Cada aplicación de software debe tener su respectivo manual para documentar la creación del diseño para el desarrollo de la misma.

MANUAL TECNICO

GAME-COMPILER

Nombre: Hellen Alexandra Lacan Hernandez
Carne: 201325674
Compiladores 1.

Contenido

\

INTRODUCCION

Este manual introduce al lector a la parte técnica de la elaboración y creación del diseño. Donde encontrara un mejor entendimiento del proceso de ejecución del juego “Game-Compiler”.

Cada aplicación o software debe tener su respectivo manual para documentar la creación del diseño para el desarrollo de la misma.

Por lo tanto se estipula la función y propiedades de cada clase utilizada para la realización de dicha aplicación.

OBJETIVOS

General

- Construir una solución que permita aplicar los conceptos adquiridos sobre lenguajes formales e implementar un analizador léxico que permita programar y simular el juego Game-Compiler.

Especifico

- Documentar todo el proceso de diseño para el desarrollo de la aplicación.
- Creación de gramáticas capaz de recuperarse de errores léxicos y sintácticos.
- Comprender la utilización de la herramienta irony.

ALCANCE DEL MANUAL

- UML
- Entorno de Desarrollo
- Especificaciones
- No incluye código fuente

ENTORNO DE DESARROLLO

- Windows 8
- IDE: Visual studio 2017
- Aplicación c#.

DESCRIPCION GENERAL DE LA APLICACIÓN

Construir un juego Game-Compiler, el cual permite realizar diferentes escenarios o niveles, para su más novedoso videojuego, así también crear escenarios con el fin de lograr probar o ejecutar el escenario con los personajes y objetos configurados.

Se cargaran 2 archivos con extensión. xconf, .xesc para iniciar el juego:

- En el primer archivo, establece la información sobre las propiedades de los diferentes fondos, escenarios personajes y objetos.
- El segundo permite visualizar el escenario, posteriormente hacer un test del escenario al probar o ejecutar el escenario.

GRAMATICA ARCHIVO DE CONFIGURACION (.XCONF)

```
INICIO  = CONFIGURACION;

            CONFIGURACION  = menor + _configuration + mayor + LISTA_CONFIGURACION + menor + slash + _configuration + mayor;

            LISTA_CONFIGURACION  = LISTA_CONFIGURACION + BACKGROUND
                                 | LISTA_CONFIGURACION + FIGURE
                                 | LISTA_CONFIGURACION + DESIGN
                                 | BACKGROUND
                                 | FIGURE
                                 | DESIGN
                                 | Empty;
            LISTA_CONFIGURACION.ErrorRule = SyntaxError + ">";
            LISTA_CONFIGURACION.ErrorRule = SyntaxError + ",";
            LISTA_CONFIGURACION.ErrorRule = SyntaxError + "{";
            LISTA_CONFIGURACION.ErrorRule = SyntaxError + "}";

            BACKGROUND  = menor + _background + mayor + LISTA_BACKGROUND + menor + slash + _background + mayor;

            FIGURE  = menor + _figure + mayor + LISTA_FIGURE + menor + slash + _figure + mayor;

            DESIGN  = menor + _design + mayor + LISTA_DESIGN + menor + slash + _design + mayor;

            LISTA_BACKGROUND  = LISTA_BACKGROUND + coma + llaveAb + ATRIBUTOS_BACKGROUND + llaveCerr
                              | llaveAb + ATRIBUTOS_BACKGROUND + llaveCerr
                                    | Empty;
            LISTA_BACKGROUND.ErrorRule = SyntaxError + "}";

            LISTA_FIGURE  = LISTA_FIGURE + coma + llaveAb + ATRIBUTOS_FIGURE +llaveCerr
                          | llaveAb + ATRIBUTOS_FIGURE + llaveCerr
                          | Empty;
```

```
LISTA_FIGURE.ErrorRule = SyntaxError + "}";

LISTA_DESIGN  = LISTA_DESIGN + coma + llaveAb + ATRIBUTOS_DESIGN +llaveCerr
                    | llaveAb + ATRIBUTOS_DESIGN + llaveCerr
                    | Empty;
LISTA_DESIGN.ErrorRule = SyntaxError + "}";

ATRIBUTOS_BACKGROUND  = ATRIBUTOS_BACKGROUND + _x + guion + _nombre + igual + identificador + ptoYComa
                          | ATRIBUTOS_BACKGROUND + _x + guion + _imagen + igual + cadena + ptoYComa
                          | Empty;
ATRIBUTOS_BACKGROUND.ErrorRule = SyntaxError + ";";

ATRIBUTOS_FIGURE  = ATRIBUTOS_FIGURE + _x + guion + _nombre + igual + identificador + ptoYComa
                      |ATRIBUTOS_FIGURE + _x + guion + _imagen + igual + cadena + ptoYComa
                      |ATRIBUTOS_FIGURE + _x + guion + _vida + igual + EXPRESION + ptoYComa
                      |ATRIBUTOS_FIGURE + _x + guion + _tipo + igual + _x + guion + FIGURE_TIPO + ptoYComa
                      |ATRIBUTOS_FIGURE + _x + guion + _descripcion + igual + cadena + ptoYComa
                      |ATRIBUTOS_FIGURE + _x + guion + _destruir + igual + EXPRESION + ptoYComa
                      | Empty;
ATRIBUTOS_FIGURE.ErrorRule = SyntaxError + ";";

ATRIBUTOS_DESIGN  = ATRIBUTOS_DESIGN + _x + guion + _nombre + igual + identificador + ptoYComa
                          | ATRIBUTOS_DESIGN + _x + guion + _destruir + igual + EXPRESION + ptoYComa
                          | ATRIBUTOS_DESIGN + _x + guion + _imagen + igual + cadena + ptoYComa
                          | ATRIBUTOS_DESIGN + _x + guion + _tipo + igual + _x + guion + DESIGN_TIPO + ptoYComa
                          | ATRIBUTOS_DESIGN + _x + guion + _creditos + igual + EXPRESION + ptoYComa
                          | Empty;
ATRIBUTOS_DESIGN.ErrorRule = SyntaxError + ";";

FIGURE_TIPO  = _heroe
                 |_enemigo
                 |identificador;

DESIGN_TIPO  = _meta
                 | _bloque
                 | _bomba
                 | _arma
                 | _bonus
                 | identificador;
```

```
EXPRESION  = EXPRESION + mas + EXPRESION
           | EXPRESION + menos + EXPRESION
           | EXPRESION + por + EXPRESION
           | EXPRESION + slash + EXPRESION
           | parentAb + EXPRESION + parentCerr
           | signoMenos + EXPRESION
           | signoMas + EXPRESION
           | numero;
```

GRAMATICA ARCHIVO DE CONFIGURACION DE ESCENARIOS (.XESC)

```
INICIO   = ESCENARIOS;

            ESCENARIOS   = menor + _x + guion + _escenarios + _background + igual + identificador + ptoYcoma + _ancho + igual + EXPRESION
                         + ptoYcoma + _alto + igual + EXPRESION + mayor + CUERPO_ESCENARIO;

            CUERPO_ESCENARIO   = LISTA_ESCENARIO + FINALESCENARIO
                               |FINALESCENARIO;
            CUERPO_ESCENARIO.ErrorRule = SyntaxError + ">";

            FINALESCENARIO   = menor + slash + _x + guion + _escenarios + mayor;

            LISTA_ESCENARIO   = LISTA_ESCENARIO + menor + _x + guion + TIPO_OBJETOS
                      | menor + _x + guion + TIPO_OBJETOS
                      | Empty;

            TIPO_OBJETOS   = PERSONAJES
                           | PAREDES
                           | EXTRAS
                           | META;

            PAREDES   = _paredes + mayor + LISTA_PAREDES + menor + slash + _x + guion + _paredes + mayor;

            EXTRAS   = _extras + mayor + LISTA_EXTRAS + menor + slash + _x + guion + _extras + mayor;

            META   =  _meta + mayor + POSICIONES_X_Y_OBJETOS + menor +  slash + _x + guion + _meta + mayor;

            PERSONAJES   = _personajes + mayor + LISTA_PERSONAJES + menor + slash + _x + guion + _personajes + mayor;

            LISTA_PERSONAJES   = LISTA_PERSONAJES + menor + _x + guion +  TIPO_PERSONAJES
                               |TIPO_PERSONAJES
                               |Empty;

            TIPO_PERSONAJES   = HEROES
                              |VILLANOS;
```

```
HEROES   =  _heroes + mayor + POSICIONES_X_Y_OBJETOS + menor + slash + _x + guion + _heroes + mayor;
POSICIONES_X_Y_OBJETOS.ErrorRule = SyntaxError + ";";

VILLANOS   = _villanos + mayor + POSICIONES_X_Y_OBJETOS + menor + slash + _x + guion + _villanos + mayor;


POSICIONES_X_Y_OBJETOS   = POSICIONES_X_Y_OBJETOS + identificador + parentAb + EXPRESION + coma + EXPRESION + parentCerr + ptoYcoma
                           |identificador + parentAb + EXPRESION + coma + EXPRESION + parentCerr + ptoYcoma
                           |Empty;
POSICIONES_X_Y_OBJETOS.ErrorRule = SyntaxError + ";";

LISTA_PAREDES   = LISTA_PAREDES + ATRIBUTOS_LISTA_PAREDES
                  |ATRIBUTOS_LISTA_PAREDES
                  |Empty;

ATRIBUTOS_LISTA_PAREDES   =identificador + parentAb + EXPRESION + coma + EXPRESION + parentCerr + ptoYcoma
                           |identificador + parentAb + EXPRESION + punto + punto +EXPRESION + coma + EXPRESION + parentCerr + ptoYcoma
                           |identificador + parentAb + EXPRESION + coma + EXPRESION + punto + punto + EXPRESION + parentCerr + ptoYcoma;
ATRIBUTOS_LISTA_PAREDES.ErrorRule = SyntaxError + ";";

LISTA_EXTRAS   =  LISTA_EXTRAS + ATRIBUTOS_LISTA_EXTRAS
                  |ATRIBUTOS_LISTA_EXTRAS
                  |Empty;

ARMAS   = menor + _x + guion +_armas + mayor + POSICIONES_X_Y_OBJETOS + menor + slash + _x + guion + _armas + mayor;

BONUS   = menor + _x + guion + _bonus + mayor + POSICIONES_X_Y_OBJETOS + menor + slash + _x + guion + _bonus + mayor;

ATRIBUTOS_LISTA_EXTRAS   = ARMAS
                           |BONUS;

EXPRESION   = EXPRESION + mas + EXPRESION
              | EXPRESION + menos + EXPRESION
              | EXPRESION + por + EXPRESION
              | EXPRESION + slash + EXPRESION
              | parentAb + EXPRESION + parentCerr
              | mas + EXPRESION
              | menos + EXPRESION
              | numero;
```

GLOSARIO

CLASE:
Es un conjunto de objetos los cuales comparten una misma estructura y comportamiento. Una clase es una plantilla que posee las variables y métodos comunes entre objetos de un cierto tipo.

GUI:
Graphic User Interface (en inglés) o Interfaz Gráfica de Usuario. Conjunto de formas y métodos que posibilitan la interacción entre un sistema y los usuarios. En otras palabras ventanas, botones, imágenes, fuentes, etc. Los cuales representan funciones, acciones e información.

XML:
Representar información estructurada en la web (todos documentos), de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por muy diversos tipos de aplicaciones y dispositivos.

IDE:
Integrated development environment (en inglés) o Entorno de Desarrollo Integrado. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un

GRAMATICA LIBRE DEL CONTEXTO:
es una gramática formal en la que cada regla de producción es de la forma:

$$V \rightarrow w$$

Analisis léxico
Scanner tiene las funciones de leer el programa fuente como un archivo de caracteres y dividirlo en tokens.

Análisis sintáctico
determina si la secuencia de componentes léxicos sigue la sintaxis del lenguaje y obtiene la estructura jerárquica del programa en forma de árbol, donde los nodos son las construcciones de alto nivel del lenguaje.

UML:
Lenguaje Unificado de Modelado (UML), este lenguaje permite diseñar, construir, visualizar y documentar un sistema, utilizando conceptos orientados a objetos.
Es el lenguaje de modelado de sistemas de software.

Irony
Es un analizador que se caracteriza por ser uno de los pocos, o porque no decir el único que funciona en base a la generación del AST (Abstract Syntax Tree ó Árbol de Sintaxis Abstracta).

Árbol de sintaxis abstracta (AST)
árbol de sintaxis, es una representación de árbol de la estructura sintáctica simplificada del código fuente escrito en cierto lenguaje de programación.