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

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.

**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 teorética que aprendí en mi último año con la clase de “Theoretic Informatics” en “Technial University of Munich” en Alemania y claro que si con la clase …

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++.

**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

**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.

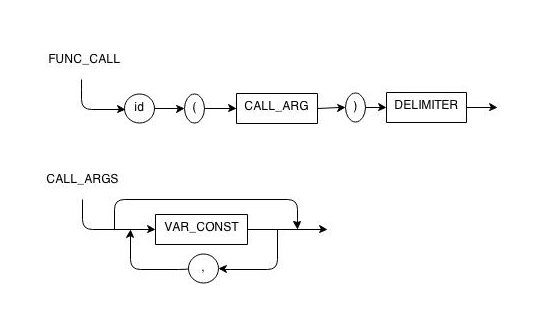
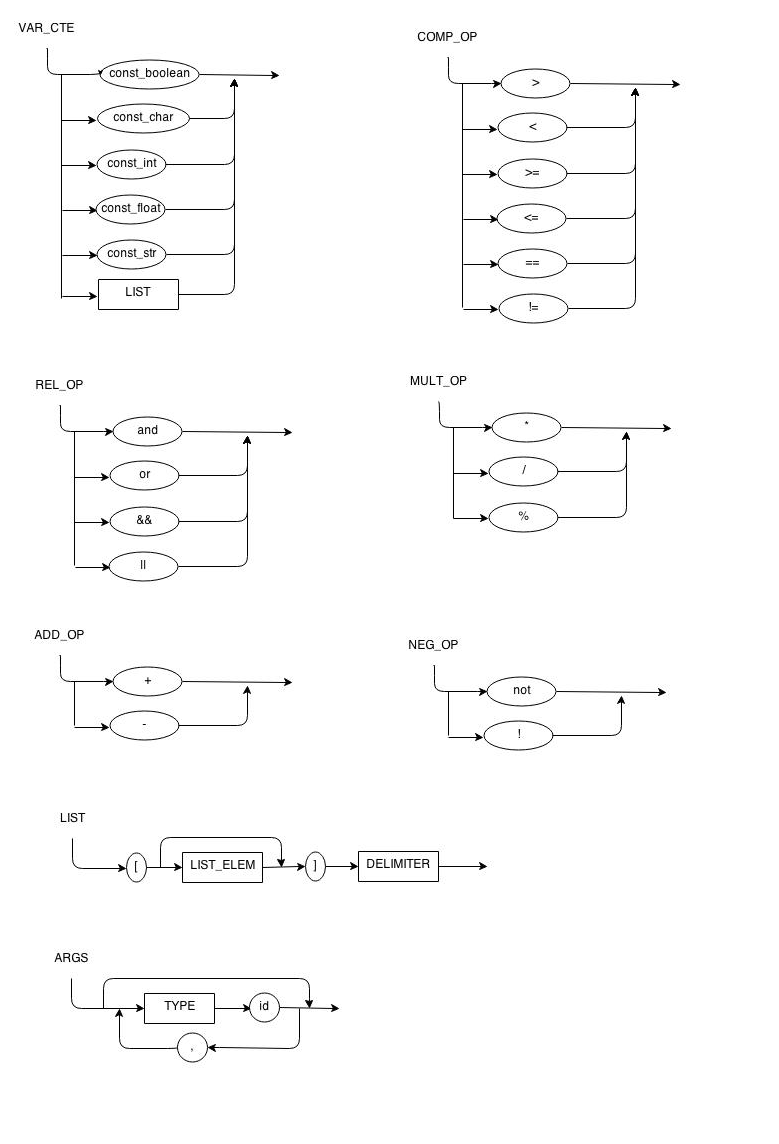
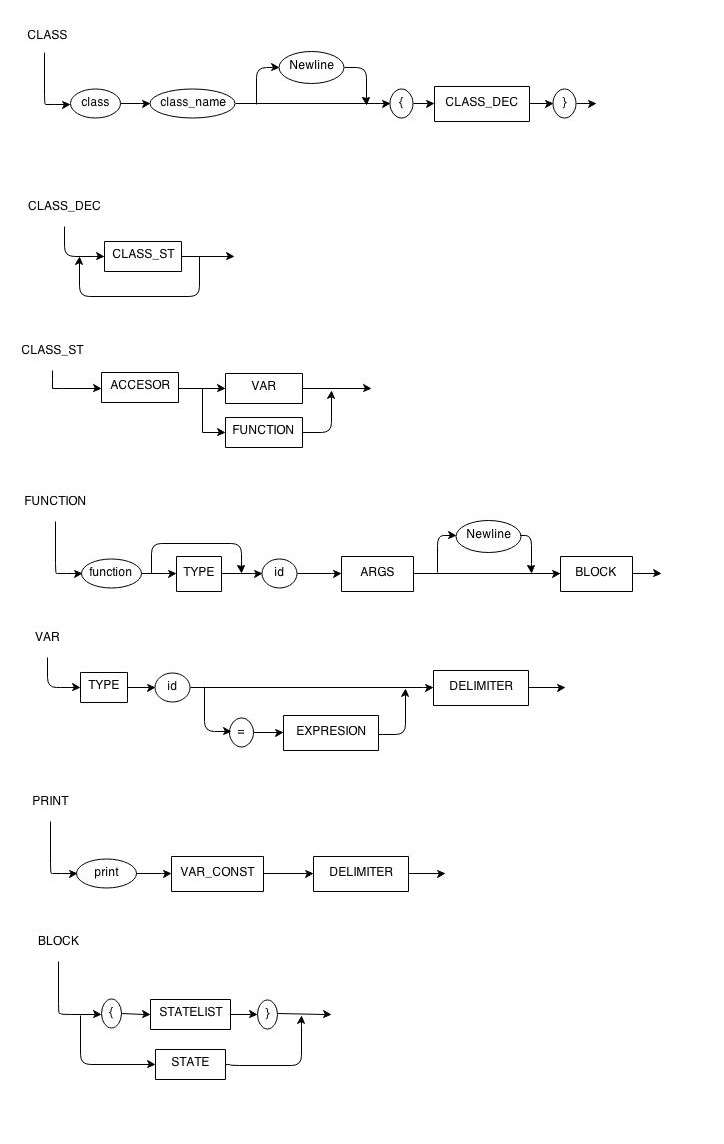
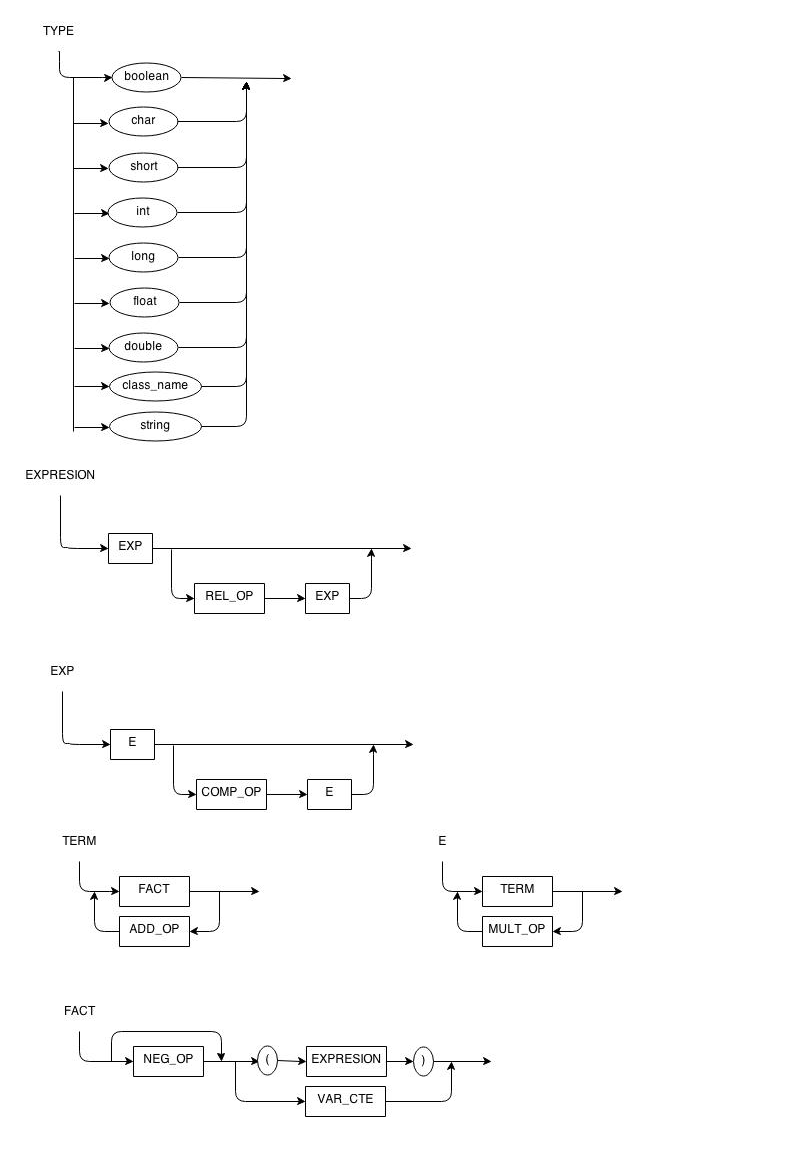
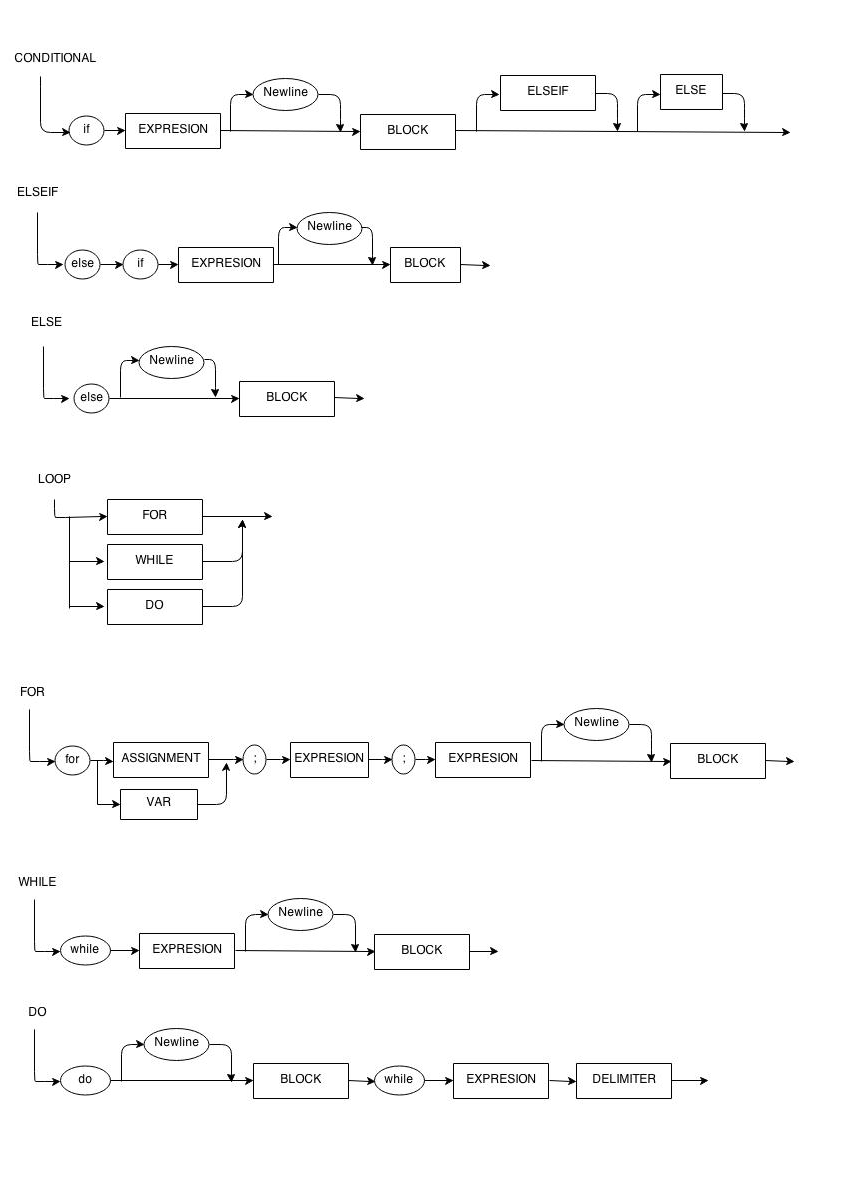
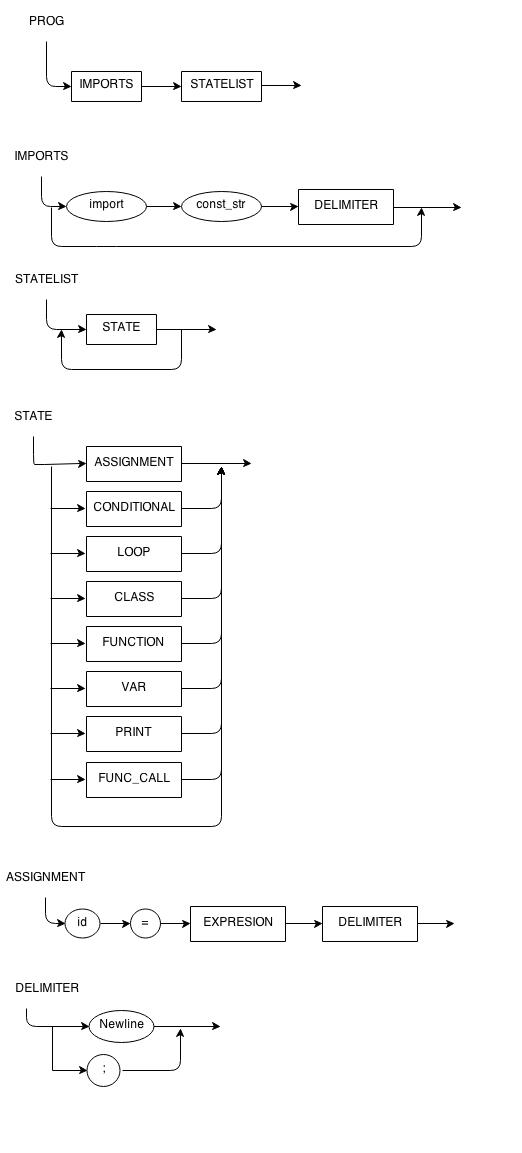
**Caracteres**

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.

**Diagramas de Sintaxis**



**Características semánticas**

Todos los imports 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

not !

\* /

+ -

< > <= >= == !=

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 |  |

**Plataforma de desarrollo**

Windows 8.1 y Linux Manjaro 1.0.9

C++ 11

Utilizamos Flex/Yacc para desarrollarlo.

**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 del Proyecto

**Nombre**

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

1. Descripción del Compilador
2. Descripción de la máquina virtual
3. Pruebas del funcionamiento del lenguaje