Enrique Acedo 7 de febrero de 2016

Generación de niveles de Mario Bros por gramáticas y algoritmo evolutivo

La práctica consiste en generar un algoritmo genético basado en gramáticas que genere niveles óptimos para el juego de Mario Bros.

La gramática utilizada ha sido la siguiente:

```
#A# S
#N# NIVEL ELEMENTOS ELEMENTO ENEMIGOS ENEMIGO OBJETOS OBJETO MONEDAS BLOQUE_-
MONEDAS BLOQUE BLOQUE_POWER TIPO_ENEMIGO CON_ALAS NUMERO CIFRA DOS_ARGUMENTOS
TRES_ARGUMENTOS CUATRO_ARGUMENTOS CINCO_ARGUMENTOS TUBE TUBE_FLOOR PLATFORM
CANNON GAP

#T# ; % - , < > ( ) 0 1 2 3 4 5 6 7 8 9 true false armored_turtle chomp_flower
goompa  green_turtle  red_turtle  monedas bloque_monedas bloque bloque_power
tube tube_floor platform cannon gap

S ::= NIVEL
NIVEL ::=  % ELEMENTOS % ENEMIGOS % OBJETOS | NIVEL % ELEMENTOS % ENEMIGOS %
OBJETOS

ELEMENTOS ::= ELEMENTO | ELEMENTOS ; ELEMENTO | PLATFORM - ELEMENTOS - ENEMI-
GOS - OBJETOS
ELEMENTO ::= TUBE | CANNON | GAP | TUBE_FLOOR

TUBE ::= tube CUATRO_ARGUMENTOS
PLATFORM ::= platform CUATRO_ARGUMENTOS
CANNON ::= cannon CUATRO_ARGUMENTOS
GAP ::= gap DOS_ARGUMENTOS
TUBE_FLOOR ::= tube_floor CINCO_ARGUMENTOS


ENEMIGOS ::= ENEMIGO | ENEMIGOS ; ENEMIGO
ENEMIGO ::= < NUMERO , NUMERO , TIPO_ENEMIGO , CON_ALAS >

TIPO_ENEMIGO ::= armored_turtle | chomp_flower | goompa | green_turtle | red_-
turtle
CON_ALAS ::= true | false

OBJETOS ::= OBJETO | OBJETOS ; OBJETO
OBJETO ::= MONEDAS | BLOQUE_POWER | BLOQUE | BLOQUE_MONEDAS

MONEDAS ::= monedas CUATRO_ARGUMENTOS
BLOQUE ::= bloque TRES_ARGUMENTOS
BLOQUE_POWER ::= bloque_power DOS_ARGUMENTOS
BLOQUE_MONEDAS ::= bloque_monedas TRES_ARGUMENTOS


DOS_ARGUMENTOS ::= , NUMERO , NUMERO
TRES_ARGUMENTOS ::= , NUMERO , NUMERO , NUMERO
CUATRO_ARGUMENTOS ::= , NUMERO , NUMERO , NUMERO , NUMERO
CINCO_ARGUMENTOS ::= , NUMERO , NUMERO , NUMERO , NUMERO , NUMERO

NUMERO ::= CIFRA CIFRA
CIFRA ::= 0 | 1 | 2 | 3 | 4 | 5 | 6| 7 | 8 | 9
```

Que produce individuos codificados que me generan niveles. Un ejemplo:

```
platform , 0 9 , 0 8 , 8 6 , 9 6 - tube , 2 1 , 8 1 , 5 6 , 0 7 ; tube , 8 9 , 9 0 , 1
0 , 4 1 - < 7 2 , 2 4 , red_turtle , false > - bloque_monedas , 7 5 , 5 8 , 7 4

 < 0 5 , 9 8 , chomp_flower , true > ; < 5 8 , 4 4 , goompa , true >
 monedas , 0 7 , 4 7 , 7 2 , 0 9

 platform , 1 2 , 0 9 , 6 9 , 8 5 - platform , 4 2 , 8 6 , 2 3 , 4 7 - platform , 2
6 , 4 3 , 6 8 , 4 7 - tube_floor , 5 1 , 0 9 , 1 6 , 2 5 , 6 7 - < 0 4 , 8 9 , chom-
p_flower , true > ; < 1 8 , 9 5 , chomp_flower , false > - monedas , 2 5 , 1 2 , 4 5 ,
6 1 - < 9 4 , 7 7 , red_turtle , true > ; < 1 7 , 0 8 , goompa , false > ; < 8 3 , 4 1
, green_turtle , false > ; < 8 1 , 6 0 , red_turtle , true > ; < 7 7 , 6 3 , green_-
turtle , false > ; < 0 4 , 1 4 , goompa , true > - monedas , 6 4 , 9 8 , 4 7 , 8 6 - <
2 2 , 5 5 , chomp_flower , false > - monedas , 1 5 , 2 6 , 0 5 , 7 1 ; tube_floor , 7
7 , 9 3 , 1 2 , 3 0 , 0 3 ; tube_floor , 2 9 , 6 5 , 5 1 , 5 4 , 9 6
 < 9 8 , 0 4 , goompa , true >

 monedas , 8 7 , 3 5 , 2 4 , 0 4 ; bloque , 8 2 , 2 1 , 2 0 ; bloque_monedas , 2 8 , 9
7 , 2 1 ; bloque_power , 2 3 , 5 8 ; monedas , 0 6 , 7 5 , 5 6 , 6 7 ; bloque_power ,
8 1 , 8 1 ; bloque_monedas , 9 8 , 2 3 , 4 5 ; monedas , 4 5 , 9 3 , 7 3 , 9 9 ; blo-
que_monedas , 9 8 , 2 3 , 4 5 ; monedas , 6 6 , 6 4 , 8 6 , 6 9
 tube_floor , 0 1 , 3 1 , 3 7 , 6 4 , 7 3

 < 9 4 , 7 7 , red_turtle , true > ; < 1 7 , 0 8 , goompa , false >

 monedas , 8 5 , 9 2 , 5 0 , 2 2 ; bloque_power , 5 9 , 6 4 ; bloque , 6 7 , 3 4 , 2 1
; bloque_power , 0 2 , 3 9 ; bloque_power , 0 2 , 7 2
```

Tiene las siguientes características:

- Los números dados que corresponden a coordenadas siempre son entre 0 y 99 y representan la proporción del valor que pueda tomar en el momento de la creación del escenario. De esta forma evitamos crear un tipo de numero dependiendo del limite de cada valor.
- Formato enemigos: < x , y , tipo_enemigo , con_alas >
- Formato de elementos: tipo_elemento ARGUMENTOS
- En función de cada tipo de elemento necesita unos argumentos u otros, siempre estarán separados por comas.
- Hay la opción de que salga una plataforma y elementos y objetos nuevos sobre ella o una plataforma sin más.
- Elementos: tubo, tubo con flor, cañón, plataforma y hueco.
- Objetos: monedas, bloques de monedas, bloques varios y bloques especiales.
- Enemigos: tortugas de diferentes colores, flores y goompa. Todos ellos pueden tener alas o no.
- Un nivel puede estar formado por elementos objetos y enemigos, o por un nivel + elementos objetos y enemigos, de esta forma recursiva se consigue generar niveles más largos. A su vez, los elementos, objetos y enemigos pueden estar formado por

uno o varios. Entonces lo que hacemos es separarlos entre distintos tipos por % y entre mismo tipo por ; para que luego al pasearlos nos sea mas fácil.

- He limitado la altura de los objetos y la anchura de los huecos al máximo que Mario puede alcanzar para que todos los niveles sean jugables en ese sentido. Por ejemplo, si me sale que la anchura de un hueco sea 99, lo convertiremos al máximo valor que puede tomar un hueco que es de 5.

Para evaluar los niveles hemos calculado las siguientes características:

- Puntuación/Lenidad: cada enemigo resta 1, cada power suma 1, cada moneda suma 0,1, hueco y cañón restan 0,5. Una puntuación de 0 será optima para un nivel igualado.
- Linealidad: es media del valor absoluto de la diferencia respecto a la altura media del nivel de cada objeto del recorrido.
- Densidad: es la media de las densidades de cada punto del recorrido. Así un punto en el que haya 4 objetos (o partes de un objeto) da igual el tipo que sean, tendrá densidad 4. A menor densidad mejor.
- Numero de elementos. A mayor numero de elementos mejor, pero tampoco es bueno que sea excesivamente alto.

Con esas 4 características, utilizo la siguiente formula para calcular el fitness:

$$Fitness = \frac{altura\ media + n^o\ elementos}{linealidad + densidad + puntuacion}$$

Luego en el algoritmo evolutivo creo una población de 200 individuos. Utilizo el método de cruce de Whigham y un porcentaje de mutación de 3. En cada generación se renuevan 3/4 + 1 de la población.

El fitness medio optimo lo he fijado en 300 y cuando lo alcance terminara el algoritmo y se ejecutara el juego con el mejor nivel generado.

Después de ejecutar varias veces genera niveles jugables por lo general.