PROYECTO FINAL TEORIA MAC MINECRAFT

Alfonso José Rodríguez Gómez

ÍNDICE

1. Introducción
    a. Presentación Practica
    b. Que es Maxscript
    c. Que es Minecraft
2. Proyecto
    a. Terreno
    b. Montañas
    c. Arboles
    d. Iluminación
    e. Animales
        i. Cerdo
        ii. Vaca
        iii. Oveja
    f. Personas/Monstruos
        i. Persona
        ii. Zombie
        iii. Ender
        iv. Creeper
        v. Dragón
    g. Análisis de interfaz
3. Conclusión

1. Introducción

a. Presentación Practica

El objetivo de esta práctica es hacer un plugin mediante la herramienta maxscript, que proporciona el programa AutoDesk3DMax Studio, que permita crear rápidamente un mundo de minecraft. El mundo tiene que ser personalizable por medio de los parámetros ofrecidos en el plugin. Para realizar esta práctica contábamos con el GeneradordeCastillos del año pasado y con la ayuda de MaxScript.

b. ¿Qué es Maxscript?

MAXScript es un lenguaje de alto nivel orientado a objetos y de sintaxis muy sencilla. Está pensado para usuarios de 3D Studio MAX que no estén familiarizados con la programación. Sus variables son sin tipo y no hay distinción entre mayúsculas y minúsculas.

Este lenguaje no necesita compilador. Todas las instrucciones se llevan a cabo en tiempo de ejecución.
Podemos utilizar MAXScript y 3D Studio MAX al mismo tiempo. Se puede crear un cubo en MAXScript y seguidamente, modificarlo en 3D Studio.
Algunos ejemplos de utilización de este lenguaje pueden ser:
Creación de escenas complejas que interactivamente serían demasiado complicadas de llevar a cabo.
Creación de herramientas de proceso por lotes.
Generación de funciones nuevas para facilitar una determinada tarea.
Uso de casi todas las funciones de 3D Studio MAX.
Creación de nuevos paneles y ventanas con utilidades para 3D Studio MAX.

c. ¿Qué es Minecraft?

Minecraft es un videojuego independiente de estilo mundo abierto escrito en Java, creado originalmente por Markus "Notch" Persson, y ahora por su compañía Mojang AB. Está sumamente inspirado en Infiniminer.

Desde su creación, Minecraft fue desarrollado exclusivamente por Notch hasta que Jens "Jeb" Bergensten empezó a trabajar con él, y se convirtió en el director de su desarrollo. La música del juego fue compuesta por Daniel "C418" Rosenfeld, y las pinturas de los cuadros fueron diseñadas por Kristoffer Zetterstrand. Inicialmente lanzado como Minecraft Clásico el 17 de mayo de 2009, luego fueron lanzadas las versiones a partir de la 1.0 el 18 de noviembre de 2011. A partir de entonces, Minecraft se expandió hacia teléfonos, tabletas y consolas.

2. Proyecto

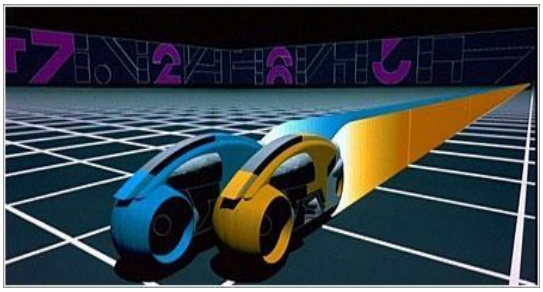
a. Terreno

Para realizar el terreno tuvimos diferentes ideas, como crear un plano y aumentarle el número de *lengthsegs* y *widthsegs* para después añadirle el modificar *edit poly* y desplazar arriba y abajo los puntos generados por las intersecciones . Después creamos una matriz de cubos con dependencia sobre el plano que hemos creado , de esta manera al mover los puntos del plano los cubos asociados se moverán de forma consecuente , de esta manera al aplicar cualquier modificar al plano del tipo wave o noise el cubo se moverá de acuerdo a la nueva posición el del punto del plano.

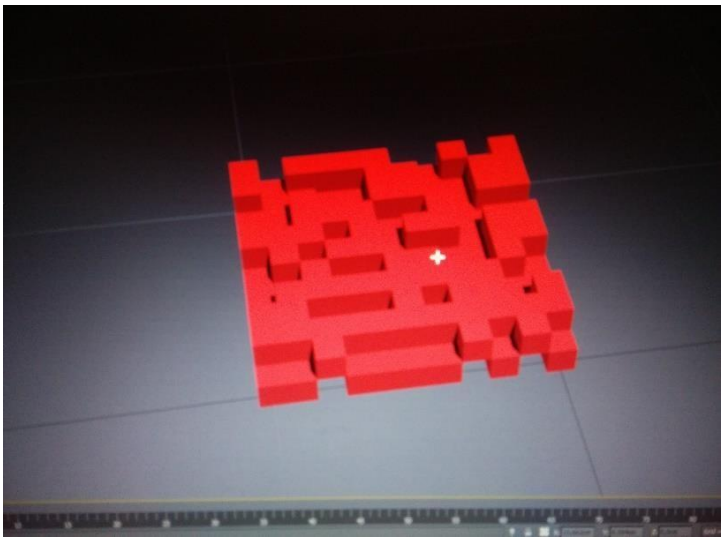
Otro camino sobre el que barajamos crear el terreno el algoritmo de ruido utilizado en el videojuego original Minecraft para crear las ondulaciones en los terreno, nos estamos refiriendo a "Perlin Noise " o traducido al español el ruido Perlin.

El ruido Perlin es un algoritmo ampliamente utilizado para la generación de texturas procedurales. Debe su nombre a Ken Perlin, que lo introdujo en los años 80 en los efectos especiales de la película Tron. Desde entonces se ha extendido su uso llegando a convertirse en un estándar. Basándonos en este algoritmo y sus variaciones podemos generar texturas que representan casi cualquier cosa de las que podemos encontramos en la naturaleza, desde nubes a terrenos.

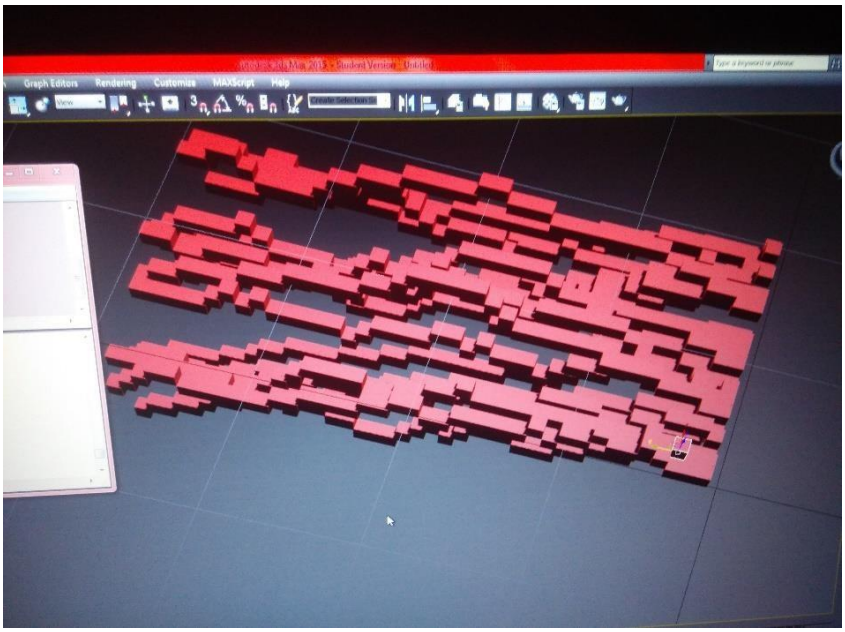
En la imagen que ponemos a continuación podemos ver una imagen del ruido Perlin.



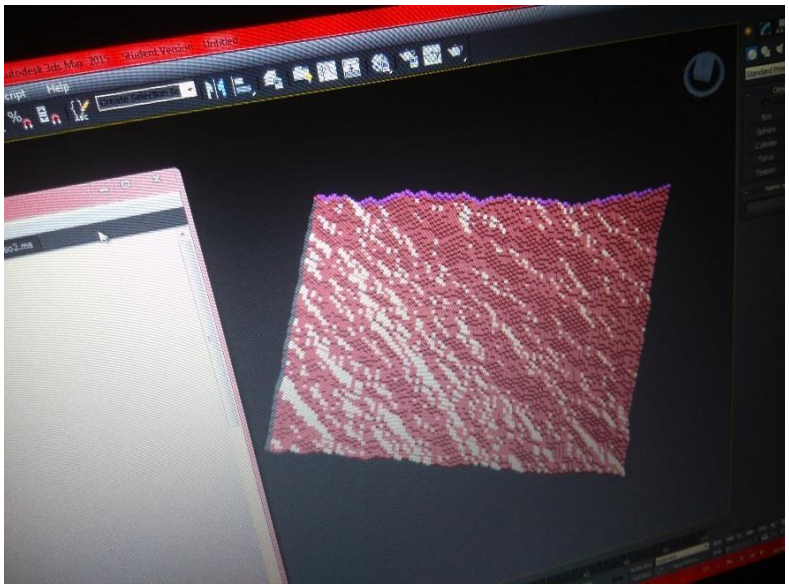
Esto nos dio la idea de implementar ese algoritmo de forma que nos devolviese las alturas de los cubos para poder realizar el terreno. Intentamos implementarlo en nuestro código de MaxScript pero no conseguimos adaptarlo. De manera que pensamos otra idea para realizar el terreno de este trabajo. Decidimos crear un algoritmo similar al Ruido Perlin. Estos fueron los primeros terrenos que construimos.



La idea de la fotografía anterior era crear un valor aleatorio entre 0 y 1 para interpolar los valores, de forma que algunos cubos se desplacen hacia abajo. Como podemos observar en la imagen se aprecian muchas irregularidades, pero ya tuvimos algo con lo que poder seguir programando código para realizar el terreno. A partir de aquí tuvimos la idea de que la altura de los cubos dependiese de los cubos vecinos.



Como vemos en la anterior imagen vemos cómo se van creando las filas y cada cubo depende del cubo anterior. Pero las columnas no están relacionadas entre sí.



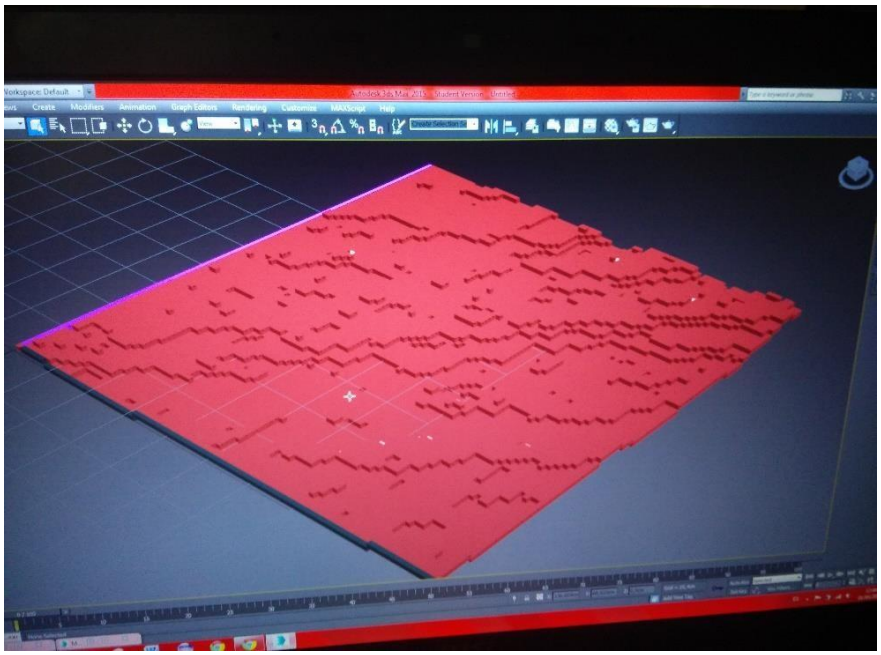
En la imagen anterior vemos como el terreno está construido de tal manera que cada cubo depende del anterior y del vecino de la izquierda. Al principio creamos una fila de cubos en la que el primer cubo se colocaba en la posición 1, 0 o -1 dependiendo del valor aleatorio y su posterior interpolación. Después los siguientes cubos de esa misma fila dependían del anterior. Después creábamos una columna de forma perpendicular a la primera fila para poder trabajar con las columnas. De nuevo esta fila depende de su vecino anterior y, como consecuencia del primer cubo creado. A partir de aquí creábamos fila tras fila para colocar correctamente el terreno. En la imagen se puede observar que hay cubos rojos y blancos, los blancos son forzados en el código para poder realizar un terreno sin huecos. Para crear cubos teníamos diferentes combinaciones:

-Cubo anterior y cubo vecino izquierda tenga la misma posición de forma que el cubo nuevo tenga 3 posibilidades para colocarse, 1, 0 y -1.

-Cubo anterior y cubo vecino izquierda tenga una diferencia de posición en Z de 1, de forma que el nuevo cubo solo pueda posicionarse en dos posibilidades.

-Cubo anterior y cubo vecino izquierda tengan una diferencia de dos en sus altura, de forma que obligásemos al algoritmo a posicionar el cubo nuevo en la posición Z resultante de hacer la media de las dos alturas de los cubos.

Cuando interpolarízamos el número aleatorio lo hacíamos de forma que cada probabilidad de desfase del terreno tuviese 33.33%, esto fue un error ya que el terreno aparecía de forma muy escarpada, hasta que un día decidimos dar más posibilidades a que el desfase fuese 0 de forma que el suelo sea más plano pero que tenga algunas irregularidades. A continuación veremos el terreno sin material bien construido.

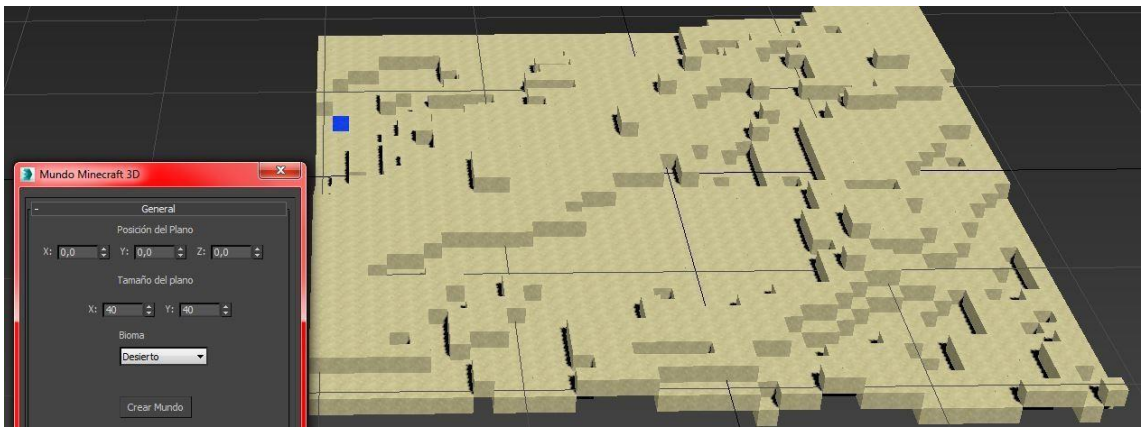


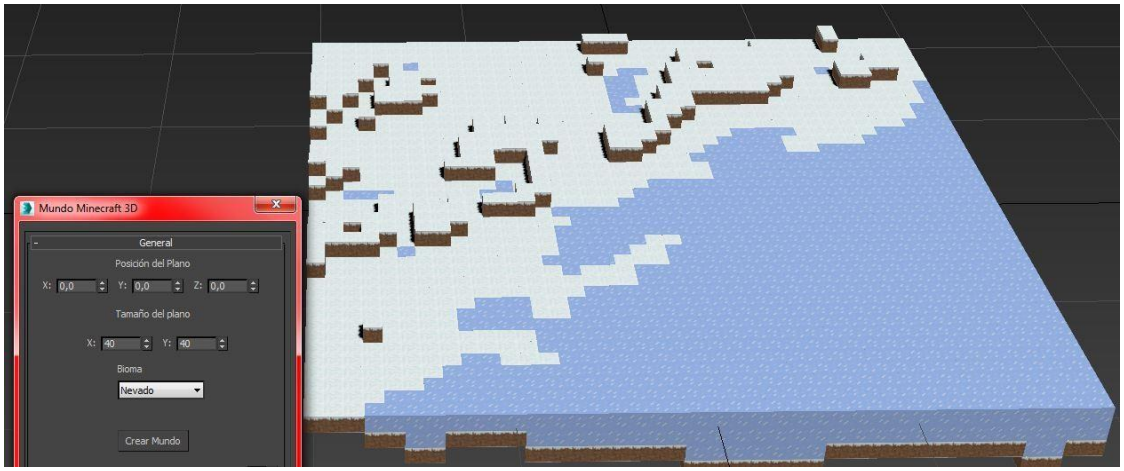
Aquí vemos el terreno de una forma plana pero con pequeñas irregularidades de forma similar a Minecraft.

Después de tener el terreno bien construido pasamos a crear el agua, de una forma aleatoria y según el terreno previamente creado. Para ello calculábamos la posición mínima del terreno para colocar el agua allí. En algunos terrenos solo había un cubo con la posición más baja, asique decidimos hacer dos niveles de agua. Los cuales cubrirían los cubos con las dos alturas más bajas. Al ejecutarlo dos veces encontramos un pequeño problema. Éste trataba que encima del nivel minimo de agua no se creaba correctamente el cubo acuoso de forma que tuvimos que arreglarlo para que el agua se colocase de forma uniforme y no poseyera socavones que serían antinaturales. A partir de aquí decidimos añadir un tercer nivel ya que sabíamos hacerlo.

Después de conseguir el agua y el terreno pensamos en añadirle materiales. Para ello pensamos la forma de añadírselos convirtiendo en *edit poly* y seleccionar por cara y de alguna forma cargar desde el script y asignarle el color. Esta forma no conseguimos implantarla ya que no conseguimos seleccionar por *Face* el cubo. Después conseguimos hallar la forma de poner materiales en la escena mediante multimaterial, esto no ayudó ya que nosotros decíamos al material cuantas caras queríamos que tuviese, en nuestro caso trabajábamos con cubo de forma que necesitábamos 6 caras. Una vez creado el multimaterial debíamos crear las texturas cargándolas desde una imagen. El siguiente paso era trasladarla a un *bitmap*. Una vez conseguido haríamos un bucle para asignar a cada cara del cubo el material. Si es una cabra su cara estaría en la posición 3, en cambio sí es un cubo de terreno su posición *top* o arriba será uno.

Después de crear correctamente los materiales decidimos crear diferentes biomas, tanto el normal como el nevado y el desértico. Cada uno de ellos tiene un agua asociado, para el bioma normal y desértico es el mismo salvo que el desértico posee menos niveles de agua para que se asemeje a la realidad. En cambio para el material nevado sus niveles de agua son agua congelada.



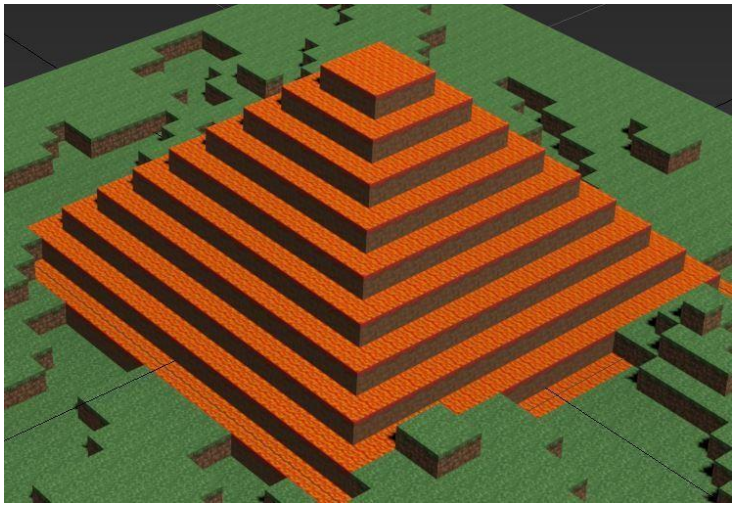


Como vemos en la última imagen se crea un surco con dos niveles de aguas, esto simula las profundidades del océano para dar más realismo a este mundo virtual.

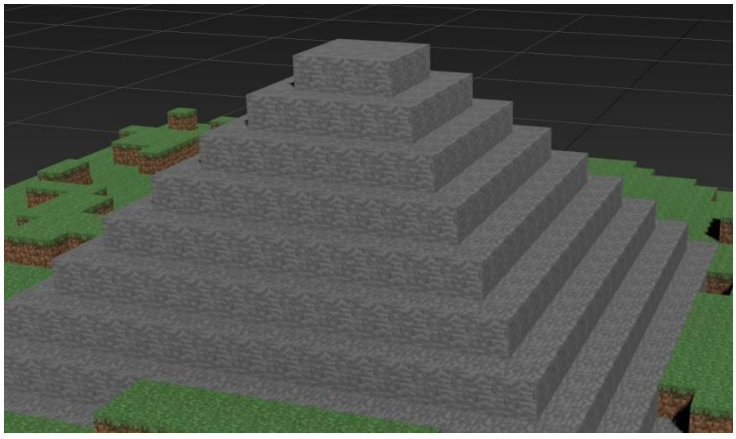
Actualmente el usuario puede decidir tanto el tamaño de los lados del mundo como donde quiera posicionar el mundo en el eje X, Y, Z.

b. Montañas

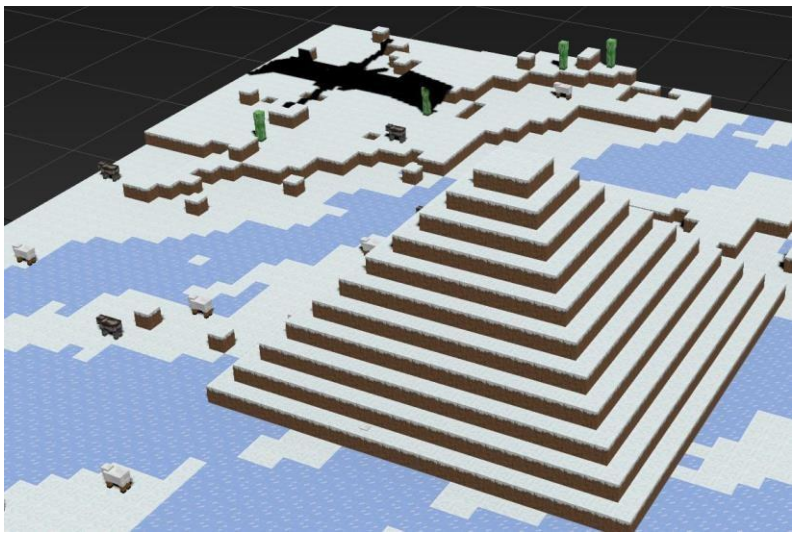
Para crear una montaña en nuestro mundo decidimos crear una pirámide de forma que la punta de ésta no existiera. El usuario en la interfaz tiene la posibilidad de activar o desactivar la montaña e introducir el valor de la altura de la montaña.



Aquí vemos una montaña de lava que simula un volcán



Para esta montaña usamos materiales de roca



Esto es una pirámide con materiales congelados junto con la sombra del Dragon del fin

c. Arboles

Los árboles que hemos introducido se basa en el propio juego Minecraft , de esta manera, nuestro arboles dispone de una serie de parámetros que podemos elegir siempre respetando un margen .
De esta manera encontramos que tanto la altura del árbol como el follaje del propio pueden ser elegidos entre varias opciones que son; estándar, nevado y por último otoños (marrón).
En nuestro script podemos elegir el numero máximos de árboles que pueden aparecen, teniendo en cuenta que hay diferentes biomas cabe resaltar que en el terreno desierto en cambio de crearse arboles estándar se crean cactus.
Sobre la creación del objeto árbol cabe destaca que las etapas de diseño fueron parecidas a las de la creación de animales.

d. Iluminación(cambio)

Uno de los extras que hemos incluido es el de incluir una serie de luces para darle un mayor ambientación a nuestro mundo .De esta manera encontramos dos estado de la luz.
En un principio consideramos que la luz del día sería un indicador para que los monstruos (zombi, creeper...) no pudiesen aparecen en la escena y de esta manera durante el día aparecería los personajes pacíficos como animales y seres humanos (persona) –
Por otro lado encontramos en la luz de noche, o mejor el estado nocturno, que únicamente los monstruos aparecerían durante este tiempo siendo el caso en que los animales los que no apareciesen emulando el rutina del videojuego.

Para la iluminación que decidimos necesitábamos dos tipos, una a mediodía y otra por la noche, para crear la iluminación se necesita el siguiente código. Aquí

simplemente estamos creando la bombilla, pero necesitamos también cambiar la intensidad y el color, si no se añadirá con los valores por defecto

e. Animales

Decidimos crear una serie de animales para el mundo de Minecraft: oveja, cerdo, vaca.

Los animales es lo más sencillo de hacer, al principio creamos los animales con posiciones absolutas, pero mediante unas variables podemos hacer que se coloquen aleatoriamente en el mundo. Para que aparezcan los animales hay que seleccionar que animales queremos que aparezcan.
El código es similar en todos los animales, esto por ejemplo seria para crear una pata de la oveja.

```
matCaraCabra=multimaterial numsubs: 6
TexturaCaraCabra=sysInfo.currentdir + "\\materiales\caracabra.jpg"
bitmapCaraCabra=bitmaptexture filename: TexturaCaraCabra
```

Para insertar el material en el animal lo primero que hacemos es dividir el cubo en 6 caras, luego introducimos la textura.

```
for p=1 to 6 do(
if p==3 then(
matCaraCabra[p].diffusemap=bitmapCaraCabra
matCaraCabra[p].showInViewport=on
)else(
matCuerpoCabra[p].diffusemap=bitmapCuerpoCabra
matCuerpoCabra[p].showInViewport=on
)
)
for k=1 to 6 do(
matCuerpoCabra[k].diffusemap=bitmapCuerpoCabra
matCuerpoCabra[k].diffusemap=bitmapCuerpoCabra
)
```

Y por si nos hace falta redimensionar o mover más fácilmente agrupamos todos los elementos.
Los animales que creamos fueron;

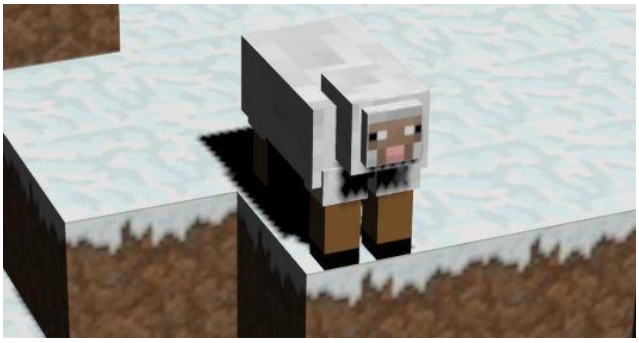
i. Cerdo

Los cerdos son criaturas pasivas que aparecieron por primera vez en el modo Prueba de supervivencia. La principal ventaja de los cerdos es la de proporcionar carne de cerdo al jugador, y por eso son una fuente no-renovable de alimentos (aunque se pueden reproducir para que dicha fuente se convierta en renovable

ii. Vaca

Las vacas son criaturas pasivas, y fueron añadidas por primera vez en Indev. Su mayor valor es como una fuente de cuero, que puede ser usado para crear la armadura más débil de los 5 tipos existentes.

iii. Oveja

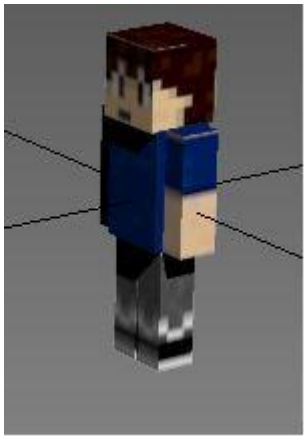


Las ovejas son criaturas pasivas y fueron las primeras en ser añadidas al juego, en la versión Survival Test.
La principal función de las ovejas es proporcionar lana.

f. Personas/Monstruos

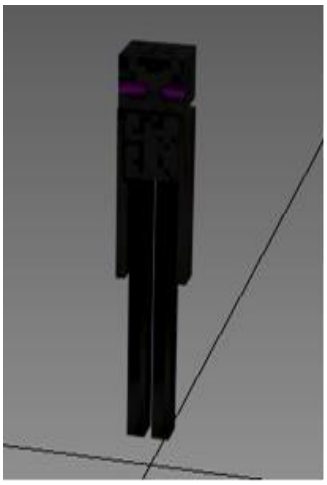
Una vez terminado los animales, al disponer de tiempo barajamos la posibilidad de introducir otros elementos. Los elementos que finalmente introducimos en el siguiente;

i. Jugador



El **jugador** es el personaje controlado a la hora de jugar a Minecraft. El jugador predeterminado se conoce generalmente como Steve, bautizado por Notch a modo de broma, e intenta ser genérico representando un humano masculino. Sin embargo, el aspecto del jugador puede modificarse en cualquier momento. En este caso Steve lleva el skin de Normal Boy , el cual se puede obtener en ;
http://www.minecraftskins.com/search/skin/normal/1/

ii. EnderMan



Un **Enderman** (a veces llamado simplemente Ender) es una criatura de aspecto oscuro, tres bloques de altura, con largos brazos y piernas. Sus ojos son brillantes y de color púrpura. También posee una particular aura semejante a las de los portales, como si cayesen flores rosas a su alrededor.
Cabe destacar sobre Enderman que cuando se posiciona con la montaña presente se colocara en la cima, mientras en otro caso el Enderman se pondrá de forma aleatoria sobre el terreno.

iii. Creeper

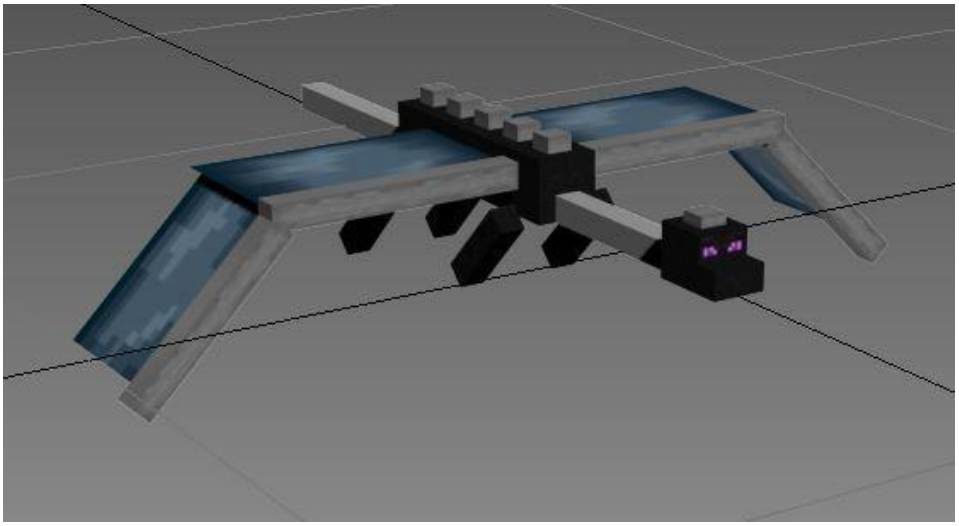
El **creeper** es una criatura hostil que explota cuando se acerca a un jugador.

iv. Zombi

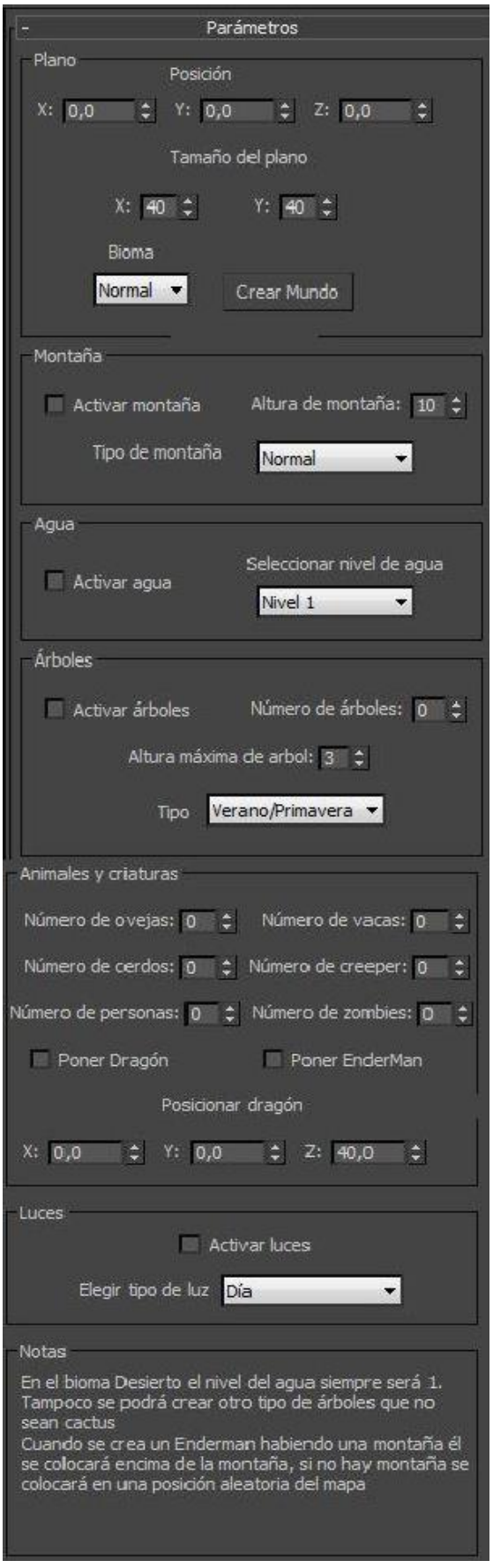
Los zombis son una de las criaturas hostiles más comunes del juego.

v. Dragon del fin

El Dragón del Fin es el primer jefe oficial del juego. Sólo se genera de forma natural en El Fin. Esta criatura no utiliza el modelo de dragón que creó Notch, sino que utiliza una textura que recuerda más a un Enderman, ya que es negro y escamoso, con los ojos morados.



g. Análisis de interfaz



Posición del plano: En este apartado aparecen valores con la posición del plano, el tamaño del plano, el bioma que puede elegir el usuario (los cuales son; normal, invierno, desierto).

Montaña: En nuestro caso únicamente se dispone de un montaña que se coloca en el medio del mapa, de esta montaña podemos elegir el tipo de montaña; normal, volcán, rocas, nevado o desértico.

Agua: Nuestro terreno se crea de forma aleatoria y con ello la extensión del agua debe ir ligada a los deseos del usuario .El usuario puede elegir cuantos niveles de profundidad desea.

Arboles: Activado árboles, acto podremos elegir el número de árboles, así como parámetros que podemos modificar como el tipo de follaje y su longitud.

Animales y Criaturas: En este apartado podemos elegir que numero de animales queremos que aparezca en la escena .Los animales sobre los que tenemos control son ; ovejas , vacas , cerdos , creeper….La forma que aparecen los animales puede ser de dos formas , una introduciendo el número que el usuario desea y la otra dándole al check (el Ender).

Luces: en este apartado podemos elegir el tipo de luces que queremos introducir en la escena, las luces de las que disponemos son; día y noche.

Notas: Hemos puesto un apartado donde se consideran puntos importantes.