**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** 2](#_Toc533674570)

[**OBJETIVOS** 3](#_Toc533674571)

[**ALCANCE DEL MANUAL** 3](#_Toc533674572)

[**DESCRIPCION GENERAL DE LA APLICACIÓN** 4](#_Toc533674573)

[**GRAMATICA ARCHIVO DE CONFIGURACION (.XCONF)** 0](#_Toc533674574)

[**GRAMATICA ARCHIVO DE CONFIGURACION DE ESCENARIOS (.XESC)** 3](#_Toc533674575)

[**GLOSARIO** 0](#_Toc533674576)

\

# **INTRODUCCION**

E

ste 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 -> 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.