\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Firma del alumno

SifuScript

Documentation del proyecto

Luis EDUARDO Sifuentes INFANTE & Jens Wöhrle

2014

1. Descripción del Proyecto

**Visión**

La construcción de un lenguaje de programación en conjunto con su compilador, así como una maquina virtual que se encargue de ejecutar los programas hechos.

**Objetivo**

Crear un lenguaje para uso embebido dentro de un videojuego que facilite la comunicación entre el juego y el usuario. Estará basado en Lua, pero dará la importancia a los tipos de datos para que se tenga más control sobre el manejo de memoria. Además permitirá la creación de clases para el manejo de estructuras similares a las que son utilizadas comúnmente en los videojuegos, además de permitir más libertad con el manejo de datos para el usuario.

**Bitácora del desarrollo**

Al comienzo del proyecto fue el desarrollo del Compilador que genera el código intermedio. Para el Debugging el output fue un String con sus Cuádruples para cada ejecución. Ahora lo hemos cambiado en Binary-Output para la ejecución porque la máquina virtual JensVM trabaja con este – es más eficiente. Luego en la vía del proyecto fue el desarrollo de la máquina virtual paralelo con la verificación de Test-Cases. El redondeo del proyecto es la optimación y el Debugging de pequeños errores y la documentación.

**Reflexión**

Jens Wöhrle

Porque ya tuve experiencia en la programación con un pequeño proyecto en mi trabajo en Alemania y utilicé Flex and Yacc. A pesar de todo aprendí mucho porque este fue mi primera vez para la programación de una máquina virtual y hacer un proyecto que es más grande que mi primer proyecto fue una experiencia buena para profundizar mi conocimiento. Particularmente con mi conocimiento teorético que aprendí en mi último año en la clase de “Theoretic Informatics” en “Technial University of Munich” en Alemania y claro que si en la clase de Compiladores aquí al TEC que no tuve en último proyecto.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tu relfexion(Leonardo Sifu)

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

1. Descripción del Lenguaje

**Nombre**

El nombre del Compilador y su Idioma es “Sifuscript”. La máquina virtual se llama “JensVM” con referencia a los dos fundadores del proyecto.

**Características**

El estilo del idioma es muy similar que el estilo de los idiomas Java o C/C++ pero con una propia sintaxis desarrollada.

**Errores posibles**

En la Compilación recibes informaciones detalles con el error que hay en tu proyecto. Una pequeña descripción del error con su posición – línea – en el código. La máquina virtual no informa de errores en general y se estrella en errores del Runtime – como división por 0 o errores en el acceso de direcciones invalidas.

1. Descripción del Compilador y la Sintaxis

**Plataforma de desarrollo**

* Windows 8.1 y Linux Manjaro 1.0.9
* Tenemos dos versiones del Compilador. Una para Linux y una otra para Windows.
* C++11
* Utilizamos Flex/Yacc zu sus herramientas para desarrollarlo. Especialmente para el Compilador. Para la máquina virtual no utilizamos otras herramientas especiales.

1. **Léxico**

**Palabras Reservadas**

* true
* false
* and
* or
* not
* if
* else
* for
* do
* while
* function
* class
* import
* print

**Ids**

Empiezan con una letra o con un guion bajo, y de ahí puede seguirle un guion bajo, una letra o un número.

**Nombres de clase**

Empiezan con una letra mayúscula, y le puede seguir una letra, un guion bajo o un digito.

**Constantes**

Una constante entera solamente tiene dígitos y debe de ser al menos uno.

**Flotantes**

Una constante flotante es un digito, que le puede seguir más dígitos, seguido de un punto y luego al menos un digito.

**Booleanos**

Una constante booleana solo puede ser true o false.

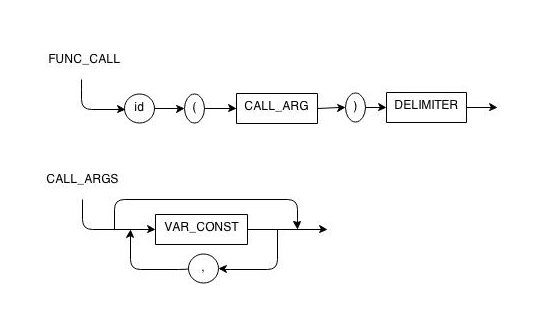
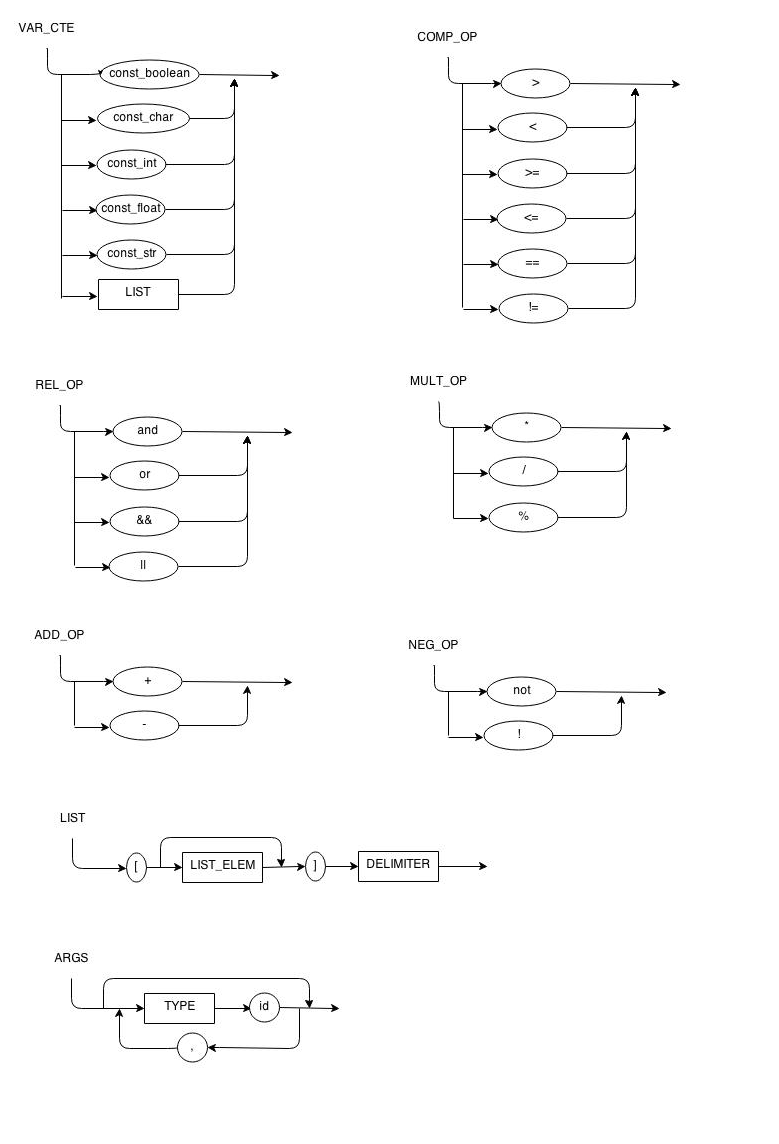
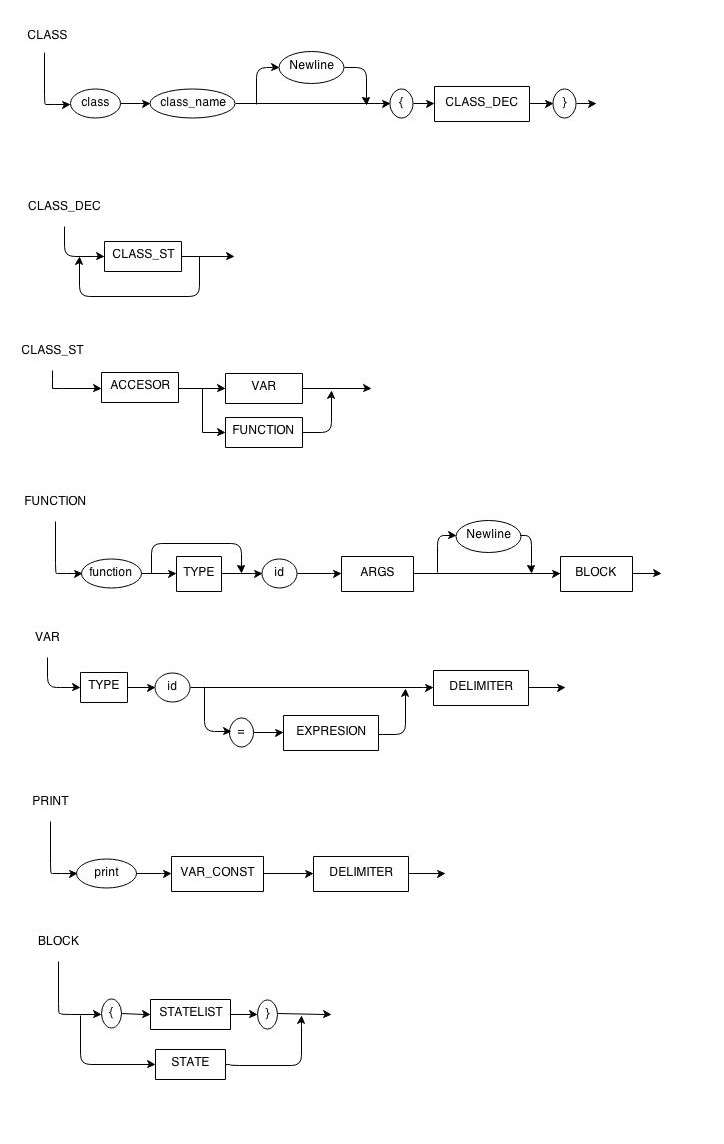
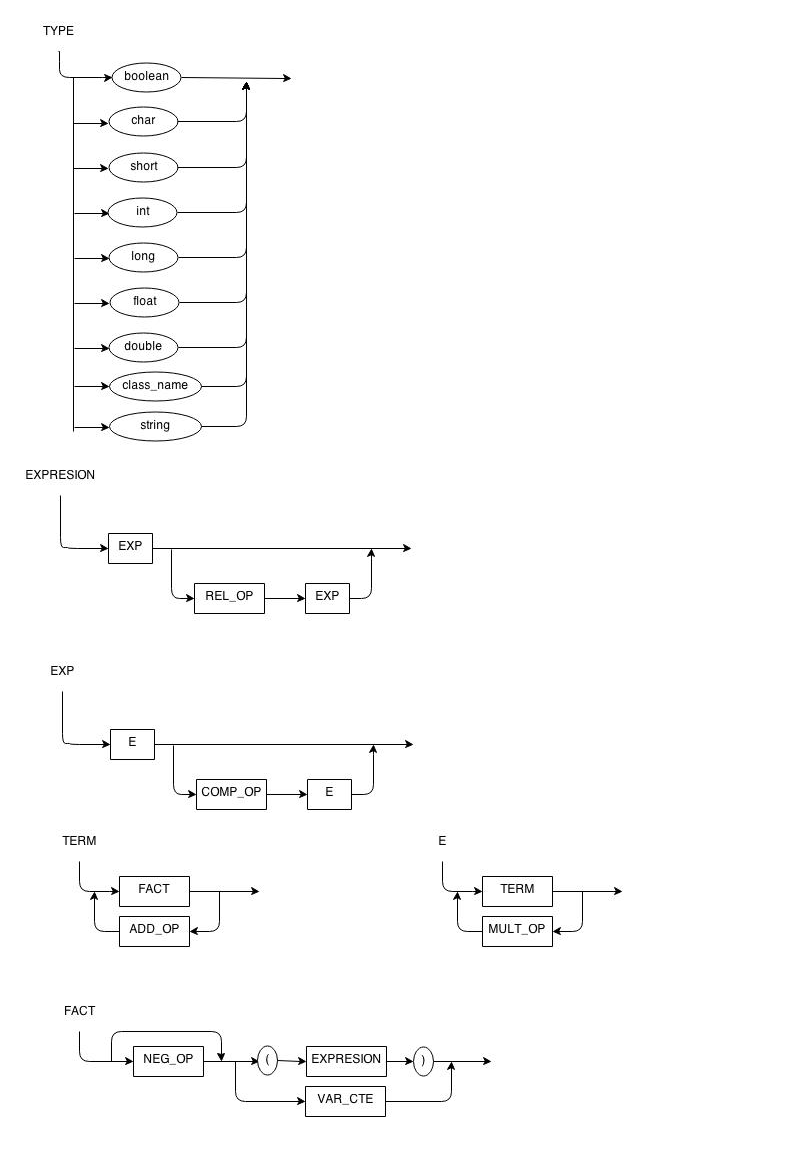
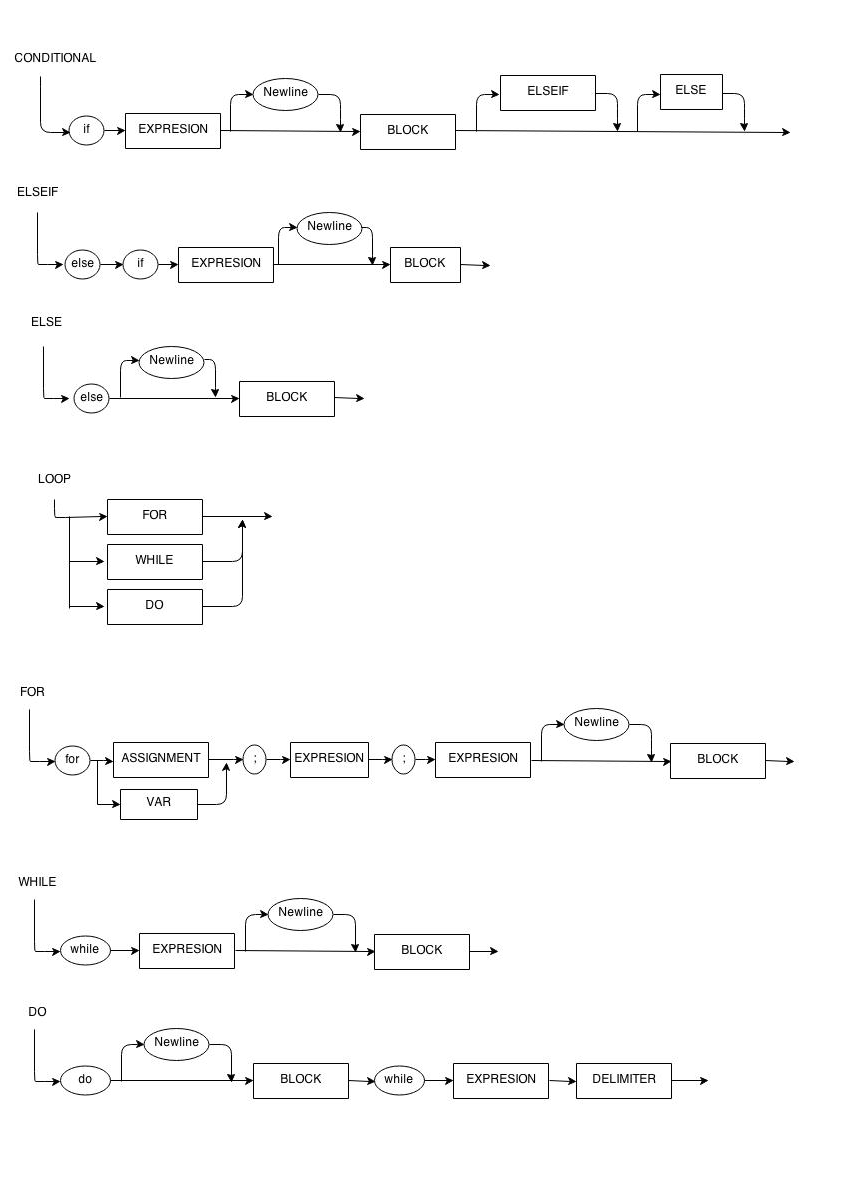
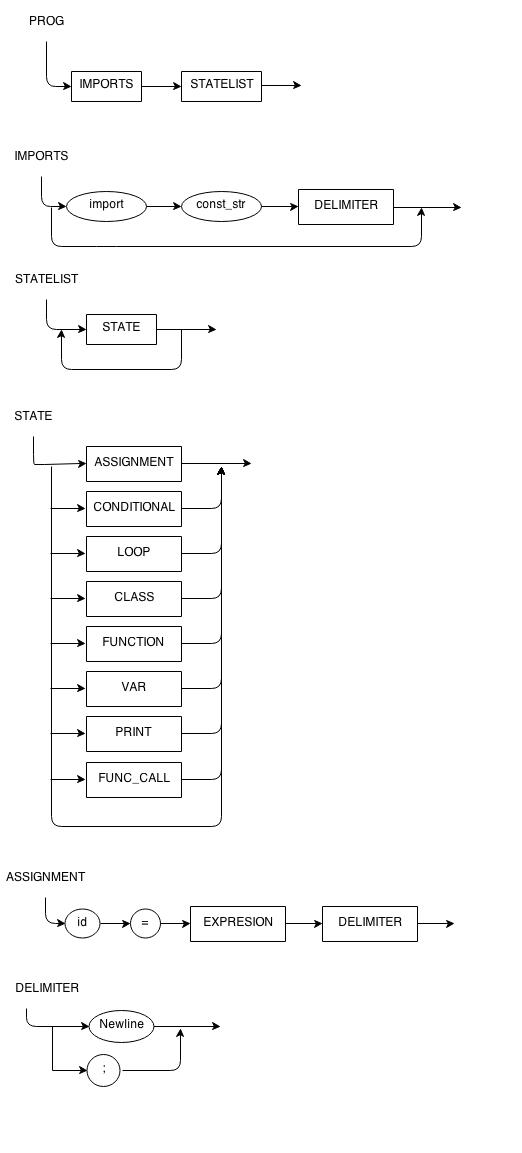
**Caracteres**

La constante de carácter debe de comenzar con comillas simples seguido de cualquier carácter (a excepción de otra comilla simple) y que termine en otra comilla simple.

**Strings**

La constante de string, al igual que la de carácter comienza con una comilla doble, seguida de cero o más caracteres a excepción de la comilla doble, y que termina en otra comilla doble.

1. **Diagramas de Sintaxis**



**Características semánticas**

Todos los importes se hacen al inicio de cada archivo

No es necesario terminar un estatuto con un punto y coma (;) sino que el salto de línea también funciona como delimitador.

Las funciones se declaran con un tipo, si se omite este tipo automáticamente se toma como si fuera una función de tipo void.

Cuando se utiliza una clase como tipo de dato es necesario haberla declarada antes de utilizarla, al igual que los ids.

Todos los operadores binarios son asociativos por la izquierda.

Orden de precedencia de mayor a menor

1. not !

2. \* /

3. + -

4. < > <= >= == !=

5. and or && ||

**Funciones especiales**

La función import se encarga de importar archivos externos y pegarlos dentro del documento.

La función print imprime un valor, solamente toma un argumento como válido.

**Tipos de datos**

|  |  |  |
| --- | --- | --- |
| Tipo | Tamaño | Limitaciones |
| char | 1 byte | Solo puede contener un solo carácter. De 0 a 255 |
| short | 2 bytes | De -32,768 a 32,767 |
| int | 4 bytes | De -2,147,483,648 a 2,147,483,647 |
| long | 8 bytes | De |
| float | 4 bytes | De 1.2E-38 a 3.4E+38 |
| double | 8 bytes | De 2.3E-308 a 1.7E+308 |
| string | Máximo 256 bytes |  |

**Bibliografía**

Tutorial de una VM en un videojuego

<http://gameprogrammingpatterns.com/bytecode.html>

Gramática de Lua

<http://lua-users.org/wiki/LuaGrammar>

Gramática de ActionScript 3

<http://svn.badgers-in-foil.co.uk/metaas/trunk/src/main/antlr/org/asdt/core/internal/antlr/AS3.g>

1. Descripción de la máquina virtual

**Herramientas**

No hemos utilizado herramientas especiales para desarrollar la máquina virtual.

**Descripción y estructura del proyecto**

|  |  |
| --- | --- |
| Main: | El “User interface” para la ejecución de la máquina virtual |
| Driver: | El corazón de la ejecución que organiza la ejecución y une la Memoria y el Parser. |
| Command\_Parser: | decodifica el Cuádruplo y ejecútalo |
| Memory, MemoryManager: | Hace la organización de la Memoria y organiza el escribir o las otra acciones en la Memoria. |

1. Pruebas del funcionamiento del lenguaje

Ejemplos del funcionamiento del lenguaje puedes encontrar en la carpeta “tests”. El código del lenguaje Sifuscript en “.sifu” y el código intermedio en “.jns”.

**User manual**

La ejecución y el empleo son simple.

Para utilizar el Compilador tiene que ejecutar el archivo binario del Compilador que se llama “sifuscript” con la dirección del .sifu archivo que contiene el código por el primer argumento. El resultado – un archivo de .jns – puede ejecutar con la máquina virtual JensVM. Por eso solo tiene que ejecutar el archivo “JensVM” con la dirección del archivo de .jns que contiene el código intermedio en el primer argumento de JensVM.

La mayoridad del debugging hicimos en Linux este es la mejor versión. En Windows pueden existir tal vez efectos inesperados.

**Windows:**

En Windows puede encontrar los archivos para la ejecución en

Sifuscript Compilador:

“Sifuscript/sifuc/VisualStudioFiles/Sifuscript/Debug” el archivo “Sifuscript.exe”

JensVM máquina virtual:

“Sifuscript/jensvm/VisualStudioFiles/jensvm/Debug” el archivo “JensVM.exe”

**Linux:**