicio ::= {Cabecera}
Cabecera ::= COLORES:{Lienzos}
Lienzos ::= Lienzo , Lienzos
| Lienzo
Lienzo ::= Id :{Id_Colores}
Id_Colores ::= Id_Color, Id_Colores
| Id_Colores
Id_Color ::= Id:{Color}
Color ::= CompRGBs
| HEX:#NumHEX
Id ::=Letra|_(Letra|Numero|_)*
CompRGBs ::=RGB,CompRGBs
|RGB
RGB ::= Red: NumRGB
| Blue: NumRGB
| Green: NumRGB
NumRGB ::= Digito(Digito|"")^2
NumHEX ::= (DigitoHEX)^6
DigitoHEX ::= Digito |A |B |C |D |E |F
Simbolo ::= _|$|-
Letra ::= [Aa-Zz]
Digito ::= [0-9]

Gramatica Tiempo(,tmp)
N: {Inicio,Cabecera,Lienzos, Lienzo,Comps_Tiempo, Comp_Tiempo,Imagenes, Comps_Imagen, Comp_Imagen, Id,Nombre }
T: {{,},,,:,",TIEMPOS,duracion,id,[,]}
S: {Inicio}
P:

```
Inicio              ::= {Cabecera}
Cabecera        ::=  TIEMPOS:{Lienzos}
Lienzos          ::=  Lienzo , Lienzos
                            | Lienzo
Lienzo             ::=  Id :{Comps_Tiempo}
Comps_Tiempo     ::=  Comp_Tiempo, Comps_Tiempo
                            | Comp_Tiempo
Comp_Tiempo::= inicio: "Id"
                            |fin : "Id"
                            |imagenes:[Imagenes]
Imagenes        ::= {Comps_Imagen},Imagenes
                            |{Comps_Imagen}
Comps_Imagen ::= Comp_Imagen,Comps_Imagen
                           |Comp_Imagen
Comp_Imagen ::= id:"Id"
                            |duracion: Entero
Id                  ::=Letra|_(Letra|Numero|_)*
Entero           ::=(Digito)+
Simbolo          ::= _|$|-
Letra              ::= [Aa-Zz]
Digito             ::= [0-9]
```

Gramatica Pintar(.pnt)

N: {Inicio,Declaraciones,Declaracion, Tipo,variables, variable,Imagenes, inicializacion, Comp_Imagen, Id,Nombre,If,While, Else, Pintar }

T: {{,},,,:,",VARS,INSTRUCCIONES ,id ,[,] ,int, String,boolean,true, false, PINTAR, if,else, while, and, or ,=, ==, <, >, >=, <=, <>,+,-,/,*,..}

S: {Inicio}

P:

```
Inicio                    ::= VARS[Declaraciones]Instrucciones
Declaraciones          ::= DeclaracionDeclaraciones
                                  |Declaracion
Declaracion             ::=    Tipo variables
Tipo::=                          |int
                                  |String
                                  |boolean
variables               ::= variable ,variables
                                  |variable
variable                ::= id =inicializacion
                                  | id
inicializacion          ::= "String"
                                  | id
                                  |numero
```

```
                                |bool
numero          ::= numero + numero
                        |numero - numero
                        |numero * numero
                        |numero / numero
                        |Entero
                        |id
boo             l::= boolean
                        |boolean OpLog bool
OpLog           ::= AND
                        |OR
boolean         ::=  true
                        | false
                        |numero OpRelacional numero

Instrucciones     ::=  INSTRUCCIONES(id)[codigo] Instrucciones
                                |INSTRUCCIONES(id)[codigo]
codigo          ::=  instruccion codigo
                        |instruccion
instruccion::= id=inicializacion
                |Pintar
                |If
                |While
                |
Pintar:=PINTAR (id_Pintar,id_Pintar,num_Pintar,num_Pintar)
id_Pintar::=String
                |id
num_Pintar::=numero..numero
                |numero
If::=    if(boolean){codigo}else
else::=    else{codigo}
          |
While           ::= while(boolean){codigo}
Id              ::=Letra|_(Letra|Numero|_)*
Entero          ::=(Digito)+
Simbolo         ::= _|$|-
Letra            ::= [Aa-Zz]
Digito           ::= [0-9]
```

Para construir estas gramáticas en forma de codigo java se utilizaron las herramientas jflex y cup generando un analizador lexico y sintactico para cada una de ellas.

Tabla De Símbolos

Para el control de declaraciones de objetos como lienzos, colores, variables, etc, se utilizó una clase tabla de símbolos que consiste en una lista de objetos que van siendo insertados conforme el análisis de los lenguajes.

```
public class TablaDeSimbolos {
    private     ArrayList<Objeto> objetos;
    public static final int TIPO_LIENZO=1;
    public static final int TIPO_COLOR=2;
    public static final int TIPO_IMAGEN=3;
    public static final int TIPO_INT=4;
    public static final int TIPO_STRING=5;
    public static final int TIPO_BOOLEAN=6;
    public TablaDeSimbolos() {
        objetos= new ArrayList<>();
    }
```

Esta clase cuenta con ArrayList de objetos que serán los que identifiquen a cada una de las instancias. La clase Objeto cuenta con un Integer tipo un String Id y un Object Valor que será como lo indica su nombre el valor del objeto con el id especificado.

```
public class Objeto {
    private String id;
    private int tipo;
    private Object valor;
    private String idPert=null;

    public Objeto(String id, int tipo, Object valor) {
        this.id = id;
        this.tipo = tipo;
        this.valor = valor;
```

Manejador De Analizadores

Para el manejo de los lenguajes y de la tabla de símbolos, se encuentra la clase Manejador Analizadores que se encarga de llamar cada analizador en orden para luego irlos pasando una misma instancia de tabla a cada una, para que todos los elementos de los cuatro archivos se guarden en la misma tabla. Esa clase verifica si existen errores no recuperables de los archivos y si no es así sigue con el siguiente analizador.

```
public class ManejadorDeAnalizadores {
private boolean analizado=false;
    private TablaDeSimbolos tds;
    public void analizar(IDE ide, ManejadorDeEntrada me) {
        analizado=false;
        if (true/*me.verificarFilesExistentes()*/) {
            ide.borrarOutput();
            ide.escribirEnOutput("Empezando Analisis...\n");
            ide.escribirEnOutput("Analisis De Archivo Lienzo\n");
             tds = new TablaDeSimbolos();
            try {
                BufferedReader br = new BufferedReader(new FileReader(me.obtenerFile(0)));
                 tds = new TablaDeSimbolos();
                AnalizadorLexicoLienzo all = new AnalizadorLexicoLienzo(br);
                AnalizadorSintacticoLienzo analizadorSintacticoLienzo = new AnalizadorSintacticoLienzo(all);
                analizadorSintacticoLienzo.setFrame(ide);
                analizadorSintacticoLienzo.setTabla(tds);
                analizadorSintacticoLienzo.parse();
                if (!analizadorSintacticoLienzo.error) {
                    if (analizadorSintacticoLienzo.errorRecuperable) {
                        ide.escribirEnOutput("Existen Errores en el Archivo .lnz pero fue posible recuperarse\n\n");
                    } else {
                        ide.escribirEnOutput("Archivo .lnz Analizado Correctamente\n\n");
                    }

                    ide.escribirEnOutput("Analisis De Archivo Color\n");
                    br = new BufferedReader(new FileReader(me.obtenerFile(1)));
                    AnalizadorLexicoColores alc = new AnalizadorLexicoColores(br);
                    AnalizadorSintacticoColores asc = new AnalizadorSintacticoColores(alc);
                    asc.setFrame(ide);
                    asc.setTabla(tds);
                    asc.parse();
```

```
    asc.parse();
    if (!asc.error) {
        if (asc.errorRecuperable) {
            ide.escribirEnOutput("Existen Errores en el Archivo .clrs pero fue posible recuperarse\n");
        } else {
            ide.escribirEnOutput("Archivo .clrs Analizado Correctamente\n\n");
        }
        ide.escribirEnOutput("Analisis De Archivo Tiempo\n");
        br = new BufferedReader(new FileReader(me.obtenerFile(2)));
        AnalizadorLexicoTiempo alt = new AnalizadorLexicoTiempo(br);
        AnalizadorSintacticoTiempo ast = new AnalizadorSintacticoTiempo(alt);
        ast.setFrame(ide);
        ast.setTabla(tds);
        ast.parse();
        if (!ast.error) {
            if (ast.errorRecuperable) {
                ide.escribirEnOutput("Existen Errores en el Archivo .tmp pero fue posible recuperarse\n");
            } else {
                ide.escribirEnOutput("Archivo .tmp Analizado Correctamente\n\n");
            }
            ide.escribirEnOutput("Analisis De Archivo Pintar\n");
            br = new BufferedReader(new FileReader(me.obtenerFile(3)));
            AnalizadorLexicoPintar alp = new AnalizadorLexicoPintar(br);
            AnalizadorSintacticoPintar asp = new AnalizadorSintacticoPintar(alp);
            asp.setFrame(ide);
            asp.setTabla(tds);
            asp.parse();
            if (!ast.error) {
                if (ast.errorRecuperable) {
                    ide.escribirEnOutput("Existen Errores en el Archivo .pnt pero fue posible recuperarse\n");
                } else {
                    ide.escribirEnOutput("Archivo .pnt Analizado Correctamente\n\n");
                }
                    ide.escribirEnOutput("Listo Para Generar\n");
```

Creación De Imágenes:

Despues del analisis exitoso de los archivos, se genera un panel para cada imagen conformado por paneles mas pequenos que representan los pixeles. Y luego con las instrucciones pintar se colorean para formar el imagen, luego de eso se utiliza un BufferedImage para convertir ese panel a un imagen png asi :
BufferedImage image = new BufferedImage (imagenesAimprimirgetWidth(), imagenesAimprimir.getHeight(),BufferedImage.TYPE_INT_RGB);
Graphics2D g2 = image.createGraphics();

```
	lienzo.getTiempo().getImagenes().get(i).getImagen().paint(g2);
	try {
		ImageIO.write(image, type, new File(file.getPath() + "/" +
imagenesAimprimir.get(i).getId() + "." + type));
	} catch (Exception e) {
		e.printStackTrace();
	}
```

Si el lienzo es un gif se utilizan estos png para generar el gif segun el tiempo de cada imagen para luego ser borrados.