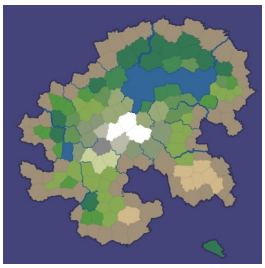
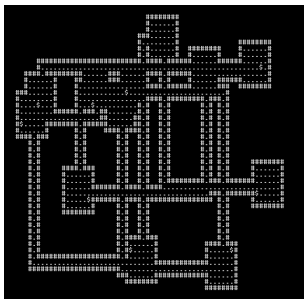
Procedural Generation
para la creación de niveles

Primera parte – Reglas Gramáticas para la creación de palabras.

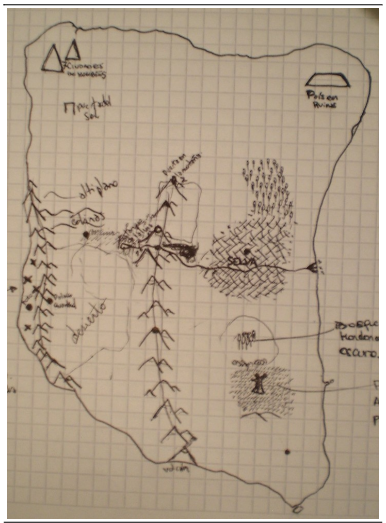
> En esta primera parte explico como llegué a conocer la temática de la creación de palabras aleatorias y cómo se pueden utilizar en el desarrollo de videojuegos para la creación “infinita” de nombres de personajes, lugares, etc.



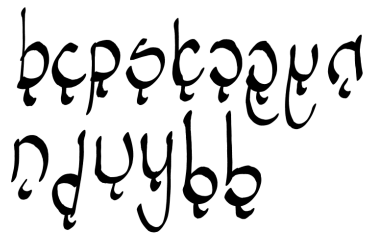


Hace unos años me surgió la idea de escribir historias y relatos fantásticos. Escribí un par de cuentos cortos con personajes como magos, brujos, guerreros de tierras lejanas, etc. Pasado un tiempo me propuse juntar estos relatos bajo un solo universo. Así nació “Anmer”, un continente con una historia muy rica y extensa, llena de bestias y personajes buenos y malos. Escribí la idea general para las diferentes épocas de “Anmer” (etapas temporales del continente en donde iban a encajar los relatos que ya había escrito).

Pero, ¿qué tiene que ver esto con la creación de niveles usando algoritmos procedurales?



Bueno, como venía diciendo empecé con la creación de un universo que encapsulara todos los relatos. Para esto fue necesario diseñar (en mi cabeza) el concepto general de este mundo. Fui respondiendo a preguntas como: qué tamaño tiene, qué tan antiguo es, qué personajes habitan allí, cuál es la historia general de esta tierra, etc. Tenía mas o menos todo resuelto, pero había algo que me picaba la curiosidad: ¿qué lenguas hablaban en “Anmer”?
Me vino a la mente la idea de Tolkien, que usó sus propios lenguajes como marco para todas su obras. Las historias aparecían después de la lengua, y no al revés. Muchas veces inventaba las palabras para los nombres y lugares y luego a partir de su significado le descubría su lugar en la historia.
Yo nunca había inventado ningún lenguaje, mucho menos sabía como empezar a crear uno.
Google! En internet existe material para todo, si sabés cómo buscar.



Investigué un tiempo sobre la creación de lenguajes artificiales “Conlangs”. Aprendí mucho sobre cómo funcionan las lenguas en general y no solo mi propia lengua materna (español). Cuando creí haber juntado un mínimo de información sobre la creación de los lenguajes artificiales me tiré de lleno a la idea.
Primero cree una forma escrita para la lengua: un alfabeto de pocos símbolos que me servía para escribir en español e inglés (y con suerte también en la lengua que quería crear). Comencé a experimentar con diferentes sonidos y creé un par de palabras, ajustando al mismo tiempo la pronunciación y la forma escrita.
En ese momento se me ocurrió la idea de automatizar la creación de nuevas palabras, nombres y frases. Como me gusta programar pensé que se podía hacer algo para obtener 1000 palabras con un solo click.

Me encontré con que esto ya se había hecho antes, y muchas veces. En internet hay docenas de sitios que permiten generar nombres aleatoriamente, entre ellos http://fantasynamegenerators.com/ (generador de nombres fácil de usar), http://klh.karinoyo.com/generate/words/ (generador de palabras utilizando reglas gramaticales)

Investigando un poco más me topé con un sitio que explicaba como se realizaban estos generadores supuestamente “aleatorios” de nombres.
Probé varias formas para generar texto “aleatorio”, entre ellas las Cadenas de Markov fueron las más interesantes. Pero lo que me terminó interesando más fue el concepto de grámaticas.
Este es un concepto muy parecido a lo que se utiliza en lenguajes de programación para la verificación de la sintaxis y semántica, aunque no lo sabía en ese entonces.

Gramáticas

La idea es crear las palabras o el texto de acuerdo a un conjunto predeterminado de reglas. Las palabras son el resultado de la combinación de estas reglas. Por ejemplo, para generar un nombre corto podríamos tener estas reglas:

```
[NOMBRE] = [C][V][C][V]
[C]
[C] = v | c | d | r | t
[V] = a | e | i | o
```

En este esquema tenemos 3 reglas: regla principal (NOMBRE), consonantes (C) y vocales (V)

- La regla NOMBRE está definida en función de otras reglas de tipo C y V.
- Las reglas C y V no utilizan otras reglas, pero tienen asignados todos los posibles valores que podrían tomar, en este caso C solo podría ser una de las letras: v, c, d, r, t; y V puede tomar valores: a, e, i, o. El valor que toma C y V se decide aleatoriamente de acuerdo a los posibles valores.

Notar que el símbolo | (pipe) se utiliza para representar una alternativa entre las diferentes opciones.

Para generar un nuevo nombre entonces utilizamos este esquema simple con nuestras tres reglas:

1. Primero se elige una regla para comenzar (en nuestro caso es la regla NOMBRE) y se evalúa y resuelve recursivamente.
2. Evaluamos el primer elemento de la regla NOMBRE , que es [C].
   C es también es una regla, por lo tanto la resolvemos. Sabemos que C solo puede tomar uno de los valores indicados (v, c, d, r, t) así que elegimos uno al azar: este es el primer carácter de nuestro nombre.
3. Si seguimos así con los demás elementos de la regla NOMBRE completamos toda la palabra.

Posibles resoluciones para la regla NOMBRE: “Roder”, “Darod”, “Vader”, etc.

Este es un ejemplo muy simple, pero la generación de palabras utilizando esta técnica puede hacerse mucho más compleja y con resultados mucho más interesantes ampliando la cantidad de reglas, combinando diferentes reglas entre sí o incluso jugando con la probabilidad para la toma de los valores finales.

Podríamos tener un nombre más complejo utilizando otro esquema de reglas:

```
[NOMBRE] = [C | V] [V] [C][FIN]
[FIN] = [FIN-MUJER] | [FIN-HOMBRE (2)]
[FIN-MUJER] = al | ia | ela
[FIN-HOMBRE] = oir | al | il
[C] = r | l | t | d | m | s
[V] = a | i | y | hu | o
```

Posibles nombres generados a partir de este esquema: “Raloir”, “Dytela”, “Huilia”, etc.

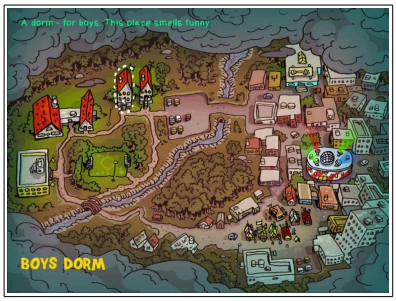
Notar que en este caso el primer elemento del nombre puede ser una consonante o una vocal. Además agregamos una terminación para diferenciar entre nombres masculinos y femeninos. Existe además doble chance de que la terminación sea para un hombre.

Todo esto es muy lindo y tiene miles de usos para la generación de nombres de personajes y lugares, pero ¿cómo lo implementamos? Después de todo si queremos utilizarlo en nuestro juego vamos a necesitar informatizarlo de algún modo.

En mi caso realicé un programa en Java que me permitiera utilizar estas reglas.
Los esquemas de reglas estaban escritos dentro de archivos de texto. Estos archivos los cargaba en mi programa: utilizando expresiones regulares se identificaban las reglas y los operandos. Luego se evaluaban las reglas N cantidad de veces para obtener la lista de palabras finales.

Esto funcionó extremadamente bien, pero por desgracia las palabras generadas por el programa no me terminaron de convencer como para incluirlas en mi lenguaje.

Pero para incluirlas en mi videojuego no estaban nada mal! Ahora podía tener un personaje diferente en cada partida y los mapas presentaban nombres diferentes para cada jugador. Utilizando esta técnica junto con otras más se puede conseguir un nivel de aleatoridad que le da mucha rejugabilidad a nuestro juego.



Bueno esto fue todo por ahora. Si quieren probar el generador de nombres que hice en Java pueden hacer clic aquí para descargar el proyecto.
Pueden probar todas las técnicas que usé para generar nombres, desde cadenas de Markov hasta Reglas Gramáticas.

En la segunda parte explicaré como aplicar este concepto de gramáticas para generar no solo palabras sino cualquier tipo de contenido dentro de nuestro juego.
Además pondremos a prueba todo esto implementando un generador de niveles dentro del motor Godot Engine.

Saludos, y hasta la próxima.