Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Lenguajes Formales y de Programación

Primer semestre de 2017
Catedrático: Ing. Luis Fernando Espino Barrios
Tutor académico: Rolando José Minera Alejandro

Proyecto de Laboratorio

1. INFORMACION GENERAL

1.1 OBJETIVO GENERAL

Esta práctica tiene como objetivo reforzar los conocimientos obtenidos hasta el momento en el curso y aplicar los conceptos ya aprendidos en clase y laboratorio para el diseño y uso de un compilador.

1.2 OBJETIVOS ESPECÍFICOS

- Que el estudiante practique el análisis léxico y sintáctico.
- Que el estudiante realice un analizador léxico a partir de un AFD
- Que el estudiante realice un analizador sintáctico a partir de la gramática.

1.3 DESCRIPCIÓN

La práctica consiste en realizar un juego a partir del análisis de un archivo de entrada que contendrá las instrucciones necesarias para generar el campo, los tesoros, los enemigos y los atributos del personaje principal.

Se deberá realizar un análisis léxico al archivo de entrada, para verificar que el archivo no contenga caracteres que no pertenecen al lenguaje. Después de esto se debe realizar un análisis sintáctico, esto para verificar que las instrucciones se encuentren de manera correcta.

2. Descripción del Lenguaje

El lenguaje se divide en diferentes secciones, permitiendo que esté sea de fácil edición para cualquier usuario, el lenguaje será de tipo XML (etiquetas). Las secciones que podemos encontrar son:

- **Variables**: En esta sección es posible definir variables. Existen 2 tipos de accesibilidad de las variables: globales y locales. Existen 4 tipos de variables: cadenas, booleanas y enteros.

- **Jugador**: En esta sección es posible definir la forma física del jugador y los atributos de esté, como por ejemplo: vidas, poderes, etc.

- **Campo**: En esta sección es posible definir la forma que tendrá el campo, tamaño, obstáculos, bonos, etc.

- **Acciones**: En esta sección se podrán definir métodos que contendrán una serie de condición a evaluar cuando el personaje alcance a obtener los bonos del campo.

- **Enemigos**: En esta sección se podrá definir los atributos que definirán la forma física de los enemigos, sus posiciones iniciales y una serie de movimientos que estos realizarán a través del campo.

3. Definición XML

3.1 XML Variables

Formato:

Esta etiqueta definirá el inicio de la sección de variables, dentro de esta pueden existir varias etiquetas de definición de variable.

Ejemplo:

```
<variables>
        <var> <%var>
        <var> <%var>
        …
<%variables>
```

Atributos:

La etiqueta de definición de variable puede contener los siguientes atributos

- **Accesibilidad**: Este atributo permite definir si la variable podrá ser utilizada en el siguiente nivel del juego. Ese atributo será definido en la primera posición de atributos de la etiqueta, para definir que es global deberá venir la palabra global, si es únicamente local el atributo no existirá.

  Ejemplo:

```
Global: <var global … > …<%var>
Local: <var> <%var>
```

- **Tipo**: Este atributo definirá el tipo de dato que utilizará la variable.

  Ejemplo:

```
Cadena: <var global tipo = ”Cadena” … > … <%var>
Entero: <var global tipo = ”Entero” … > … <%var>
Booleano: <var global tipo =  ”Booleano” … > … <%var>
```

- **Valor**: Este atributo definirá el valor que tendrá la variable, de venir vacío las variables tomaran los siguientes valores:
  - Cadena - vacía
  - Entero - 0
  - Booleano - true.

  Ejemplo:

```
<var global tipo=”Cadena” valor=”Hola mundo”> … <%var>
<var tipo=”Booleano” valor=”falso”>… <%var>
<var tipo=”Entero” valor=”320”>… <%var>
```

  Nota: Dentro de valor, si el tipo fuese Entero, su valor puede ser expresado como una expresión aritmética, siendo posible utilizar los siguientes operadores:
  - Suma +
  - Resta –
  - Multiplicación *
  - División /
  - Paréntesis ( )
  - Corchetes [ ]
  - Llaves { }

- **Nombre**: El nombre de la variable será definido entre la etiqueta de inicio y la de cierre.
  Ejemplo:

```
<var tipo = ”Entero” valor = ”320” > Mi_variable <%var>
```

  Nota: Los nombres de variables deben iniciar con un carácter y luego pueden venir más caracteres, números, guion medio o guion bajo. No puede iniciar con algo que no sea carácter, puede contener mayúsculas o minúsculas y no puede contener espacios.

3.2 XML Jugador

Formato:

Esta etiqueta definirá el inicio de la sección de jugador, dentro de esta existen ciertas etiquetas que definirán los atributos de potenciación.

Ejemplo:

```
<jugador … >
        <vidas>…<%vidas>
        <poder> …<%poder>
        <remoto> …<%remoto>
        <fantasma>… <%fantasma>
<%jugador>
```

Atributos:

La etiqueta de definición de jugador contendrá los siguientes atributos dentro de la etiqueta y las siguientes sub-etiquetas:

- **Color**: Esta etiqueta debe recibir algún color en texto, ya sea directamente o por medio de variables, el jugador se debe mostrar como una figura geométrica, definida por el usuario, rellena del color que se indica acá, esto en caso de que el atributo traje este vacío o no se encuentre la imagen de referencia.
- **Traje**: Esta etiqueta debe contener una dirección de una imagen, la cual será la forma física del jugador, esta puede ser definida directamente como texto o por variables.

Ejemplo:

```
<jugador color = “rojo” traje = “/img/imagen.png” > < %jugador>
```

Vida:

Esta etiqueta define el número de vidas que tendrá el jugador durante la pantalla. Su valor debe ser de tipo entero o de una variable que sea entera.

Ejemplo:

```
<vidas>valor<%vidas>
<vidas> variable <%vidas>
```

Poder:

Esta etiqueta indica el nivel de poder que tendrá el jugador, deberá ser un número entero mayor a 0 y menor o igual a 10. Su valor debe ser de tipo entero o de una variable que sea entera. La función de este poder es el de indicar el número de cuadros que abarcara el golpe de la bomba para los 4 puntos cardinales.

Ejemplo:

```
<poder> valor <%poder>
<poder> variable <%poder>
```

Remoto:

Esta etiqueta indica si el atributo remoto está o no activado, el tipo de valor de esta etiqueta es booleano (verdadero o falso) por valor o por variable. La función de este poder es el de permitir que el jugador detone las bombas de manera manual, presionando una tecla que el usuario debe definir.

Ejemplo:

```
<remoto> valor <%remoto>
<remoto> variable <%remoto>
```

Fantasma:

Esta etiqueta indica si el atributo fantasma está o no activado, el tipo de valor de esta etiqueta es booleano (verdadero o falso) por valor o por variable. La función de este poder es el de permitir que el jugador pueda atravesar los obstáculos, mas no a los enemigos.

Ejemplo:

```
<fantasma> valor <% fantasma >
< fantasma>”variable” <% fantasma >
```

3.3 XML Campo

Formato:

Esta etiqueta y sus atributos definirán la forma del campo, dentro de esta existen ciertas etiquetas que definirán las posiciones de los obstáculos, tesoros, salida, etc.

Ejemplo:

```
<campo>
        <rocas>
                <roca>
                        <x> <%x>
                        <y> <%y>
                <%roca>
                <roca> … <%roca>
                …
        <%rocas>
        <tesoros>
                <llave>
                        <x><%x>
                        <y><%y>
                <%llave>
                <bono>
                        <x><%x>
                        <y><%y>
                <%bono>
                <salida>
                        <x><%x>
                        <y><%y>
                <%salida>
        <%tesoros>
<%campo>
```

Atributos:

La etiqueta de definición de campo contendrá los siguientes atributos dentro de la etiqueta y las siguientes sub-etiquetas:

- **Ancho**: Este atributo debe recibir algún valor entero, variable o valor, el cual indicara el número de posiciones que existirán de ancho en el campo.
- **Alto**: Este atributo debe recibir algún valor entero, variable o valor, el cual indicara el número de posiciones que existirán de alto en el campo.
- **Color**: Este atributo debe recibir algún color en texto, ya sea directamente o por medio de variables, el juego debe mostrar el fondo del campo de este color si el atributo textura este vacío o no se encuentre la imagen de referencia.
- **Textura**: Este atributo debe contener una dirección de una imagen, la cual será dibujado de fondo del campo, esta puede ser definida directamente como texto o por variables.

Ejemplo:

```
<campo ancho=”20” alto=”10” color=”amarillo” textura=”/img/campo.png”>
<%campo>
```

Rocas:

Esta etiqueta contendrá múltiples sub-etiquetas las cuales definirán la forma de los obstáculos y la posición de estos.

Ejemplo:

```
<rocas> … <%rocas>
```

Roca:

Esta sub-etiqueta contendrá como atributo y como sub-etiquetas lo siguiente, la sub-etiqueta roca puede venir seguida n veces, únicamente la primera etiqueta es obligatoria sus atributos, de no definirse en la siguiente toma los mismos que la anterior.

- **Atributos**:
    - **Color**: Este atributo debe recibir algún color en texto, ya sea directamente o por medio de variables, el juego debe mostrar los obstáculos como una figura geométrica rellena de este color si el atributo textura este vacío o no se encuentre la imagen de referencia.
    - **Textura**: Este atributo debe contener una dirección de una imagen, la cual será dibujado de fondo del campo, esta puede ser definida directamente como texto o por variables.

- **Durable**: Este atributo recibe de valor un 1 o 0, por valor o por variable, el cual indica si el obstáculo puede ser destruido (1) o no (0), de no contener el atributo se toma como 1.

Ejemplo:

```
<roca color = “negro” textura = ”/img/roca1.png” durable = ”1”><%roca>
```

- **X**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición x en la que se encuentra dibujado el obstáculo.

  Ejemplo:

```
<X>5<%X>
```

- **Y**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición y en la que se encuentra dibujado el obstáculo.

  Ejemplo:

```
<Y>2<%Y>
```

Tesoros:

Esta etiqueta contendrá múltiples sub-etiquetas las cuales definirán la forma de los tesoros, las acciones a tomar y la posición de estos.

Ejemplo:

```
<tesoros> … <%tesoros>
```

Llave:

Esta sub-etiqueta contendrá como atributo y como sub-etiquetas lo siguiente, la sub-etiqueta llave puede encontrarse en el archivo únicamente 1 vez.

- **Atributos**:

  - **Color**: Este atributo debe recibir algún color en texto, ya sea directamente o por medio de variables, el juego debe mostrar la llave como una figura geométrica de este color rellenada si el atributo textura este vacío o no se encuentre la imagen de referencia.

- **Textura**: Este atributo debe contener una dirección de una imagen, la cual será la textura al dibujar la llave, esta puede ser definida directamente como texto o por variables.

- **Acción**: Este atributo recibe de valor un texto, por valor o por variable, el cual indica una función que se definirá más adelante y que debe ser llamada una vez el jugador toma el tesoro.

Ejemplo:

```
<llave color = ”anaranjado” textura = ”/img/llave.png” acción = ”” > <%llave>
```

- **X**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición x en la que se encuentra dibujado el tesoro.

  Ejemplo:

```
<X>5<%X>
```

- **Y**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición y en la que se encuentra dibujado el tesoro.

  Ejemplo:

```
<Y>2<%Y>
```

Bono:

Esta sub-etiqueta contendrá como atributo y como sub-etiquetas lo siguiente, la sub-etiqueta bono puede encontrarse en el archivo como máximo 3 veces.

- **Atributos**:

  - **Color**: Este atributo debe recibir algún color en texto, ya sea directamente o por medio de variables, el juego debe mostrar el bono como una figura geométrica de este color rellenada si el atributo textura este vacío o no se encuentre la imagen de referencia.

  - **Textura**: Este atributo debe contener una dirección de una imagen, la cual será la textura al dibujar el bono, esta puede ser definida directamente como texto o por variables.

  - **Acción**: Este atributo recibe de valor un texto, por valor o por variable, el cual indica una función que se definirá más adelante y que debe ser llamada una vez el jugador toma el tesoro.

Ejemplo:

```
<bono color = ”rojo” textura = ”/img/bono.png” acción = ”” > <%bono>
```

- **X**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición x en la que se encuentra dibujado el tesoro.

  Ejemplo:

```
        <X>5<%X>
```

- **Y**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición y en la que se encuentra dibujado el tesoro.

  Ejemplo:

```
        <Y>2<%Y>
```

Salida:

Esta sub-etiqueta contendrá como atributo y como sub-etiquetas lo siguiente, la sub-etiqueta salida puede encontrarse en el archivo únicamente 1 vez.

- **Atributos**:
  - **Color**: Este atributo debe recibir algún color en texto, ya sea directamente o por medio de variables, el juego debe mostrar la salida como una figura geométrica de este color rellenada si el atributo textura este vacío o no se encuentre la imagen de referencia.
  - **Textura**: Este atributo debe contener una dirección de una imagen, la cual será la textura al dibujar la salida, esta puede ser definida directamente como texto o por variables.
  - **Acción**: Este atributo recibe de valor un texto, por valor o por variable, el cual indica una función que se definirá más adelante y que debe ser llamada una vez el jugador toma la salida.

  Ejemplo:

```
<salida color = ”verde” textura = ”/img/salida.png” acción = ”” > <%salida>
```

- **X**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición x en la que se encuentra dibujada la salida.
  Ejemplo:

```
<X>5<%X>
```

- **Y**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición y en la que se encuentra dibujada la salida.
  Ejemplo:

```
<Y>2<%Y>
```

3.4 XML Acciones

Formato:

Esta etiqueta y sus atributos, definirán las posibles acciones que se realizarán de manera automática al ser llamados cuando se tome un tesoro.

Ejemplo:

```
<acciones>
        <accion><%accion>
        …
<%acciones>
```

Nota: · La etiqueta de definición de acciones no contiene atributos, únicamente las siguientes sub-etiquetas.

Acción:

Las acciones a realizar se encontrarán descritas dentro de esta sub-etiqueta, y debe contener los siguientes atributos:

- **Id**: Este atributo debe recibir un valor de texto, por valor o por variable, el cual definirá el nombre con el cual se puede llamar esta acción.

Ejemplo:

```
<acción id = ”Toma_llave”> (condicion) <%accion>
```

Condición:

La definición de las acciones tiene limitación, el alcance será el siguiente:

- **Si**: Condición que será evaluada y si es cumplida se realizan las acciones que estén dentro.

Ejemplo:

```
Si (X < 10){
        Valor = falso;
        Punteo += 10;
}
```

- **Ademas si**: Esta condición será evaluada si la condición anterior no se cumple, si esta se cumple se deben de realizar las acciones que se declaren adentro.

Ejemplo:

```
Ademas si (X >= 10){
        Valor = falso;
        Punteo += 10;
}
```

- **Si no**: Se llegara a esa condición si todas las anteriores no se cumplieron, y las acciones que se declaren adentro se ejecutaran.

Ejemplo:

```
Si no{
        Valor = verdadero;
        Punteo += 10;
}
```

Operadores y acciones:

Operadores Lógicos:

Estos operadores serán utilizados dentro de las condiciones para validar si se realiza o no una acción. Los posibles operadores son los siguientes:

- <
- >
- = =
- <=
- >=
- Y
- O
- ( )

Acciones Aritméticas:

Estas acciones serán descritas dentro de las condiciones, si existen, y servirán para cambiar los valores de las variables globales y locales. Los posibles operadores aritméticos son los siguientes.

- +
- –
- *
- /
- + =
- - =
- ( )

3.5 XML Enemigos

Formato:

Esta etiqueta y sus atributos definirán a los enemigos dentro del campo, dentro de esta existen ciertas etiquetas que definirán las posiciones de los enemigos, movimientos, etc.

Ejemplo:

```
<enemigos>
        <enemigo>
                <movimientos>
                        <movimiento><%movimiento>
                        …
                <%movimientos>
        <%enemigo>
        …
<%enemigos>
```

Enemigo:

Esta etiqueta contendrá los siguientes atributos y múltiples sub-etiquetas las cuales definirán la forma de los enemigos, movimientos y la posición de estos.

- **Color**: Este atributo debe recibir algún color en texto, ya sea directamente o por medio de variables, el juego debe mostrar al enemigo como una figura geométrica rellena de este color si el atributo textura este vacío o no se encuentre la imagen de referencia.
- **Textura**: Este atributo debe contener una dirección de una imagen, la cual será dibujada representando al enemigo esta puede ser definida directamente como texto o por variables.
- **X**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición x en la que se encuentra dibujado el enemigo de manera inicial. Posición en X.
- **Y**: Dentro de esta etiqueta debe existir un valor, por valor o por variable, que indica la posición y en la que se encuentra dibujado el enemigo de manera inicial. Posición en Y.

Ejemplo:

```
<enemigo color =”morado” textura=”/img/enemigo1.png” X=”5” Y=”6’>
 <%enemigo>
```

Movimientos:

La etiqueta de definición de movimientos contendrá las siguientes sub-etiquetas:

Movimiento:

Dentro de esta etiqueta será definida una dirección a la cual el enemigo debe moverse de manera automática, esta etiqueta puede ser definida n veces, por lo cual los movimientos deben realizarse uno detrás de otro.

Ejemplo:

```
<movimientos>
        <movimiento> izquierda <%movimiento>
        <movimiento> derecha <%movimiento>
        …
<%movimientos>
```

Los posibles movimientos a realizar son:

- Izquierda
- Derecha
- Arriba
- Abajo

Es posible repetir un movimiento, para lo cual, para ahorrar tiempo y código, será posible colocar una etiqueta de la siguiente manera:

```
<movimientos>
    <movimiento> izquierda <%movimiento>
    <movimiento> derecha <%movimiento>
    <%movimiento>
<%movimientos>
```

Lo cual indica que el último movimiento será también derecha.

4. Funcionalidad

Es importante tomar las siguientes consideraciones para la funcionalidad del proyecto al momento de la calificación:

- Los archivos deberán tener un nombre definido, el cual será el siguiente, donde el # es un número.
    - Ejemplo: Bombermap_#.xml

- La aplicación debe realizar un análisis automático de cada mapa, previo a jugar, por lo que el editor únicamente servirá para modificar archivos que posiblemente contengan errores.
- El momento de iniciar el juego, la aplicación debe ir a la carpeta definida abajo, y buscar el mapa con el menor #, analizarlo y de no contener errores iniciar el juego, si contienen errores debe generar lo especificado abajo y debe buscar el siguiente archivo dependiendo del #.
- Una vez iniciado el juego, el jugador debe matar a todos los enemigos y encontrar la llave para poder utilizar la salida.
- Al momento de ganar el nivel, la aplicación debe cargar el siguiente mapa dependiendo del #.
- El tiempo disponible para cada campo puede ser editable dentro de la aplicación de manera global para todos los campos.

- Debe existir una variable global para cada mapa que se llamara PTS, la cual puede ser usada en cualquier parte del código y en cualquier operación, la cual definirá los puntos que lleva hasta el momento el jugador. Debe ser visible para el usuario.
- Si un tesoro es definido en la misma posición de un obstáculo, este debe de estar debajo del obstáculo y al ser destruido se debe mostrar.
- Las teclas para manejar al jugador pueden ser editable dentro de la interfaz de juego, teniendo por lo menos teclas de los 4 movimientos y 2 botones, 1 para colocar la bomba y otro para activar poderes, si se tuviesen.

5. Interfaz grafica

Componentes mínimos de la interfaz:

- **Editor de texto**: Debe ser un campo de texto en el cual se puedan escribir nuevos archivos.
- **Abrir**: Debe permitir abrir archivos con extensión .XML, el cual contendrá código de la forma descrita anteriormente.
- **Analizar**: Debe realizar el análisis léxico y sintáctico, si no encuentra ningún error se le debe notificar al usuario, de lo contrario debe generar una página html con los errores y las tablas descritas anexos, la página debe tener un estilo visiblemente atractivo y el nombre de la página debe ser ERRORES_#.HTML, donde # es el mismo que el archivo del campo. Exista errores o no siempre se debe generar la tabla de simbolos.
- **Jugar**: Inicia el juego.
- **Acerca de**: Debe mostrar en una ventana desplegable los datos del estudiante y del curso.
- **Salir**: Debe terminar la ejecución de la aplicación.

6. Ejemplos de funcionalidad

6.1 Estructura mínima de la tabla de símbolos:

Se deberá de generar una página web la cual contendrá los datos del estudiante, la fecha-hora en la que se generó el archivo y la tabla de símbolos. La estructura mínima de la tabla es la siguiente:

No.	Lexema	Token	Palabra reservada	Línea	Columna
1	45	Numero	NO	Linea 1	Columna 1
2	<campo>	Etiqueta apertura: campo	SI	Linea 2	Columna 5
3	Campo	Etiqueta apertura: campo	SI	Linea 3	Columna 6

6.2 Estructura mínima de la tabla de errores léxicos (Tabla 1):

En el caso que existieran errores léxicos de deberá generar una página web la cual contendrá los datos del estudiante, fecha-hora en la que se generó el archivo y una tabla de errores, la estructura de la tabla de errores deberá ser:

No.	Carácter	Descripción	Línea	Columna
1	$	Carácter no permitido	Linea 2	Columna 1
2	@	Carácter no reconocido	Linea 2	Columna 15
3	~	Carácter desconocido	Linea 6	Columna 26

6.3 Estructura mínima de la tabla de errores sintácticos (Tabla 2):

En el caso que existieran errores léxicos de deberá generar una página web la cual contendrá los datos del estudiante, fecha-hora en la que se generó el archivo y una tabla de errores, la estructura de la tabla de errores deberá ser:

No.	Token	Lexema	Palabra reservada	Ubicación	Token Esperado
1	5	Hola	NO	Linea 2, columna 4	ID
2	campo	campo	SI	Linea 2, columna 5	Numero
3	Token 7	+	SI	Linea 3, columna 86	logico

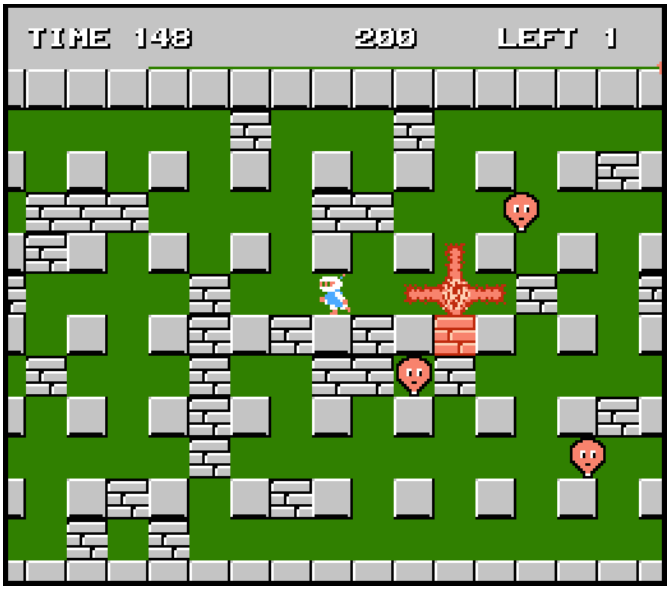
6.4 Referencias y propuestas:

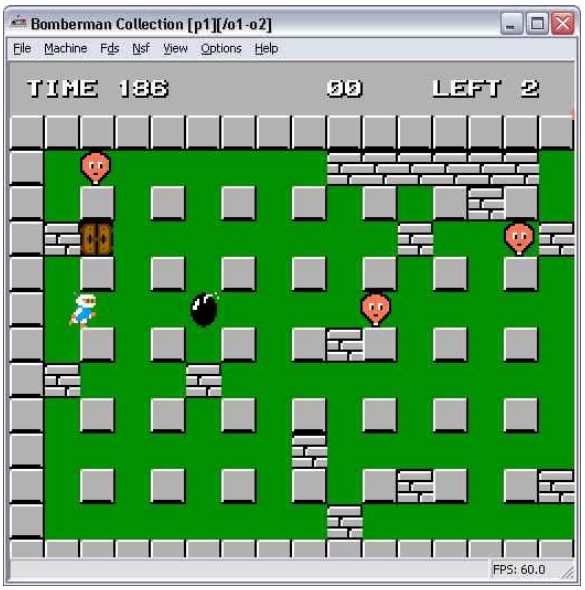
Referencias:

http://www.8bbit.com/bomberman

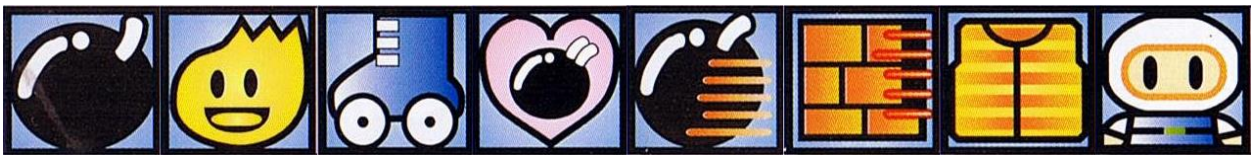
Propuestas – Bomberman NES:

Ejemplo campo, personaje, enemigo, roca, salida, bomba y explosion:





Ejemplo Bono:



7. Entrega y restricciones

7.1 Entregables Fase I

- Analizador Léxico y su respectivo resumen de análisis: Tokens y errores
- Autómatas y Expresiones Regulares
- Código Fuente con Validación por medio de estados
- Interfaz grafica
- Archivos de Prueba

7.2 Entregables Fase II

- Analizador Sintáctico y Reporte de Errores
- Juego Finalizado
- Manual de Usuario y Técnico

7.3 Restricciones

- El proyecto se debe de desarrollar de forma individual.
- El proyecto se deberá de desarrollar utilizando C#, por lo que se recomienda utilizar Microsoft Visual Studio como IDE.
- No se deberá de utilizar ninguna librería para la lectura de XML.
- Copia parcial o total del proyecto tendrán una nota de 0 puntos y se notificara a la escuela para que se apliquen las sanciones correspondientes.
- La calificación del proyecto será personal y durará como máximo 30 minutos, en un horario que posteriormente será establecido, se debe de tomar en cuenta que durante la calificación no podrán estar terceras personas alrededor.
- **No se dará prórroga.**

7.4 Fecha de entrega

- Fase I: martes 04 de abril de 2017.
- Fase II: sábado 29 de abril de 2017.

Formato de entrega:

[LFP]ProyectoFase1_<carnet>.rar

[LFP]ProyectoFase2_<carnet>.rar