UNIVERSIDAD CARLOS III DE MADRID

GRADO EN INGENIERÍA DE SISTEMAS AUDIOVISUALES

CURSO 2016-2017

PROYECTO DE FIN DE GRADO

VISUALIZACIÓN DEL FCA EN MINECRAFT

ALUMNO: VÍCTOR GARCÍA FERNÁNDEZ

TUTORES: CARMEN PELÁEZ MORENO
FRANCISCO JOSÉ VALVERDE ALBACETE

Agradecimientos:

A toda mi familia y amigos por no cesar en su apoyo
y a mis tutores Carmen y Francisco José por la ayuda

Índice general

Índice de ilustraciones

Capítulo 1: Introducción

1.1: Definición del proyecto

El objetivo de este proyecto es realizar una visualización tridimensional para el análisis de conceptos formales (o FCA, de las siglas en inglés de Formal Concept Analysis). En el presente documento, se llevará a cabo un desarrollo de cuáles han sido los puntos a tratar para la realización el proyecto, así como una explicación de la naturaleza de los propios conceptos formales.

El conjunto de conceptos formales dentro de un mismo contexto (llamado contexto formal) tiene la estructura de un retículo, que será la estructura a representar en este proyecto. Más adelante se realizará una explicación mas detallada de la teoría de retículos así como su relación con el FCA.

Para el desarrollo de la visualización, se ha procedido a realizar una extensión o modificación "mod", como lo denominaremos a continuación, de la plataforma Minecraft aprovechando la capacidad de libre creación y el mundo abierto de dicha plataforma. Otro de los motivos por los que utilizar Minecraft como soporte para la visualización de conceptos formales es debido al carácter multiplataforma del programa, lo que da a lugar a que el mod pueda ser instalado en multitud de máquinas de diferentes características desde una misma compilación del programa.

El mod de Minecraft se encargará de la visualización de los retículos. Esta visualización será inmersiva, es decir, podremos realizar una exploración por el interior del propio retículo y explorar las diferentes partes del mismo. La generación del retículo conceptual viene dada por el programa "Concept Explorer FX" (Conexp-FX) y mediante una conexión entre los dos programas (Conexp-FX – Minecraft) se transfieren las coordenadas tridimensionales del retículo desde Conexp-FX a la plataforma de Minecraft con el mod instalado. La conexión entre los programas se realizará mediante sockets, por lo que es necesario el uso simultáneo de los dos programas. Los controles para la exploración dentro del entorno de Minecraft son los propios de la plataforma, mediante el teclado y el ratón para el desplazamiento en el interior del contexto.

Conexp-FX es un programa de exploración de conceptos, capaz de generar un modelo de retículo del contexto formal además de permitir una interacción con el usuario. El programa permite tanto generar nuevos contextos formales, como importalos desde el sistema local de archivos. Una vez cargado el contexto formal en el programa se permite la modificación del mismo por el usuario, estas modificaciones se realizan en tiempo real en el programa, por lo que resultó idóneo para la realización de este proyecto.

El lenguaje de programación utilizado es Java para ambos programas. El mod de Minecraft está creado desde cero mientras que para el programa de Conexp-FX ha sido necesarios una serie de cambios en su código fuente, añadiéndole así funcionalidad para poder establecer la conexión Conexp-FX – Minecraft. Que ambas plataformas estén programadas en Java facilita tanto dicha conexión, como otorgarle en carácter multiplataforma a ambos programas, como se mencionó anteriormente.

1.2: Estructura del proyecto:

El proyecto se divide en dos partes principales diferenciadas. Por un lado está la parte de generación del contexto formal y de su retículo asociado, realizado por el programa Conexp-FX; mientras que por el otro se encuentra la plataforma Minecraft con el mod para la representación del retículo. Por lo tanto, el funcionamiento del proyecto en su totalidad será de la siguiente forma:

1. **Conexp-FX**
    - 1.1. Creación o importación del contexto formal.
    - 1.2. Generación del retículo.
    - 1.3. Posibles modificaciones.
    - 1.4. Envío de la información del contexto a la plataforma Minecraft.
2. **Minecraft**
    - 2.1. Recepción de la información del contexto de Conexp-FX
    - 2.2. Análisis de las características del contexto.
    - 2.3. Generación del mundo tridimensional con el retículo representado.
    - 2.4. Exploración inmersiva del retículo.

Una vez realizado este proceso el programa mostrará la interfaz del mundo de Minecraft donde se puede visualizar el retículo y realizar una exploración del contexto formal.

Capítulo 2: Estado del arte:

En éste capítulo se realizará un estudio de la metodología de la que parte el proyecto, que en este caso es el FCA, así como de algunos de los programas utilizados para la visualización de conceptos y los motivos por los que se ha elegido una de esas opciones.

2.1: Análisis en Conceptos formales (FCA).

Se define como Análisis en Conceptos Formales (ing. "*Formal Concept Analysis*") a la teoría que, partiendo de una tabla de objetos y atributos incidentes con otros, es capaz de extraer las relaciones entre los propios datos y estructurarlos en una jerarquía de inclusión. La finalidad de esta estructura es organizar los datos de manera similar al pensamiento humano, mediante una relación de objetos y sus atributos. De este modo se tienen agrupaciones de objetos con atributos (o características) comunes, y a su vez, esos grupos tendrán otras características comunes entre sí, dando lugar a una estructura jerárquica que facilitará su comprensión y estructuración.

En los siguientes apartados se explicarán los términos utilizados en el FCA.

2.1.1: Extensión, intensión y concepto.

Las bases del FVA fueron establecidas por Rudolf Wille en el artículo *Restructuring Lattice Theory - An approach based on hierarchies of concepts* publicado en 1981. En este artículo, Wille presenta la motivación de , no sólo desarrollar el análisis de grandes conjuntos datos desde los fundamentos matemáticos, sino de añadir herramientas para mejorar la comprensión de los mismos desde el pensamiento humano como la teoría de retículos que se explicará más adelante. Como Rudolf Wille afirma en el mencionado artículo:

> "La reestructuración de la teoría de retículos es un intento de volver a fortalecer los lazos con nuestra cultura general, por la vía de que la teoría se interprete de la manera más concreta posible y a través de esto se incentive una comunicación mejor entre los teóricos de los retículos y los potenciales usuarios."

Para ello, Wille se apoya en las categorías de extensión e intensión de la lingüística y la lógica del siguiente modo:

Se define como extensión a cada conjunto de objetos con características comunes, mientras que al grupo de características comunes a un objeto se le denomina intensión. La suma de ambas partes (cada extensión con su correspondiente intensión) forma un "concepto formal", donde el adjetivo "formal" hace referencia a que es el resultado de un planteamiento matemático. Por lo tanto, se puede afirmar que un concepto formal está definido tanto por su intensión como su extensión.

Al conjunto de conceptos dentro de un mismo marco, es decir, que cuyos objetos y atributos están relacionados de una manera en particular, se le denomina contexto formal. Para ilustrar estos conceptos se utilizará el siguiente contexto formal como ejemplo:

	Femenino	**Joven**	**Adulto**	**Masculino**
Chica	X	X		
Mujer	X		X	
Chico		X		X
Hombre			X	X

Donde las filas están indexadas en cada uno de los objetos y las columnas en cada una de los atributos.

Este contexto formal da lugar a los siguientes conceptos formales:

C1: {Chica, Mujer, Chico, Hombre} , { }

C2: {Chica, Mujer} , {Femenino}

C3: {Chica, Chico} , {Joven}

C4: {Mujer, Hombre} , {Adulto}

C5: {Chico, Hombre} , {Masculino}

C6: {Chica}, {Femenino, Joven}

C7: {Mujer}, {Femenino, Adulto}

C8: {Chico}, {Joven, Masculino}

C9: {Hombre}, {Adulto, Masculino}

C10: { }, {Femenino, Joven, Adulto, Masculino}

Donde en el primer corchete se especifican todos los objetos propios de cada concepto, es decir, su extensión. Mientras que en el segundo tenemos las características o intensión.

2.1.2: Subconcepto y superconceptos.

Se define como subconcepto al concepto formal que contiene en su interior al total de la extensión de un segundo concepto formal, idénticamente un concepto es subconcepto de otro si el conjunto de sus características está contenida en el segundo. Mientras que a ese segundo concepto se le denomina superconcepto del primero, es decir, su intensión contiene el total de la intensión del primero.

Así, por ejemplo, volviendo al ejemplo anterior tenemos que el concepto C3 sería subconcepto de C6 o el concepto C7 sería superconcepto de C2.

2.1.2: Retículos.

La relación de conceptos formales tiene una estructura ordenada, donde cada concepto guarda una relación con otro por su intensión o extensión. A este tipo de estructuras se denominan retículos. La representación gráfica de un retículo tiene la forma de un diagrama de Hasse, que es la representación del un conjunto ordenado eliminando la información redundante. Un ejemplo de dicho diagrama se representa en la figura siguiente:

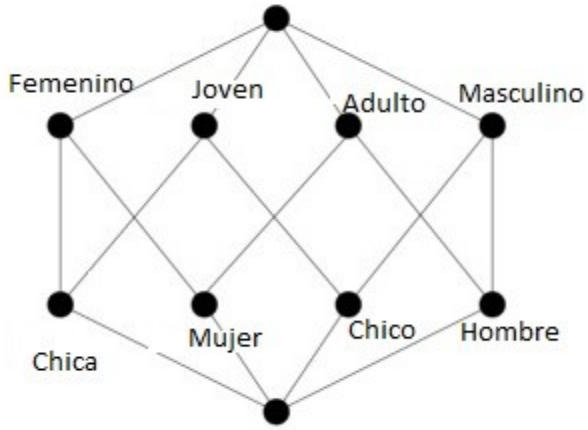

Ilustración 1: Retículo conceptual correspondiente al contexto formal anterior.

En la Ilustración podemos observar la relación de los conceptos formales del contexto formal explicado anteriormente. En general, los retículos obtenidos no son tan "regulares" como el mostrado en la figura, y por ello en necesario usar para su construcción y navegación herramientas adecuadas.

2.1.3: Software de Análisis Formal de Conceptos.

Existen múltiples plataformas para el FCA. Algunas de ellas incluyen una representación gráfica que es el objeto de interés de este proyecto. En este apartado se va a proceder a analizar ciertos ejemplos de estas plataformas, y el motivo por el cual se ha elegido una como soporte para realizar el proyecto.

- **Galicia[1]:**

Este software desarrollado por el Departamento de Informática e Investigación Operativa de la Universidad de Montreal ofrece funciones avanzadas para el manipulado de datos y el procesamiento de varios tipos de contextos formales. Los contextos se pueden cargar desde un archivo almacenado (el software admite múltiples formatos de contextos) o componer un contexto desde un editor interactivo contenido en el propio programa.

[1]Página oficial del software Galicia: http://www.iro.umontreal.ca/~galicia/

Desde el editor se pueden realizar diversas operaciones de los contextos como, por ejemplo, variar el número de objetos y características y la relación entre los mismos, admitiendo así, cambios a nivel de concepto formal. En la siguiente ilustración se muestra una captura del editor de conceptos formales, en el que se ha creado el ejemplo mostrado con anterioridad.

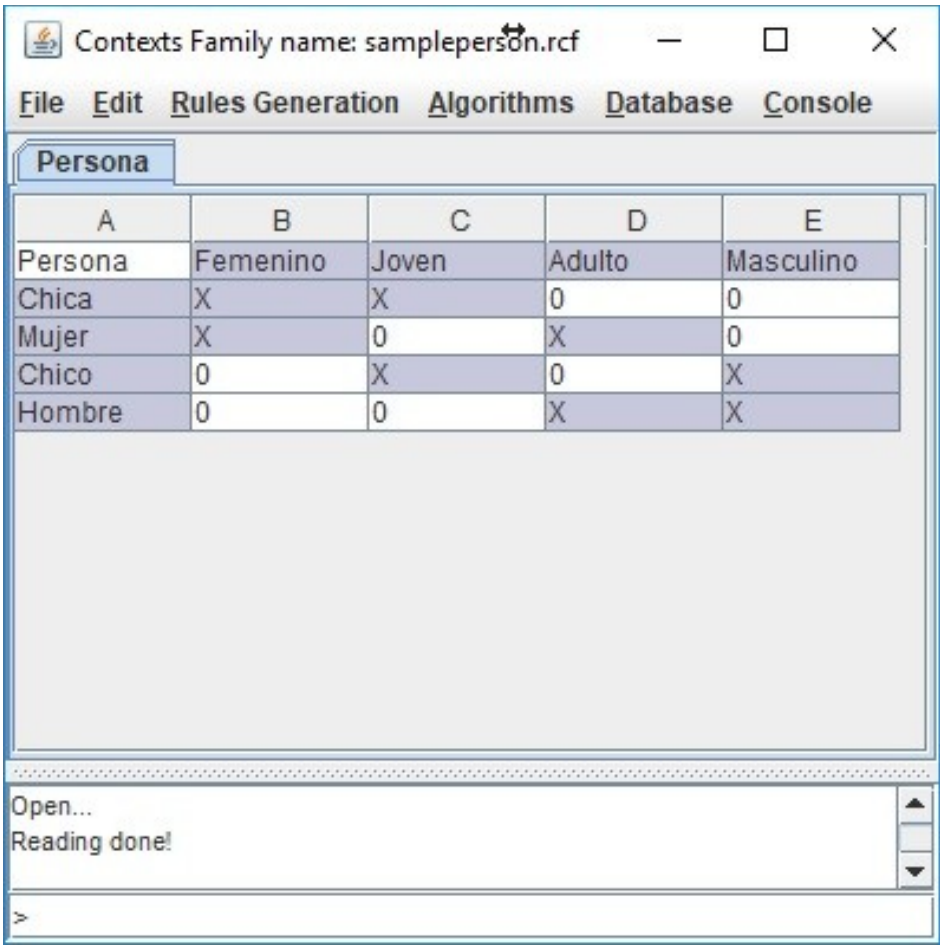

Ilustración 2: Editor de Contextos Formales de Galicia

En cuanto a la interfaz, resulta bastante sencilla e intuitiva, lo que facilita su uso e interpretación de los contextos formales. A continuación se puede observar el ejemplo dentro de su interfaz.

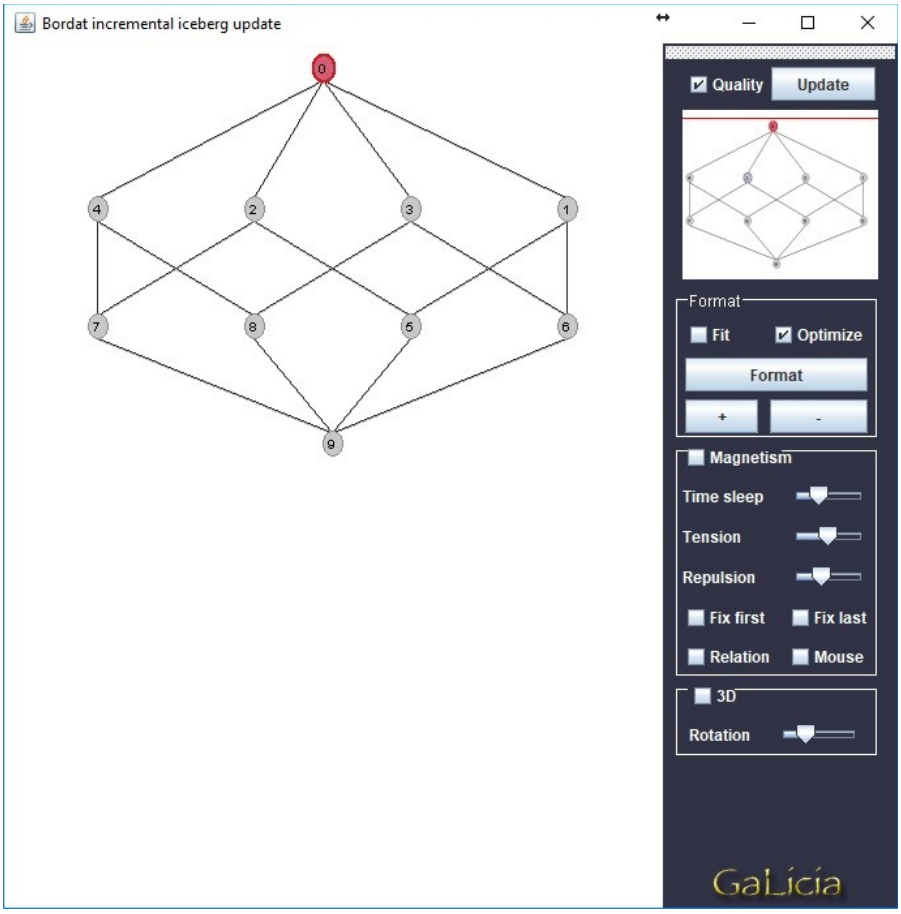

Ilustración 3: Contexto formal representado en Galicia

Además, Galicia también permite una representación en tres dimensiones del retículo, teniendo así, una de las características principales en las que se basa este proyecto.

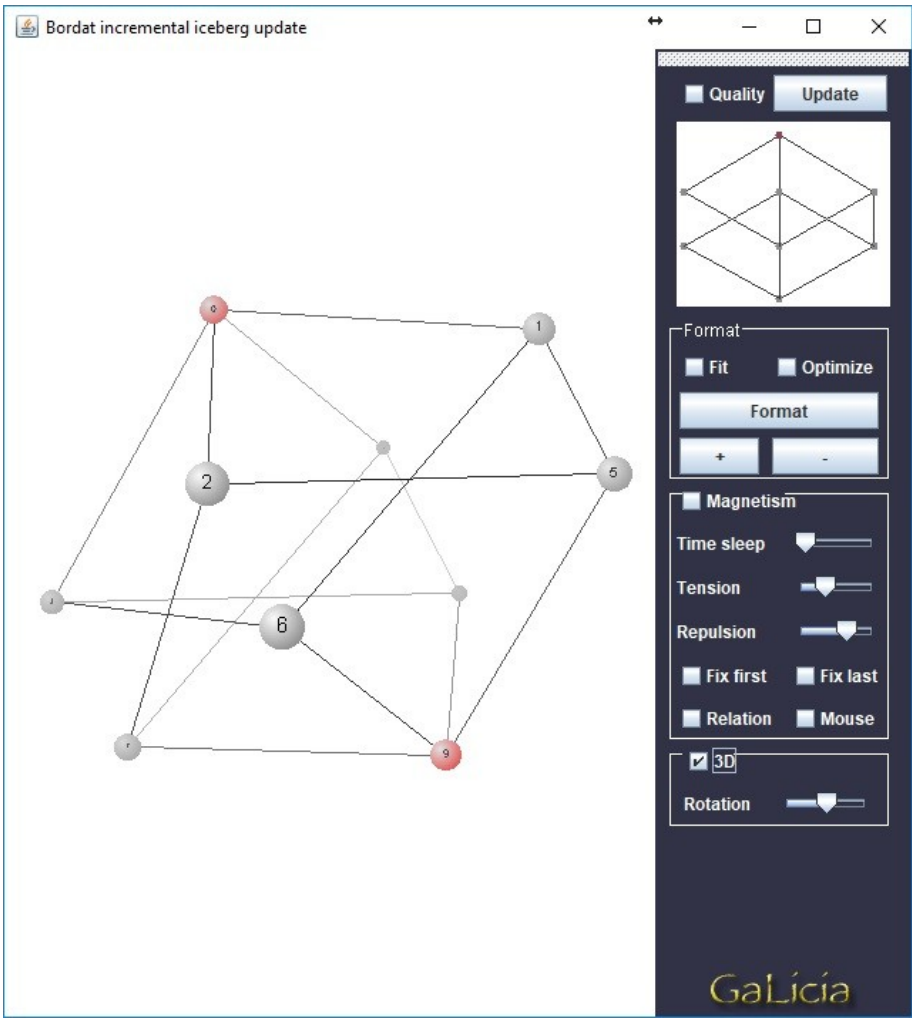

Ilustración 4: Representación en 3D del Contexto Formal en Galicia

Por contra, no se ha conseguido el acceso al código fuente del programa para poder utilizar la plataforma como ayuda para este proyecto.

- **FcaStone[2]:**

FcaStone es un software conversor de formatos de archivos para herramientas de FCA, desarrollado por Uta Priss. Su objetivo es mejorar la interacción entre el Análisis Formal de Conceptos, la edición de gráficos y el software de gráficos vectoriales.

Junto con la herramienta de visualización de gráficos Graphviz[3], FcaStone es capaz de convertir contextos formales en retículos para ser representados.

[2]Enlace al Sourceforge de FcaStone: http://fcastone.sourceforge.net/
[3]Página oficial del software Graphviz: http://www.graphviz.org/

A continuación se muestra un ejemplo de su representación gráfica:

	Boolean lattice	CDS lattice	geometric lattice	metric lattice	atomistic	Brouwerian	complemented	distributive	dually semimodular	graded	modular	relatively complemented	sectionally complemented	semimodular	Stonean	uniquely complemented
I	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
II		×			×		×		×	×			×			
III			×		×		×			×		×	×	×		
IV				×					×	×	×			×		
1				×		×		×	×	×	×			×	×	
2		×	×	×	×		×		×	×	×	×	×	×		
3					×		×					×	×			
4			×		×		×			×		×	×	×		
5					×				×	×						
6				×		×		×	×	×	×			×		
7					×		×			×						
8							×			×				×		
9		×			×		×		×	×			×			
10		×			×		×		×	×		×	×			

Ilustración 5: Ejemplo de contexto formal.

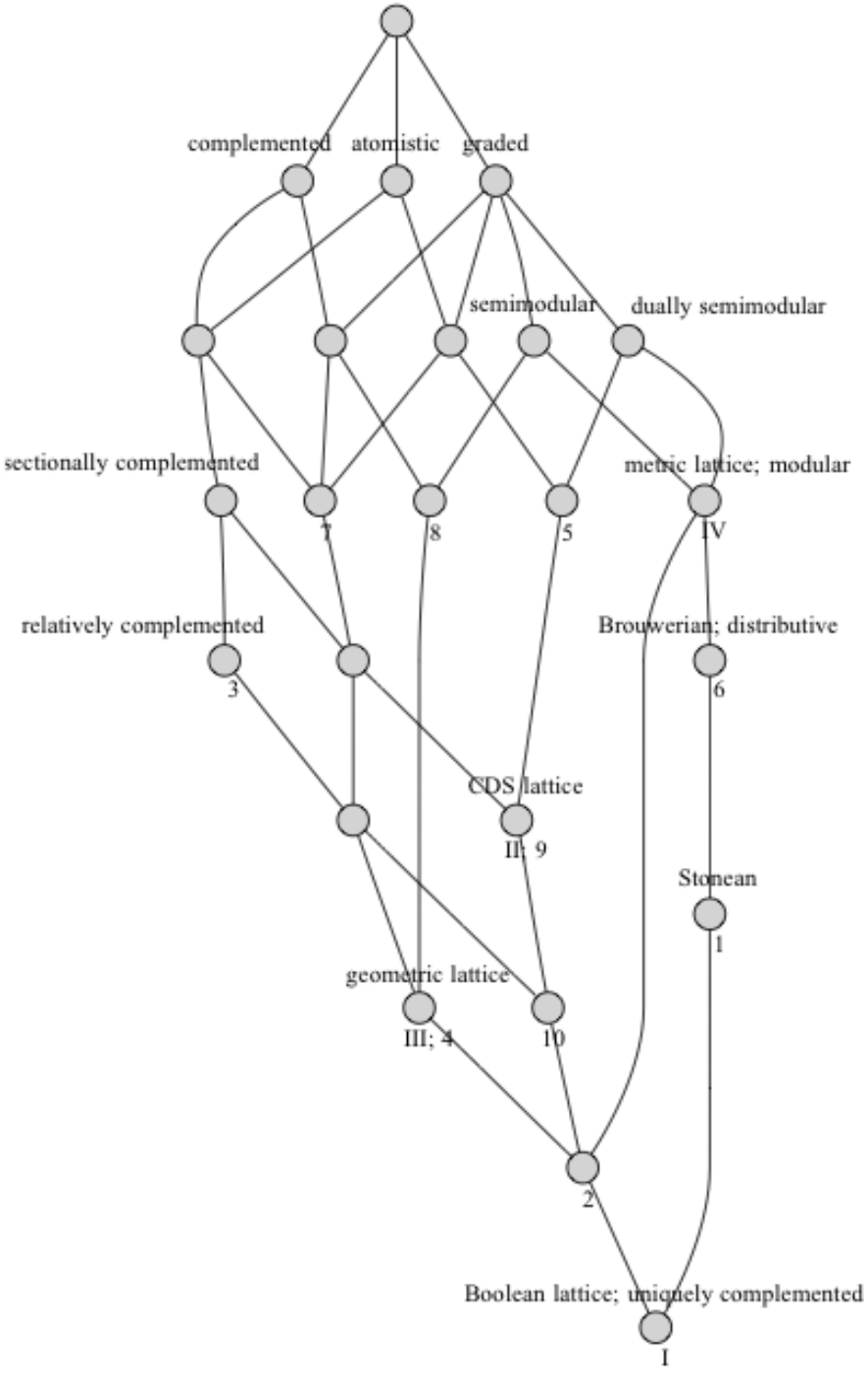

Ilustración 6: Representación gráfica del retículo del contexto formal anterior

- **ConExp[4]:**

ConExp se trata de una herramienta básica para el estudio e investigación del FCA. Su desarrollo data del año 2000 y fue implementado por la Universidad Técnica Nacional de Ucrania

[4]Enlace al Sourceforge de Conexp: https://sourceforge.net/projects/conexp/?source=typ_redirect

con la supervisión. Actualmente es un proyecto de código abierto y se encuentra alojado en la plataforma Sourceforge.

La funcionalidad es similar al resto de programas de Análisis Formal de Conceptos. Según el manual incluido en el propio programa, las funcionalidades son las siguientes:

- Edición de contexto.
- Construcción de retículos desde el contexto formal.
- Encontrar bases de implicaciones que sean fieles al contexto.
- Encontrar bases de reglas de asociación que sean fieles al contexto.
- Realizar exploración de atributos.

Como en el caso anterior, se mostrarán ilustraciones del editor de contextos formales como de la representación de retículos.

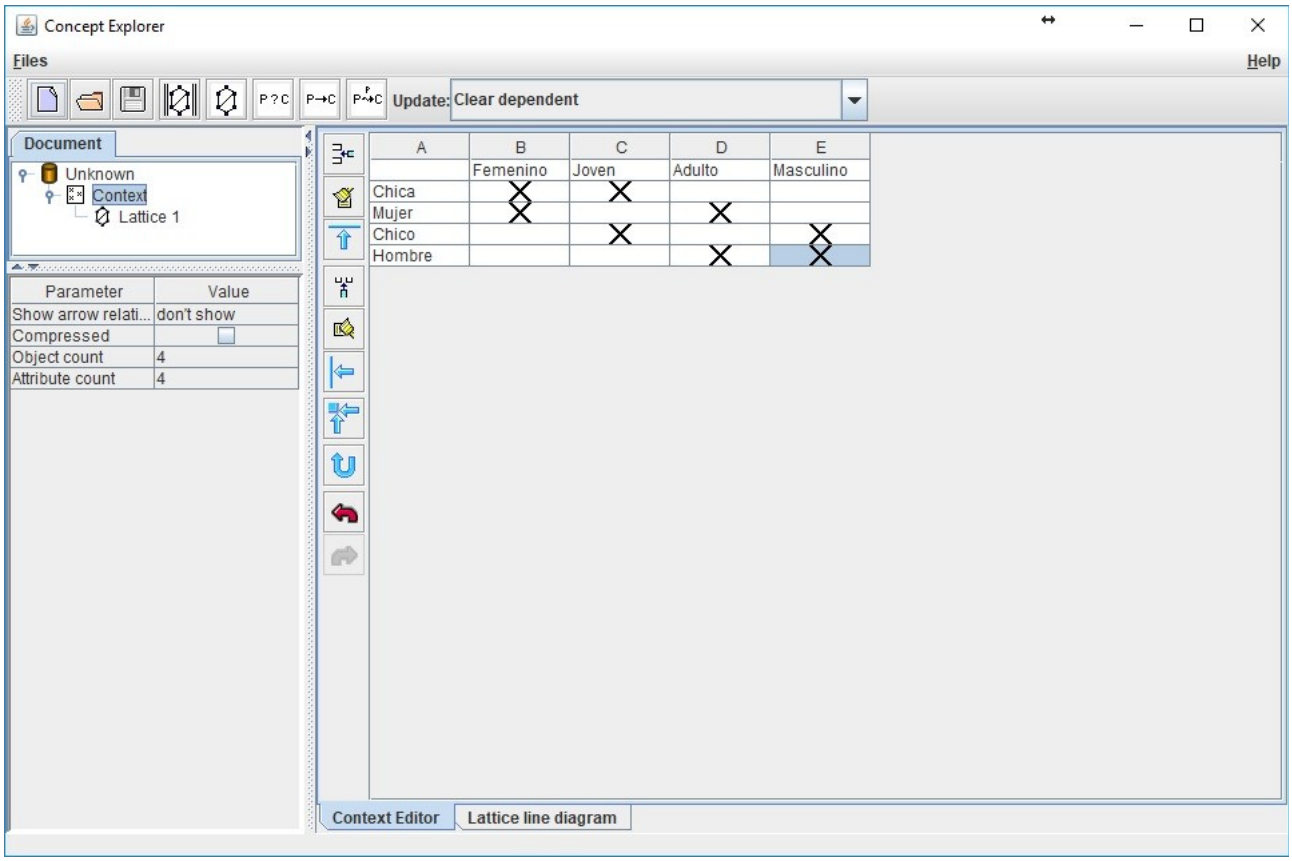

Ilustración 7: Editor de contextos formales de ConExp.

- **ConExp-FX[5]:**

El software ConExp-FX es una reimplementación de la plataforma ConExp mencionada anteriormente desarrollada en lenguaje Java. Esta herramienta, aporta a mayores una mejora en la interacción del usuario y la representación gráfica del retículo.

El proceso de funcionamiento del programa ConExp-FX es el siguiente:

1. Creación o importación desde el sistema de archivos locales de la información del contexto formal.
2. Generación de una representación bidimensional del retículo asociado al contexto formal.
3. Posibilidad de modificar la representación del retículo (tres dimensiones, coordenadas polares, etc) y visualización del mismo.

[5]Página de Conexp-FX: https://lat.inf.tu-dresden.de/~francesco/conexp-fx/conexp-fx.html

El editor de contextos formales de ConExp-FX tiene el siguiente aspecto.

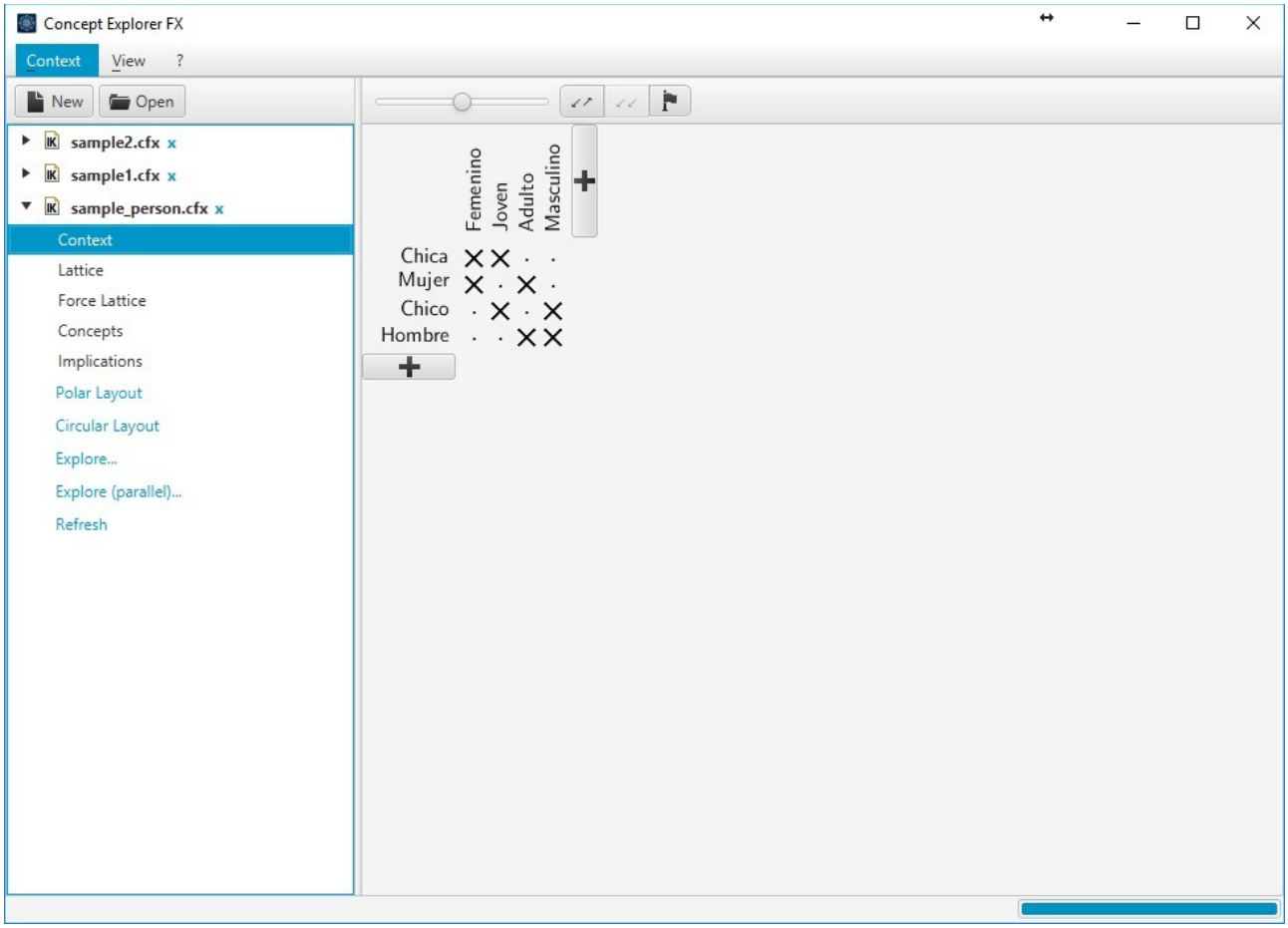

Ilustración 9: Editor de contextos formales de ConExp-FX

Con el editor se ha generado el ejemplo utilizado durante el documento para la explicación del Análisis de Conceptos Formales. En la siguiente ilustración se muestra el ejemplo utilizado de contexto formal visualizado mediante el programa ConExp-FX

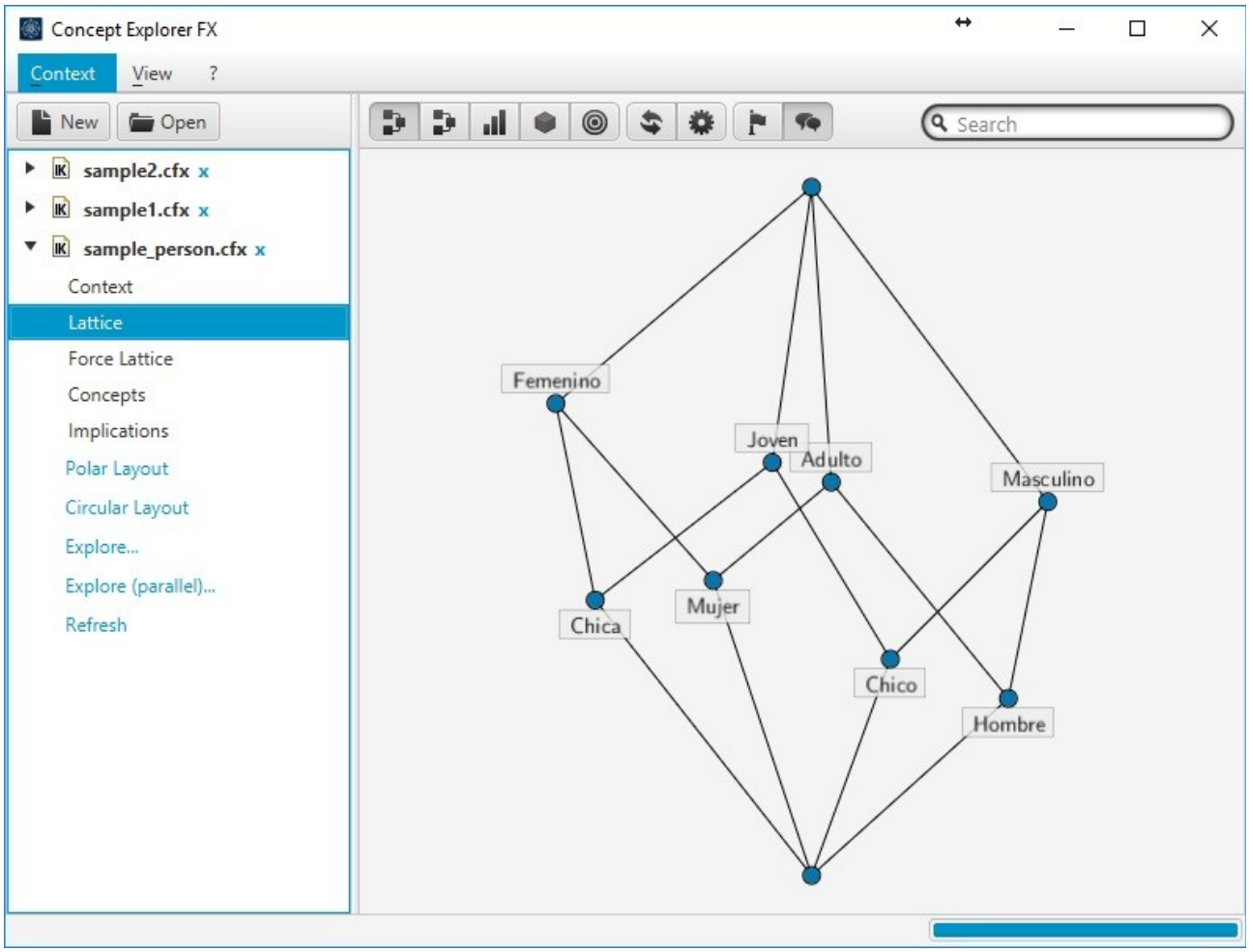

Ilustración 10: Interfaz de ConExp-FX,

Además, ConExp-FX también tiene la funcionalidad de representar los retículos en tres dimensiones, como se muestra a continuación.

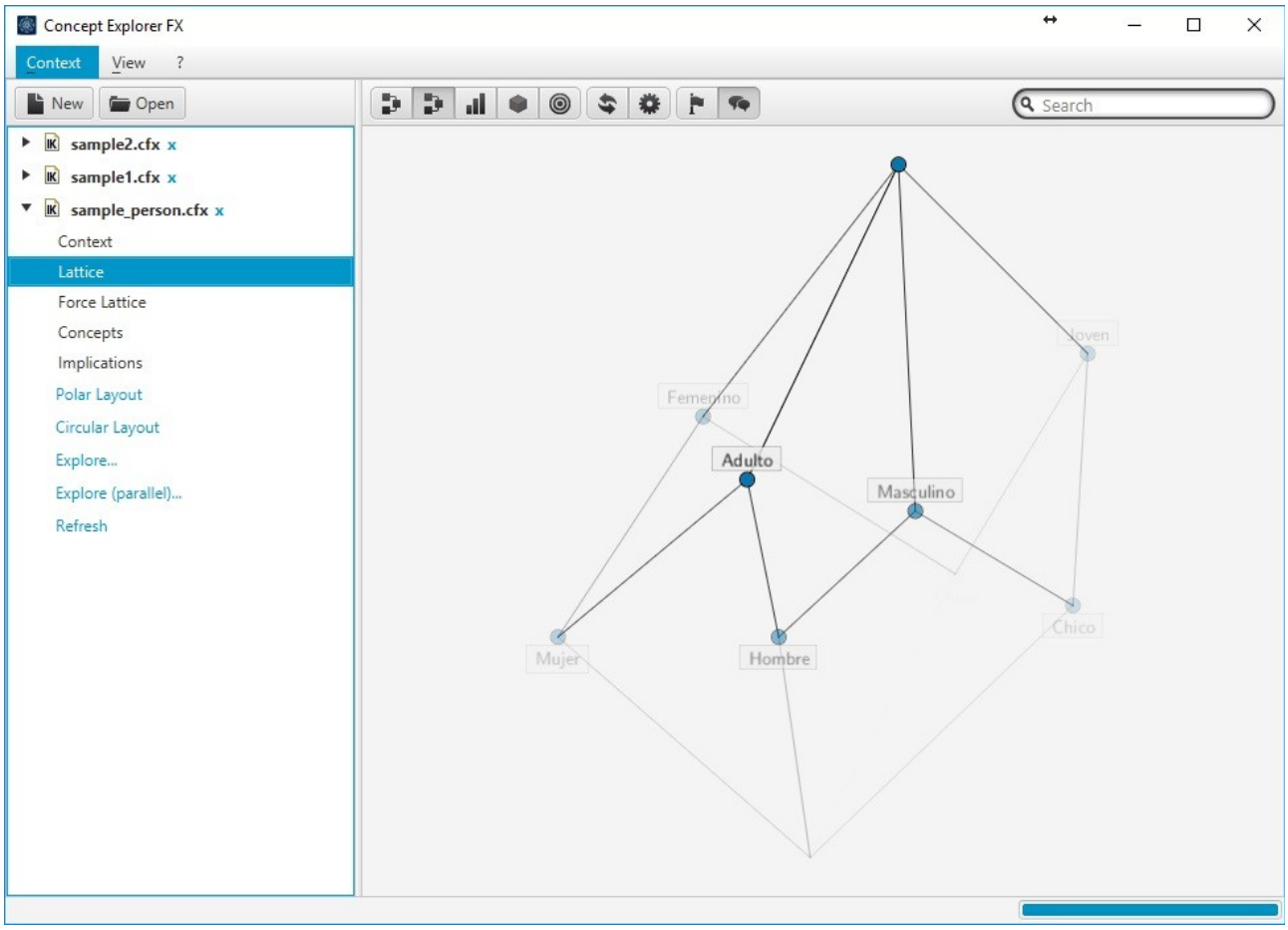

Ilustración 11: Representación en 3D del contexto formal con ConExp-FX

Otra de las ventajas de ConExp-FX es que su código está alojado en la plataforma GitHub, por lo que es accesible para su desarrollo.

Por todos estos motivos, es por lo que la plataforma ConExp-FX ha sido seleccionada como soporte para este proyecto.

2.2: Minecraft.

Minecraft[6] es un videojuego de construcción de un género que se denomina “sandbox”. Este término hace referencia a los juegos de mundo abierto y no lineales, es decir, no es el tipo de videojuego que sigue una historia en la que perseguir un objetivo concreto, sino que el objetivo lo pone el propio jugador. El videojuego fue creado por el sueco Markus Persson y luego desarrollado en profundidad por la empresa, fundada por el propio Persson, llamada Mojang AB[7] en 2009. La

[6]Página oficial de Miecraft: https://minecraft.net/
[7]Página oficial de Mojang AB: https://mojang.com/

primera versión pública de Minecraft fue lanzada el 17 de mayo de 2009 y desde entonces, ha recibido múltiples actualizaciones que han ampliado características del videojuego.

Esta plataforma ha sido la seleccionada para el desarrollo del proyecto, la representación de contextos formales en tres dimensiones. A continuación se explicarán algunas de características propias del videojuego y su relación al desarrollo del proyecto.

Ilustración 12: Mundo de Minecraft.

2.2.1: Estilo del juego.

Como ya se ha mencionado anteriormente, Minecraft es un videojuego de construcción en un mundo virtual de tres dimensiones, donde la unidad básica de la que se componen todos los elementos del mundo de Minecraft se denomina bloque. El usuario puede interaccionar con los bloques, ya sea colocándolos, recolectándolos o destruyéndolos, lo que otorga la experiencia de libre construcción base del videojuego. Además el videojuego también permite la exploración libre en las tres dimensiones del mundo de Minecraft.

Esta funcionalidad resultó ser uno de los principales atractivos para el desarrollo del proyecto, ya que permite la creación de estructuras en cualquier parte del mundo de Minecraft (esas estructuras serán los conceptos formales en este proyecto).

Ilustración 13: Mundo de Minecraft desde el punto de vista del usuario.

2.2.4: Elementos de Minecraft.

Dentro del mundo de Minecraft tenemos una seria de conceptos diferentes con los que podemos actuar. A continuación se listarán esos conceptos con una descripción de cada uno de ellos:

- **Jugador:** Es la interfaz entre el usuario y el mundo de Minecraft. Las acciones que realicemos mediante los controles de ratón y teclado se verán reflejados en el jugador y será siempre la referencia central en el mundo. El jugador viene representado en el juego como una figura antropomórfica y con ella se podrán realizar las acciones de desplazamiento por el mapa y la interacción con el resto de elementos del mundo Minecraft. El jugador también dispone de un inventario en el que se pueden almacenar algunos de estos elementos.

Ilustración 14: Jugador de Minecraft.

- **Bloques:** Es la unidad básica del mundo Minecraft. Todas las estructuras que son representadas en el juego están formadas a partir de estos bloques. Se tratan de unos cubos de 16x16x16 pixeles de dimensión con multitud de metadatos en su interior. Estos metadatos son los que hacen que se comporten de una manera u otra cuando el jugador interactúa con ellos. Cada bloque viene asociado a una textura que le otorga el aspecto visual al bloque. La textura es un archivo .png de 16x16 pixeles que se mostrará en cada una de las caras del cubo que forma el bloque.

Ilustración 15: Ejemplo de bloque

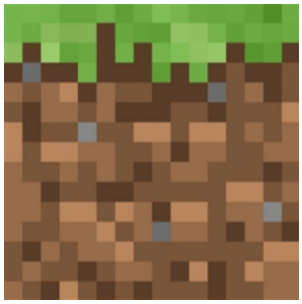

Ilustración 16:
Ejemplo de textura.

- **Items:** Son objetos que se crean en el propio videojuego pero que no ocupan un lugar en el mundo de Minecraft, es decir, no forman parte de las estructuras que forman el mundo. La funcionalidad de estos objetos es añadirle otro tipo de interacciones al jugador, como por ejemplo, mejorar su capacidad para destruir bloques.

Ilustración 17: Ejemplo de items en Minecraft.

2.2.3: Mundo de Minecraft.

El mundo en el que se distribuyen todos los elementos del mundo de Minecraft es un matriz tridimensional con coordenadas “x”, “y” y “z”, donde cada posición en la matriz representa una celda en el mundo. Las coordenadas “x” y “z” hacen referencia al plano horizontal, mientras que la coordenada “y” hace referencia a la vertical del mundo. El tamaño de la celda es equivalente al tamaño de un bloque.

El origen de coordenadas se encuentra en el centro del mundo de Minecraft y el límite se encuentra a 32000000 bloques en cada una de las direcciones. Para que nos hagamos una idea, haciendo una conversión en la arista de un bloque es de un metro de longitud, el mundo de Minecraft nos daría una superficie de aproximadamente $4{,}1x10^{9}$ km², unas 8 veces superior a los $5{,}1x10^{8}$ de la superficie terrestre, por ejemplo. En la siguiente ilustración podemos observar una comparación del mundo de Minecraft con ejemplos de otros cuerpos celestes del universo.

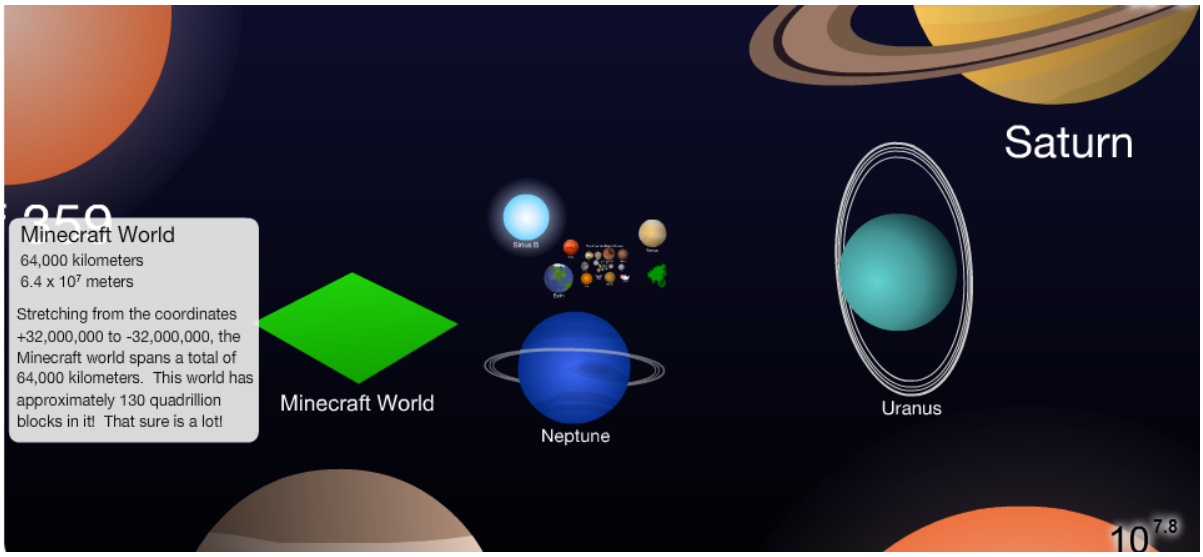

Ilustración 18: Comparación de tamaños del mundo Minecraft

Por lo que se puede asegurar que el mundo de Minecraft no está limitado en cuestión de tamaño. Sin embargo si está limitado por la capacidad computacional de la máquina en la que se está ejecutando Minecraft. Para la generar la visualización, se divide el mapa en secciones de 16 bloques de distancia llamados "*chunks*" (cuya traducción literal sería "trozo" o "pedazo") que se almacenan en la memoria de la máquina. En la configuración de Minecraft se puede ajustar el rango de mapa visible por el usuario, que se traduce en el número de *chunks* almacenados en la memoria y mostrados simultáneamente en pantalla, para que se ajuste así a la capacidad computacional de la máquina.

2.2.4: Modos de juego.

Al iniciar el juego, se presenta la posibilidad de crear un nuevo mundo o cargar uno ya creado anteriormente y que hayamos almacenado en la máquina. Al crear un mundo nuevo, se muestran una serie de opciones por las que se regirá el mundo creado. Por defecto, se genera un

nuevo mundo mediante un algoritmo pseudo-aleatorio, por lo que la posibilidad de generar dos mundos iguales es casi nula, sin embargo, se puede modificar el algoritmo de creación por el usuario. También existe la posibilidad de cargar un mapa albergado en un servidor externo al que podemos acceder por red.

Para el caso de este proyecto se ha de crear un mundo con el algoritmo llamado *"Superflat"* (extraplano en castellano) que generará un mundo que sólo contendrá un plano de bloques en el plano y = 0. Visualmente se traduce en una superficie plana hasta los límites del mapa.

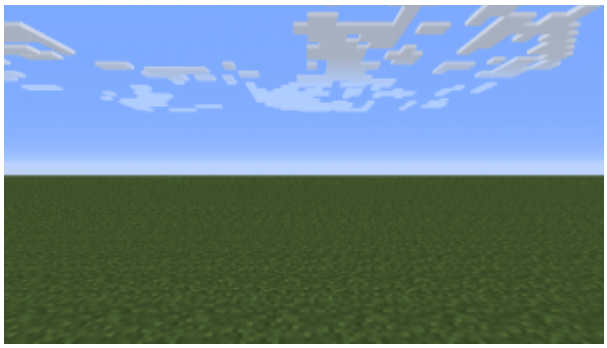

Ilustración 19: Mundo Supereflat

Una vez creado el mundo, los cambios que realicemos en el mismo podrán ser almacenados para poder modificar el mundo en diferentes sesiones.

En las opciones de creación del mundo se ha de especificar el modo de juego en el que se basará nuestro mundo. Para ello se tienen las siguientes opciones:

- **Survival:** Modo de supervivencia en la que el jugador tendrá que explorar el mundo generado para encontrar recursos que le ayuden a sobrevivir. El jugador tiene asociados unos puntos de salud que tendré que mantener por encima de 0 para continuar con la aventura. En el mapa se generarán los recursos necesarios para sobrevivir así como ciertos personajes no controlados por el usuario con el objetivo de los puntos de salud del jugador. Existen situaciones en el mundo en el que se puede generar un punto de control, que es el momento en el que se almacenan los cambios en el mundo. En el instante en el que los puntos de salud del jugador lleguen a 0, todos los avances realizados después de haber alcanzado el punto de control se perderán.

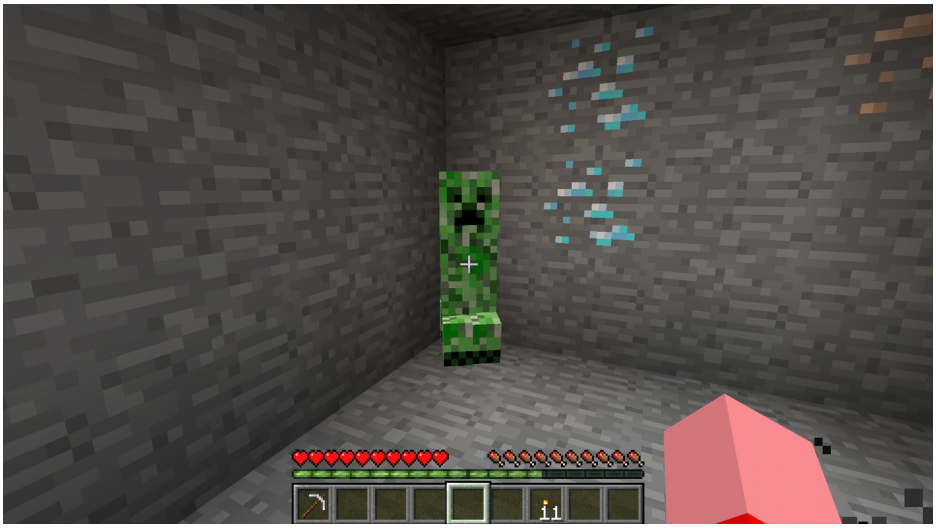

Ilustración 20: Modo survival

- **Hardcore:** En castellano, modo extremo, se trata de un modo similar al anterior con la diferencia que en este caso existen los puntos de control. Por lo que en el momento en el que los puntos de salud lleguen a 0, se perderá también el mundo creado.

- **Creative:** Es el modo de juego creativo. En este caso el juego se basa únicamente en la construcción libre (sin puntos de salud ni personajes ajenos al jugador) y se tendrá acceso a la totalidad de los bloques de Minecraft para poder ser colocados en cualquier punto. Otra de las diferencias con el resto de modos es que en este modo se tiene la capacidad de vuelo, que nos permitirá desplazarnos con libertad a cualquier punto del mundo de Minecraft. Estas características propias de este modo de juego hacen que sea propicia para la naturaleza de este proyecto. En la ilustración se puede observar como en este modo desaparecen los puntos de salud.

Ilustración 21: Modo creative

2.2.5: Mods.

Un mod (proveniente de la palabra en inglés modification) es una extensión para un software que permite alterar la funcionalidad del programa para añadir nuevas opciones o variar el comportamiento del mismo.

Para el caso de este proyecto, ha sido necesario desarrollar un mod de Minecraft para que sea el programa capaz de interactuar con el Análisis Formal de Conceptos y a visualización de retículos, modificando las características de Minecraft a la hora de generar un nuevo mundo.

En el siguiente apartado se procederá al análisis de la herramienta Minecraft Forge, que permite la creación y ejecución de mods en Minecraft.

2.3: Minecraft Forge.

Minecraft Forge[8] es una herramienta que permite la instalación de mods en Minecraft, y proporciona las librerías necesarias para el desarrollo de los mods.

Forge permite la modificación de prácticamente todas las interacciones del jugador con el mundo de Minecraft, así como una completa edición del mundo creado.

2.3.1: Instalación de Forge.

Para poder ejecutar y desarrollar mods para Minecraft usando la herramienta Forge, es necesario primero instalar la versión correcta de Forge, ya que así se garantizará la correcta ejecución del mod.

Las versiones de Forge tienen compatibilidad regresiva, es decir, una versión de Forge instalada en una máquina podrá ejecutar mods de esa versión o inferiores, pero no se asegura la ejecución de mods desarrollados en una versión superior. Pese a esto, se recomienda instalar la misma versión de Forge en la que se ha desarrollado el mod para así asegurar su correcto funcionamiento.

Se pueden descargar todas las versiones de Forge desde su propia plataforma online, donde se albergan tanto las versiones de ejecución, como las librerías para desarrollo. La versión de ejecución presenta la siguiente interfaz.

[8]Página oicial de Minecraft Forge: https://files.minecraftforge.net/

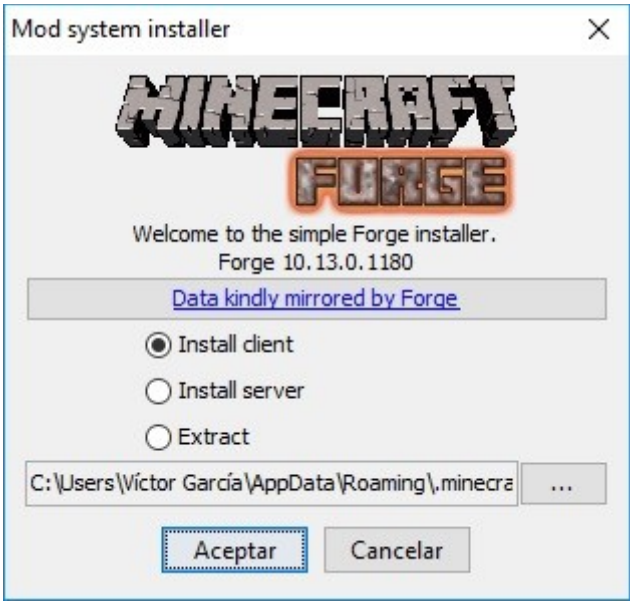

Ilustración 22: Instalador de Minecraft Forge

Como se puede observar, existen 3 opciones que son *Install client* para ejecutar los mods en modo cliente, *Install server* para el modo servidor, u *extract* que realiza una copia del .jar de Minecraft Forge en el directorio seleccionado.

Una vez instalado, se ha de seleccionar la versión instalada de Forge en la interfaz de lanzamiento de Minecraft.

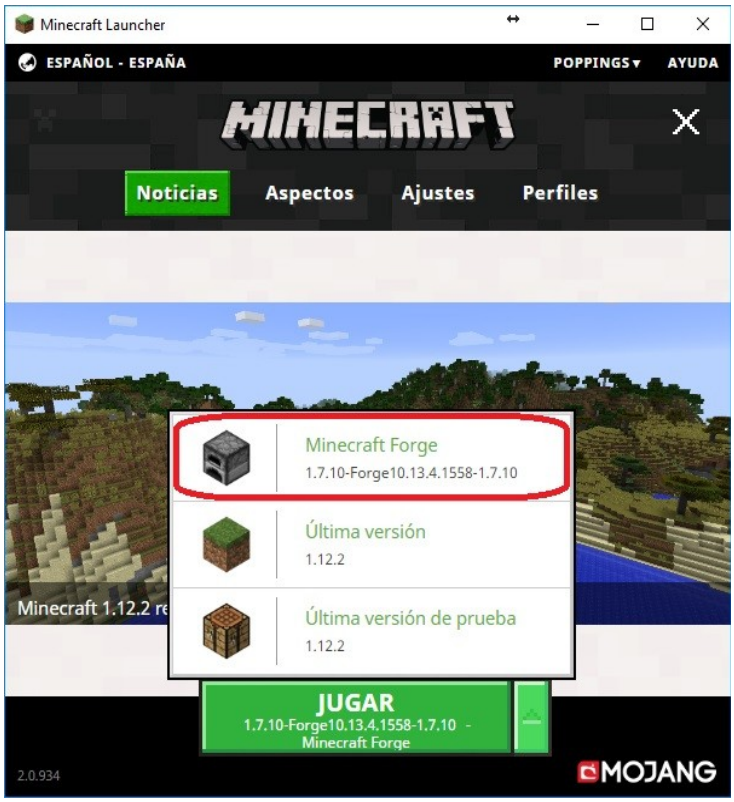

Ilustración 23: Interfaz de lanzamiento de Minecraft.

Si todo el proceso se ha realizado correctamente, se ejecutará Minecraft con la versión de Forge que se haya instalado.

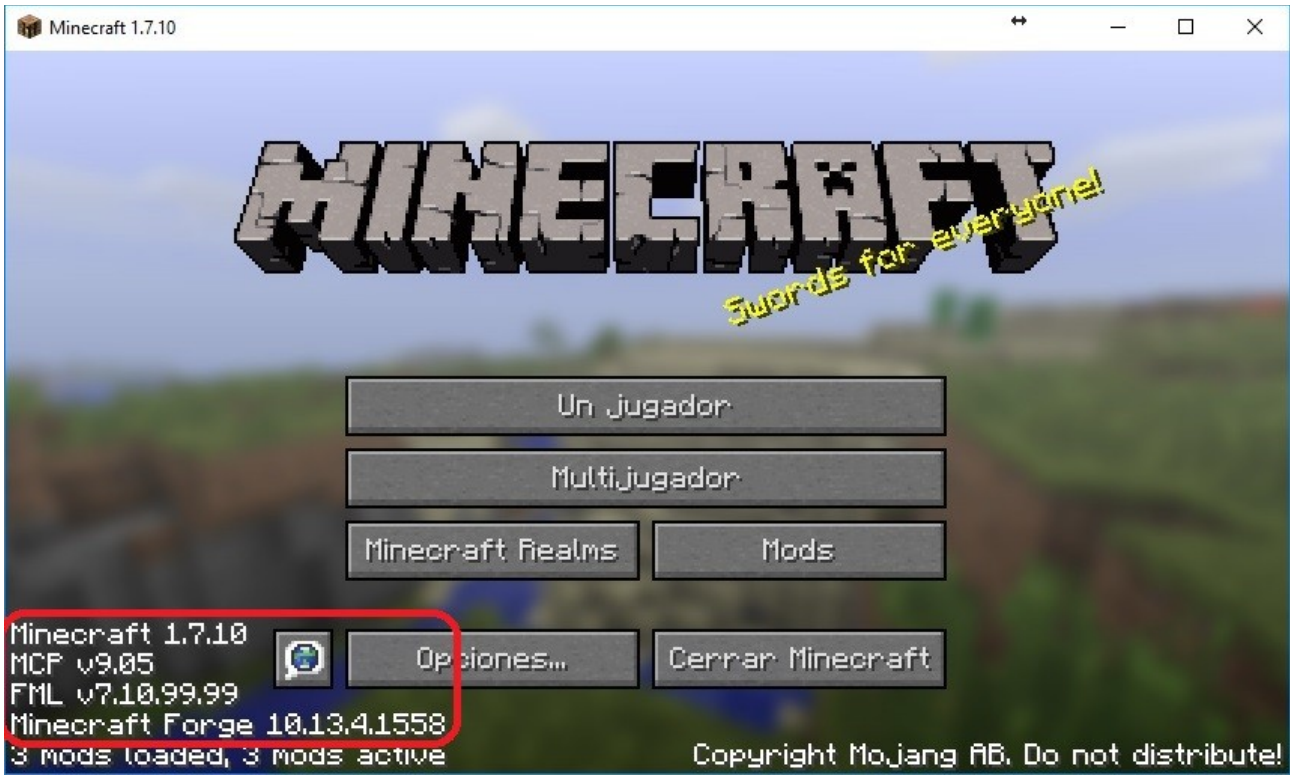

Ilustración 24: Interfaz de inicio de Minecraft con Forge instalado.

2.3.2: Desarrollo.

El lenguaje de programación en el que se desarrollan los mods de Minecraft Forge es Java, por lo que, a mayores de todas las funciones que otorga las librerías de Forge, también podremos incluir cualquiera de las librerías creadas para Java y sus correspondientes funciones. Esto hace que las posibilidades a la hora de desarrollar un mod para Minecraft sean muy amplias.

Para que el mod funcione, es de carácter obligado crear una serie de clases en Java específicas para que el mod se cargue de forma correcta en el videojuego. Cada una de estas clases hace referencia a una de las fases de ejecución del juego y en ellas se tienen que especificar ciertos parámetros que caracterizarán el mod.

- **Main.java.**

Esta será la clase principal del mod. En ella se especifican los parámetros que definen al mod (*Id., Name y Version*) y se declaran las funciones de las fases de carga del juego que son *preInit, init* y *postInit*, en relación a las funciones que serán llamadas antes de la carga del mundo (*preInit)*, en el instante de la carga del mundo (*init)* y en el momento de cierre del mundo (*postInit*). Las funciones dentro de éstas, están declaradas en otras tres clases, denominadas *proxy*, que se explican a continuación.

- **ClientProxy.java, ServerProxy.java y CommonProxy.java.**

Estas tres clases contienen las funciones que serán llamadas en la clase principal del mod. Esta diferenciación de clases es debida a que en algunas ocasiones sólo es necesaria que determinada función sea ejecutada en la máquina del cliente (*ClientProxy*), en la máquina del servidor (*ServerProxy*) o en ambos (*CommonProxy*).

Como el mod desarrollado para este proyecto sólo se va a ejecutar en modo local de la máquina, se utilizará únicamente la clase CommonProxy. Así, si en algún momento se quiere utilizar este mod como servidor, no será necesario realizar cambios en el código.

Existen otras clases que no son obligatorias para el funcionamiento del mod pero que son necesarias para el desarrollo de nuestro proyecto. Estas clases se encargarán de la creación de nuevos bloques (*BasicBlock.java, ModBlock,java, etc.*) y de la generación de nuevos mundos (*ModWorlrdGen.java*).

Otra clases muy útiles son las de tipo *EventHandler,* que también estarán según se quieran ejecutar en el lado del cliente, servidor o ambos. Estas clases contienen las funciones que serán llamadas en el momento en el que se produzcan determinados eventos en la ejecución del mod.

Finalizada la parte de programación, el mod se podrá compilar utilizando una herramienta de compilación Gradle. Una vez realizada la compilación se generará un archivo .jar (*Java Archive*). Con toda la información del mod.

2.3.3: Instalación de mods con Minecraft Forge.

Una vez se tiene el archivo .jar con todos los datos del mod que se quiera ejecutar en Minecraft se puede proceder a su instalación. Para ello, se tiene que tener con anterioridad instalado en la máquina la plataforma Minecraft y la versión de Forge con la que ha sido desarrollado el mod.

Para que el mod se ejecute, es necesario almacenar el archivo .jar del mod en la carpeta "Mods" situada en el directorio donde se encuentre instalado Minecraft. Hecho ésto, se podrá ejecutar el mod desde Minecraft.

Capítulo 3: Contribuciones personales.

El desarrollo de este proyecto se divide en dos partes diferenciadas. Por un lado están los cambios en el código de ConExp-FX para poder extraer la información de los conceptos formales y que puedan ser trasladados al mod de Minecraft, y por otro lado se encuentra el desarrollo del mod de Minecraft que interpretará esos datos y mostrará la representación tridimensional inmersiva e interactiva por parte del usuario. A continuación se procederá a una explicación del modo en el que se ha realizado el desarrollo de la totalidad del proyecto, incluyendo la parte de ConExp-FX y la parte del mod de Minecraft.

3.1: Cambios en ConExp-FX.

Como se ha explicado con anterioridad en el apartado 2.1.3 sobre el software de FCA, ConExp-FX es una plataforma que permite la representación tridimensional de contextos formales. El código fuente de este programa es de libre acceso y está alojado en la plataforma GitHub y a partir de este código se han realizado los cambios necesarios para el funcionamiento del proyecto.

3.1.1: Resumen de cambios.

Para poder añadir la funcionalidad de visualización de conceptos formales en Minecraft utilizando ConExp-FX como software que genera la información del contexto formales se ha realizado un análisis de cómo realiza la visualización tridimensional en el propio programa y de cómo se puede extraer la información el contexto desde el mismo código para poder ser transferida a Minecraft. El funcionamiento del programa una vez realizados los cambios será de la siguiente forma:

1. Creación o importación desde el sistema de archivos locales de la información del contexto formal.
2. Generación de una representación bidimensional del retículo asociado al contexto formal.
3. Activar la opción “Visualización Tridimensional”.
4. Modificar las posiciones de los conceptos formales si fuera necesario.

5. Activar la opción “Transferir a Minecraft”.

La interfaz de usuario se verá también alterada debido a los cambios. Ha sido añadido un nuevo botón a la interfaz con la funcionalidad de transferir la información del contexto formal a Minecraft. En el momento en el que el usuario accione el botón, se transferirá a Minecraft tanto la información de cada uno de los conceptos con su respectiva intensión y extensión, como la su posición en la representación tridimensional de ConExp-FX para poder hacer la exploración inmersiva en Minecraft.

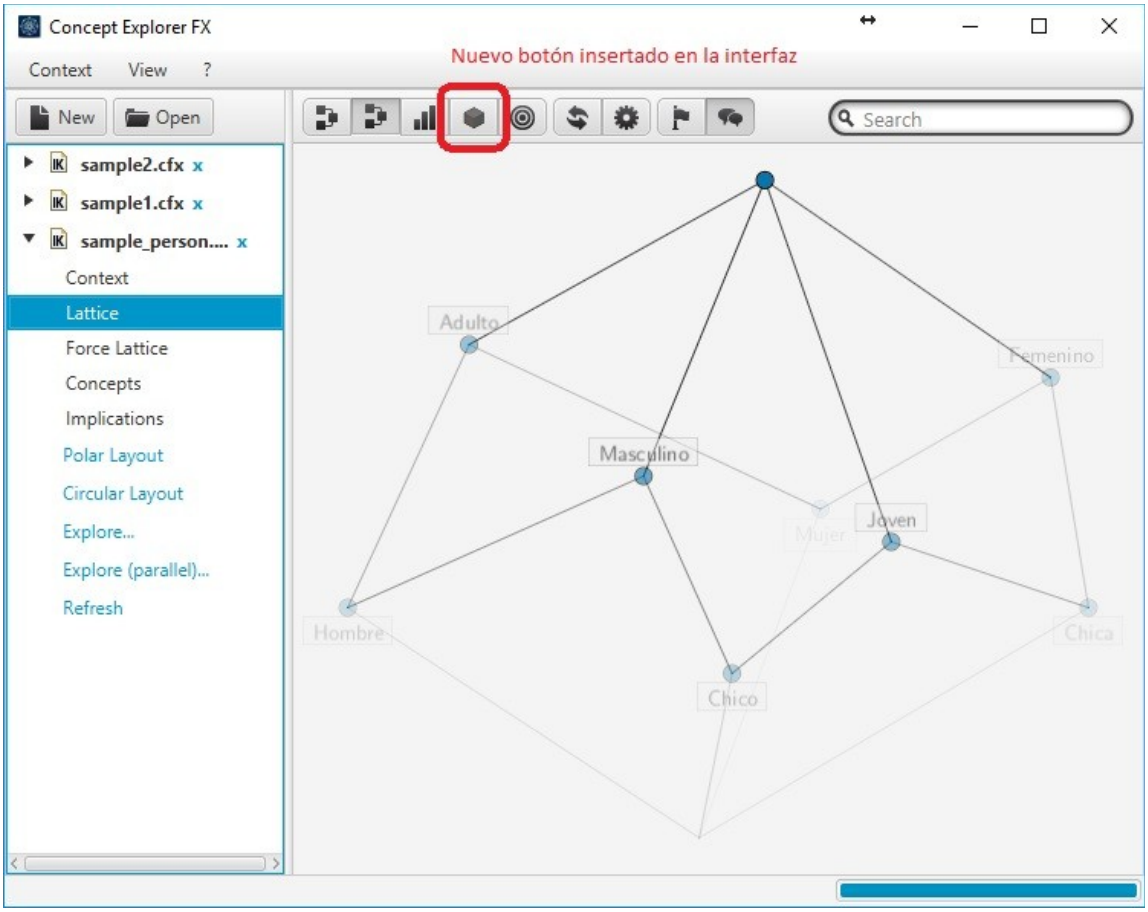

Ilustración 25: Botón con la opción Transferir a Minecraft en la interfaz de ConExp-FX.

A continuación se hará una explicación de cuáles han sido los cambios a nivel de código fuente del programa para la realización de estas modificaciones.

3.1.2: Especificación del diseño las clases de ConExp-FX.

La implementación de los cambios en ConExp-FX ha sido realizada sobre el propio código fuente del programa, modificando o agregando sus clases de Java. Los cambios realizados, organizados por los paquetes en los que se encuentran las clases, han sido los siguientes:

conexp.fx.gui.graph.option

GraphTransformation.java:

Se ha añadido la declaración para a opción "Minecraft"

conexp.fx.gui.graph

ConceptGraph.java:

En esta clase se ha implementado la funcionalidad del botón para transferir a Minecraft, modificando la función *createTransformationBox,* que se encarga de la creación de los botones de la interfaz del programa para poder añadir el botón que dará comienzo a la nueva rutina de ejecución para el proyecto.

Para el diseño del botón se ha utilizado el mismo tipo de interfaz que el resto de los botones, además se ha tenido que incluir una nueva imagen para el botón en las recursos del programa. Se trata de una imagen de 16x16 píxeles de tamaño con la representación de un cubo (cube.png). Esta imagen ha sido creada con Paint3D[9], un programa de edición de imágenes en tres dimensiones de Microsoft.

Ilustración 26: cube.png

Se ha modificado la función *computeValue* que se encarga de ejecutar las funciones correspondientes a cada uno de los botones de la interfaz de ConExp-FX. Para el caso de la función del botón Minecraft, se ha implementado la función *minecraftOption,* la cuál se encarga de lo siguiente:

- Obtiene la información del todos los conceptos formales del contexto que se encuentran en la lista *fca.concepts*.

[9]Enlace a la página de Paint 3D: https://www.microsoft.com/es-es/store/p/paint-3d/9nblggh5fv99

- Recorre la lista extrayendo la información de cada uno de los conceptos formales. Esta información incluye la extensión y la intensión del concepto formal en forma de vector, y la posición del contexto tridimensional en coordenadas cartesianas x-y-z.
- Toda esta información se introduce en un objeto del tipo JSON*.
- El objeto JSON es enviado a Minecraft mediante la función *sendMsg,* función que será explicada más adelante.

*Un objeto JSON es un objeto propio de Java (contenido en la librería *org.json*) que contiene las funciones necesarias para generar un texto JSON. Por su parte, JSON (acónimo de las siglas en inglés de *JavaScript Object Notation*) es un formato de texto estándar muy utilizado para el intercambio de datos debido a su sencillez y ligereza. A continuación se muestra el texto en formato JSON, para el contexto de ejemplo del proyecto.


```
{
  "context":[
    {
      "extent":"[Chico, Chica, Hombre, Mujer]",
      "x":0,
      "index":0,
      "y":20,
      "z":0,
      "intent":"[]"
    },
    {
      "extent":"[Hombre, Mujer]",
      "x":-18,
      "index":1,
      "y":30,
      "z":14,
      "intent":"[Adulto]"
    },
    {
      "extent":"[Chica, Mujer]",
      "x":16,
```

```
  "index":2,
  "y":32,
  "z":20,
  "intent":"[Femenino]"
},
{
  "extent":"[Chico, Hombre]",
  "x":-6,
  "index":3,
  "y":38,
  "z":8,
  "intent":"[Masculino]"
},
{
  "extent":"[Chico, Chica]",
  "x":6,
  "index":4,
  "y":42,
  "z":12,
  "intent":"[Joven]"
},
{
  "extent":"[Chico]",
  "x":-2,
  "index":5,
  "y":50,
  "z":18,
  "intent":"[Masculino, Joven]"
},
{
  "extent":"[Mujer]",
  "x":2,
  "index":6,
  "y":40,
```

```
      "z":30,
      "intent":"[Adulto, Femenino]"
    },
    {
      "extent":"[Hombre]",
      "x":-24,
      "index":7,
      "y":46,
      "z":22,
      "intent":"[Adulto, Masculino]"
    },
    {
      "extent":"[Chica]",
      "x":18,
      "index":8,
      "y":46,
      "z":24,
      "intent":"[Femenino, Joven]"
    },
    {
      "extent":"[]",
      "x":-4,
      "index":9,
      "y":60,
      "z":40,
      "intent":"[Adulto, Masculino, Femenino, Joven]"
    }
  ]
}
```

SocketClient.java:

Esta clase ha sido añadida para establecer la conexión con el mod de Minecraft. La comunicación entre las dos plataformas se realizará mediante *sockets.*

Un *socket* es un mecanismo de conexión entre dos programas mediante el protocolo TCP/IP. Que la conexión se realice de esta forma permite que no sea necesario que los programas se encuentren en la misma máquina, aunque no sea el caso de esta versión del proyecto. Sin embargo si que se requiere que ambas máquinas se puedan localizar entre ellas. Esta localización se obtiene mediante dos recursos, la dirección IP (que identifica a la máquina en la red) y el puerto (que identifica al programa dentro de la máquina). El cliente debe obtener de algún modo los parámetros del servidor para que la conexión sea posible. Para esta versión del proyecto, en la que los dos programas se encuentran en la misma máquina, los parámetros son:

- **IP:** *localhost* (es equivalente a la IP 127.0.0.1 que hace referencia a la tarjeta de red de la propia máquina).
- **Puerto:** 5000.

La arquitectura de comunicación por *Sockets* es de tipo Cliente/Servidor. El papel del cliente es iniciar la comunicación mientras que el servidor se encuentra a la espera de que el cliente empiece a comunicarse. Es necesario especificar el momento en el que se cierra la conexión para saber cuando se ha terminado de enviar información. En el caso de este proyecto, el cliente será el programa ConExp-FX (de ahí que el nombre se la clase sea *SocketClient*) y el servidor será el mod ejecutado en la plataforma Minecraft. Así pues, el proceso que realiza esta función será el siguiente:

- Creación de un objeto de tipo *Socket*, en el que se especifican la IP y el puerto del servidor.
- Creación de un objeto *DataOutputStream* que establece la conexión con los datos del *Socket* creado anteriormente.
- Envío del mensaje mediante la función *writeUTF.*
- Cierre de la conexión.

En caso de error en cualquier punto de la conexión, la función reportará un error que se mostrará por consola.

3.2: Mod de Minecraft (FcaMc)

En este apartado se procederá a la explicación del funcionamiento y de la implementación del mod creado para la plataforma Minecraft al que se le ha dado el nombre de *FcaMc,* en referencia a la suma de las siglas en ingles de Análisis Formal de Conceptos (FCA) y Minecraft (MC).

3.2.1: Estructura del mod.

El mod se ejecuta en paralelo a la ejecución del programa de Minecraft. Desde el momento que se inicia Minecraft, si el mod está instalado correctamente, se empezará a ejecutar las funciones del mod de la manera explicada en el apartado 2.3.2 que hace referencia al desarrollo y las fases de ejecución del mod de Minecraft.

El proceso de funcionamiento de Minecraft con el mod FcaMc instalado es el siguiente:

- Al iniciar Minecraft, se ejecutan las funciones contenidas en la función *preInit*. En este punto se crean los bloques que serán utilizados para la visualización de los conceptos formales.
- Se selecciona la opción de juego *Singleplayer.*
- Se crea un mundo nuevo (*Create New World*) que hará ejecutar las funciones contenidas en la función *init.* El mundo nuevo ha de tener las siguientes características:
    - *Game Mode Creative,* para que nos permita una libre exploración por el mundo.
    - *World Type: Superflat,* de este modo el mundo que se genera facilitará la visualización del retículo.
- Se espera a que se reciba el mensaje por parte de ConExp-FX con la información referente al contexto formal.
- Se analiza la información recibida y que creará el nuevo mundo con la visualización del retículo con todos los conceptos formales.
- Se presentará el entorno de juego en que se encontrará el retículo y se podrá interactuar con los conceptos para ver información relativa a su intensión y su extensión.

A continuación se procederá a una explicación de cada una de las clases involucradas en este proceso, así como de las funciones y parámetros necesarios para el funcionamiento correcto del mod.

3.2.2: Especificación del diseño de clases del mod FcaMc.

Del mismo modo que se hizo en el apartado 3.1.2 que explicaba el diseño de las clases en ConExp-FX, en este apartado haremos el mismo procedimiento con el mod FcaMc, mediante una organización de los paquetes y las clases utilizadas.

com.vgarcia.fcamc

En este paquete se almacenan las clases referentes a las fases de ejecución del entorno de Minecraft.

Main.java

En esta clase se encuentran los parámetros que definen al mod como son el nombre (*FcaMC*), un parámetro para la identificación del mod llamado ID (FCAMC) y la versión del mod (1.0.0).

Además también se encuentran las funciones referentes a las fases de ejecución de Minecraft: *preInit, init y postInit.* Todas estas funciones tienen como parámetro de entrada un evento que lanza el propio Minecraft cuando se alcanzan cualquiera de estas fases.

En el interior de estas funciones se encuentra una llamada a las funciones *proxy.* En el caso de FcaMc, todas las funciones *proxy* son del tipo *CommonProxy* ya se han de ejecutar tanto si el mod se está ejecutando en modo de cliente como de servidor.

CommonProxy.java

Al igual que *Main.java,* esta clase también contiene las funciones *preInit, init y postInit,* pero en este caso ya tenemos funciones con utilidades específicas.

En la función *preInit* tenemos la función de *ModBlocks.init* que creará los bloques que se utilizarán para representar los conceptos formales.

Seguidamente se encuentra la función *init*, en la que se creará un objeto de tipo *ModWorldGen* (objeto de mundo modificado) y se ejecuta la función *registerWorldGenerator,* que tiene como parámetro de entrada el objeto creado con anterioridad. Ésta función se encargará de modificar el nuevo mundo creado de Minecraft.

EventHandlerCommon.java

En esta clase se encuentran las funciones orientadas a eventos. Sólo hay una función que requiere de eventos en este proyecto, *onDestroyedBlock.* Esta función se ejecuta en el momento que se interactúa con un bloque propio de un concepto formal y muestra una ventana con la información de la extensión e intensión de un concepto.

Ilustración 27: Ventana con la información del concepto formal

com.vgarcia.fcamc.block

En este paquete se encontrarán las clases necesarias para la creación de bloques personalizados.

BasicBlock.java

Esta clase contiene la información de un nuevo objeto tipo de *BasicBlock* con características propias de un bloque, ya que esta clase hereda de la clase *Block.java,* propia de la librería proporcionada por Forge. De este objeto, las funciones más relevantes para la funcionalidad del proyecto son *setBlockName setBlockTextureName,* que caracterizan el nombre por el que será reconocido el bloque por el mod, y la textura de ese bloque.

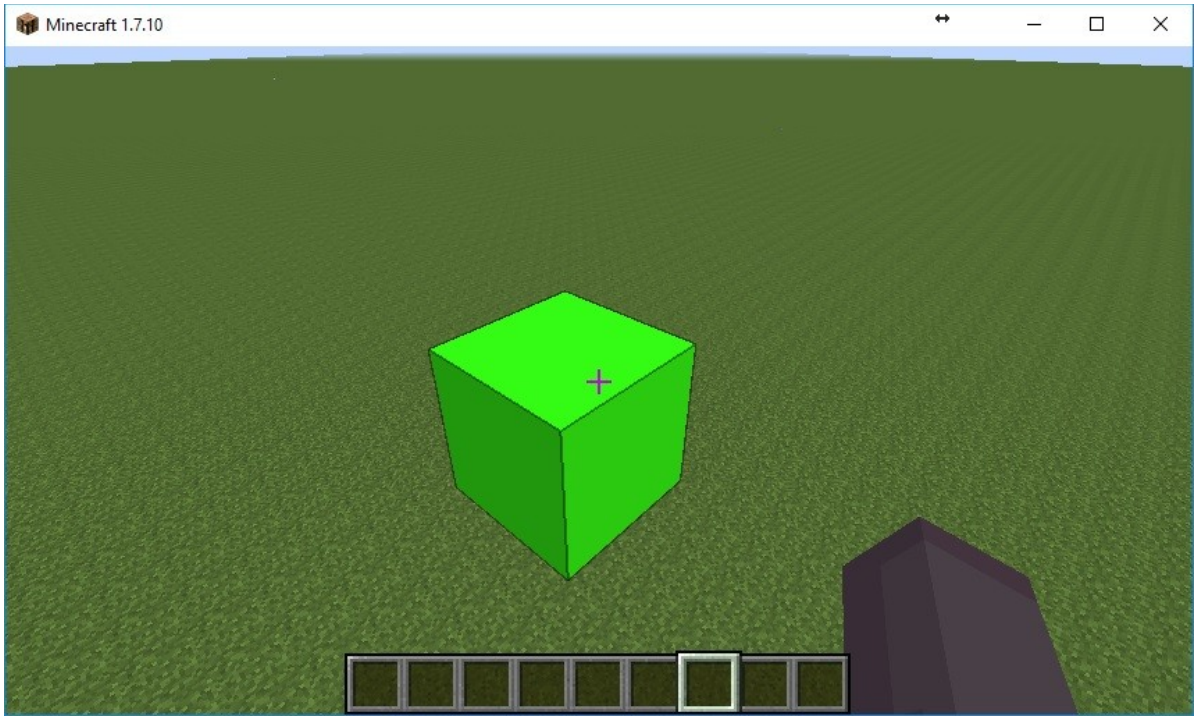

Ilustración 28: Ejemplo de BasicBlock

ModBlocks.java

En esta clase se crean los nuevos objetos de tipo *BasicBlock.* Se han creado un total de 115 bloques diferentes, cada uno con un nombre y una textura distinta. Las 115 texturas han sido generadas automáticamente con un *script* programado en Java.

com.vgarcia.fcamc.world

ModWorldGen.java

Esta clase es la encargada de modificar el mundo creado por Minecraft para incluir en él la representación del retículo conceptual.

Lo primero que establece es la posición inicial del jugador en el mundo, que por defecto es aleatoria, en la posición 0,0,0; así siempre se empezará con esa posición como referencia. Seguidamente se hace la llamada a las funciones *ServerSocektConex.getMsgSocket,* que obtiene el mensaje enviado por ConExp-FX; *JSONContext.listConcepts,* que obtiene una lista de conceptos formales a partir del mensaje recibido y *Shapes.conceptGraph,* que genera el retículo en el mundo de Minecraft.

com.vgarcia.fcamc.util

ColorSpace.java

Para poder tener siempre una referencia de la posición de manera visual de la posición de los conceptos formales. Se recurrió a la siguiente solución, asignar a cada uno de los conceptos formales un color que hará referencia a la posición del color en el espacio de color HSV. De este modo, si se visualiza un concepto formal con mayor luminosidad que otro (tono de color más claro), será porque se encuentra en una posición más alta en la jerarquía del retículo. Los conceptos se representarán con colores más saturados a medida que se alejan del centro del retículo y el matiz del color del concepto dependerá del arco cuya tangente formen las coordenadas x y z.

Para implementar esta solución, en esta clase se encuentran las funciones *getColorFromLocation,* que devuelve el color que corresponde a la posición de cada concepto formal; y *getClosest,* que devuelve el valor de color más próximo al color obtenido y que puede se encuentra entre las texturas predeterminadas.

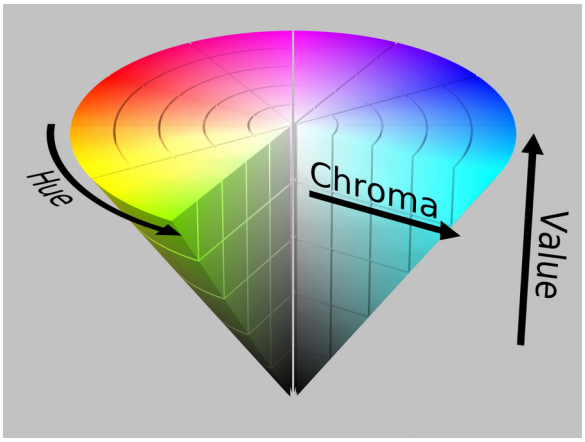

Ilustración 29: Espacio de color HSV

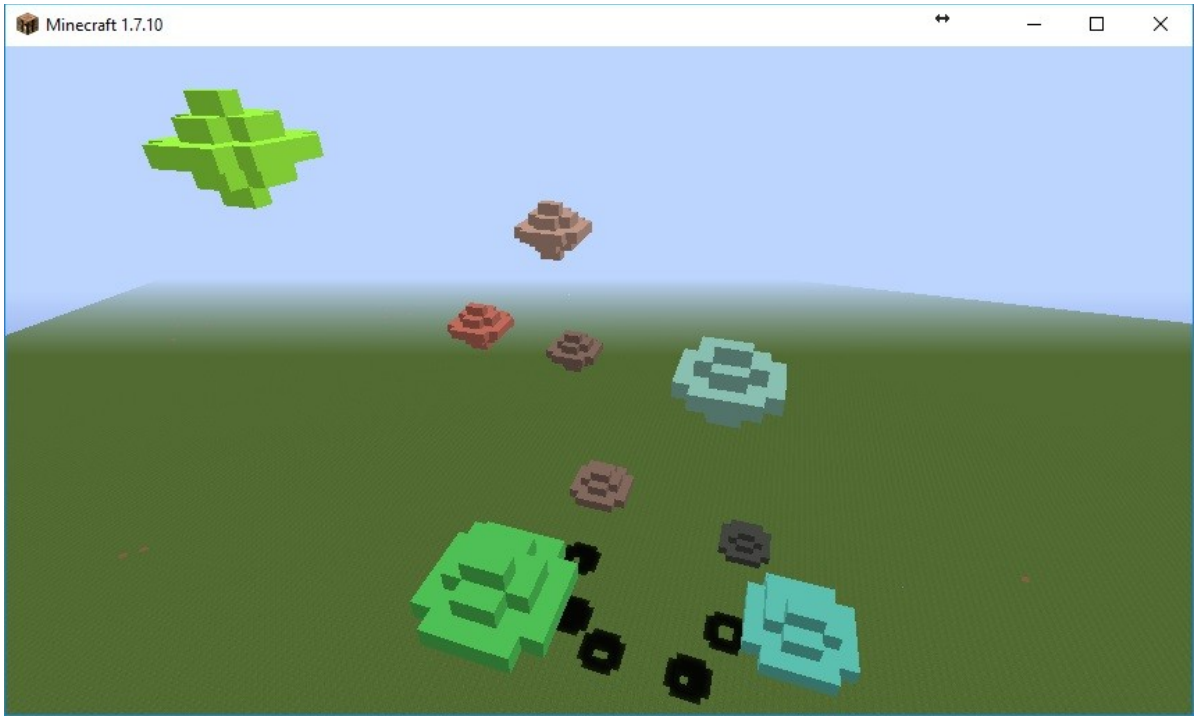

Ilustración 30: Ejemplo de espacio de color HSV en el retículo

JSONContext.java

Las funciones contenidas en esta clase devuelven la información contenida en los textos JSON recibidos de ConExp-FX. Estas funciones son *listConcepts,* que devuelve una lista con la posición de cada uno de los conceptos formales, y *getRelations* que devuelve una lista con la información de extensión e intensión de los conceptos.

ServerSocket.java

Ésta sería la clase espejo de la clase SocketClient de ConExp-FX. Esta clase contiene la función *getMsgSocket* que obtiene el mensaje enviado desde ConExp-FX. El funcionamiento es similar a la clase cliente con la diferencia de que las funciones de escritura del cliente ahora son funciones para la lectura del mensaje.

Shapes.java

En esta clase se encuentran las funciones que definen la forma en los conceptos formales en Minecraft. Para ello se han utilizado las siguientes funciones:

- *circulo:* Esta función genera un círculo en el plano x-z de bloques de tipo *ModBlock*. Tiene como entrada las coordenadas x-y-z en la que se encontrará el centro del círculo y el valor del radio del círculo (en número de bloques).

Para ello se utiliza la función *setBlock*, que coloca un bloque en el mundo en unas coordenadas x-y-z dadas. La función consta de dos bucles anidados, uno aumenta el radio de cero hasta el radio especificado, y el segundo aumenta el ángulo de la circunferencia para crear la cirucnferencia.

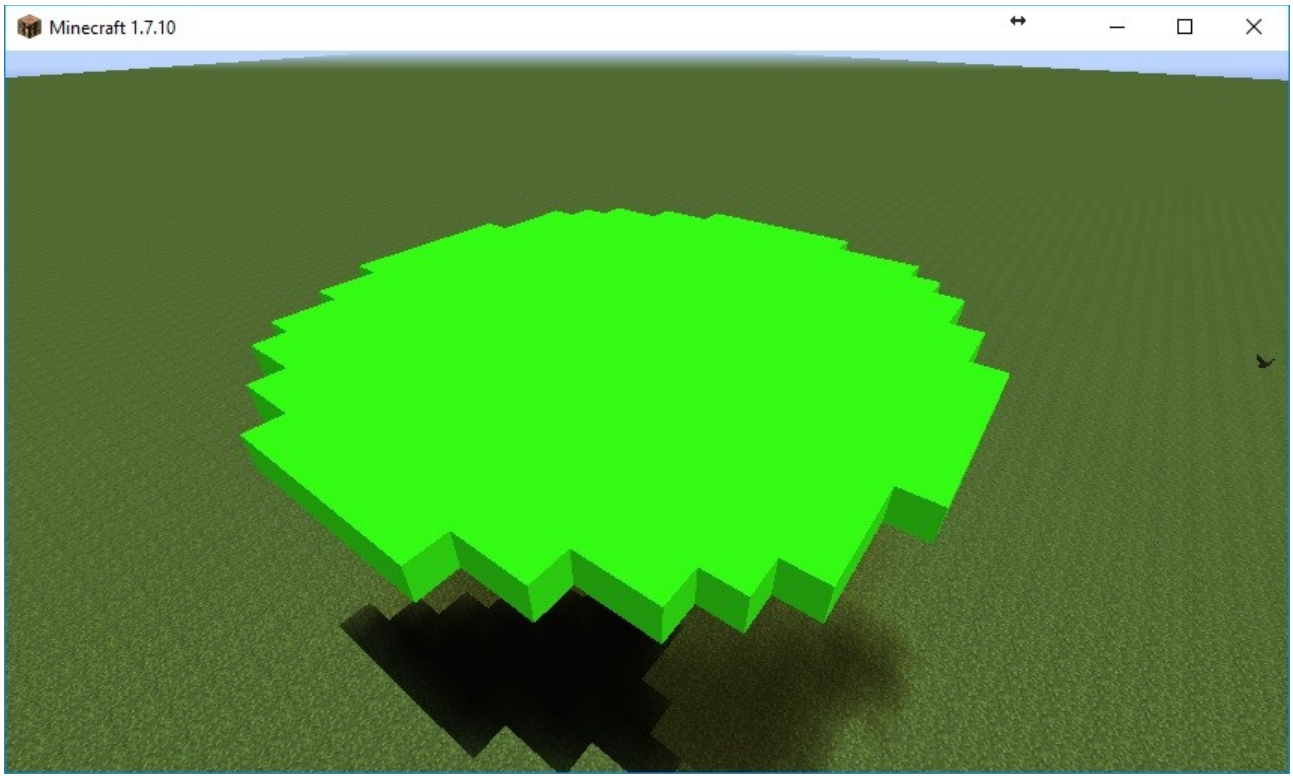

Ilustración 31: Ejemplo de círculo

- *esfera:* Los parámetros de entrada son los mismos que en la función anterior. Genera un círculo del radio indicado en el centro de la esfera y se disponen círculos de menor radio en las posiciones superiores e inferiores del centro hasta llegar a un radio cero.

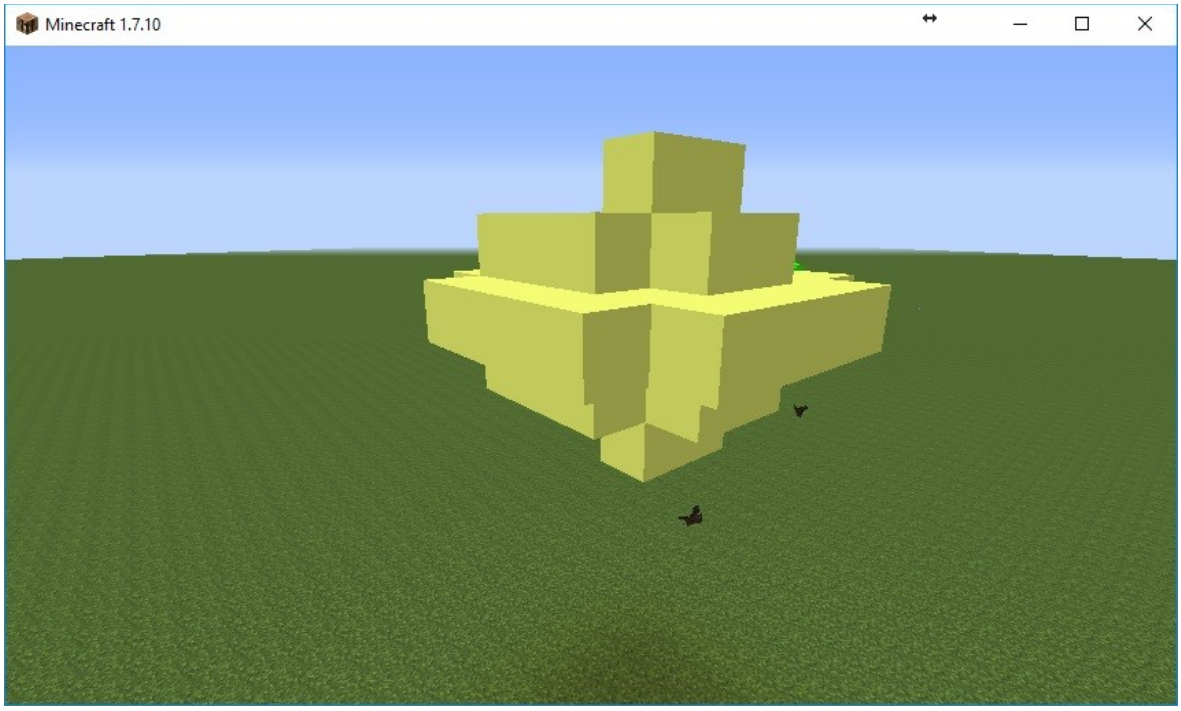

Ilustración 32: Ejemplo de esfera

- *conceptGraph:* Partiendo de la lista de posiciones de los conceptos formales, se dispone una esfera en cada una de esas posiciones, generando así la visualización del retículo.

Ilustración 33: Ejemplo de retículo generado con conceptGraph

3.3: Pruebas y resultados.

Explicado ya todo el desarrollo del programa, se va a proceder ahora a realizar las pruebas de su funcionamiento y un análisis de los resultados de esas pruebas. Para ello utilizará el contexto formal del apartado 2.1 que ha servido de ejemplo para el proyecto, y se harán otras pruebas con contextos formales más complejos para observar el comportamiento del programa.

El primer paso será generar el contexto formal en ConExp-FX y realizar la visualización tridimensional.

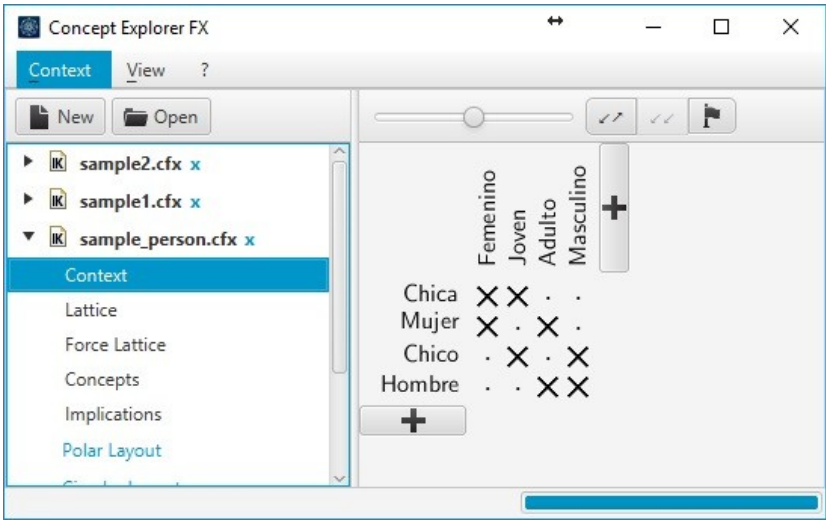

Ilustración 34: Contexto Formal de prueba en ConExp-FX.

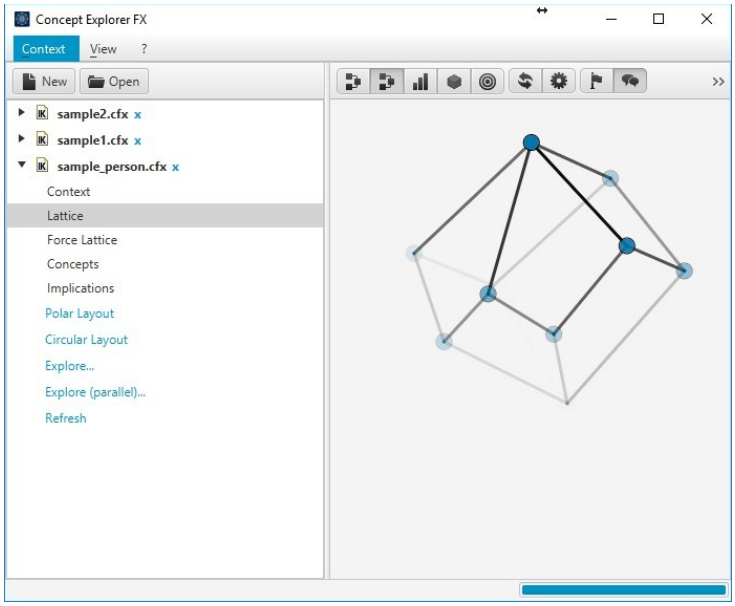

Ilustración 35: Representación tridimensional del retículo de prueba en Conexp-FX

Una vez hecho ésto, se procede a crear el mundo nuevo en Minecraft, y pulsar el botón “Transferir a Minecraft” de ConExp-FX. El resultado es el siguiente:

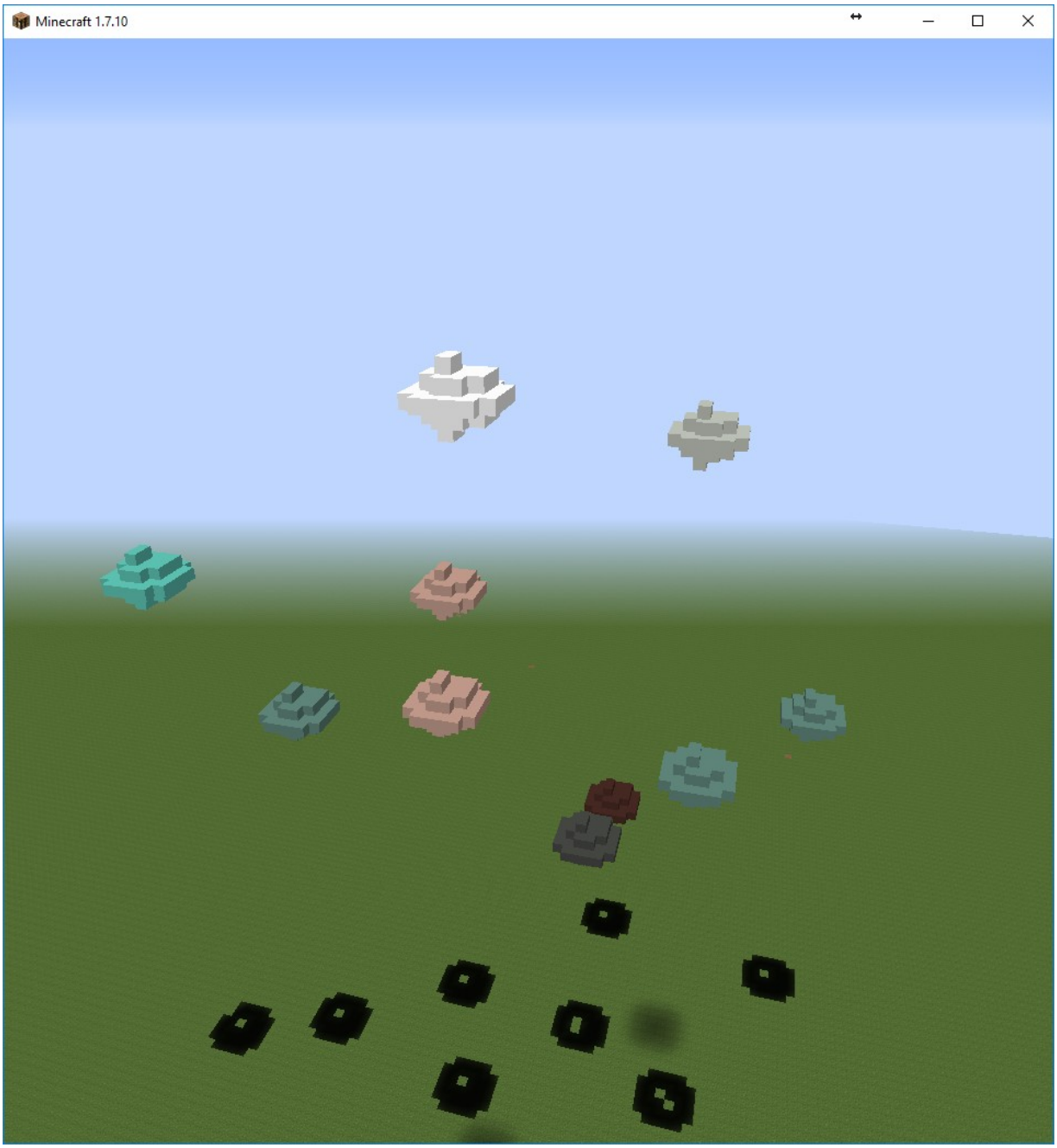

Ilustración 36: Representación del retículo de prueba en Minecraft

A simple vista se puede observar que la forma es similar a la ofrecida por ConExp-FX (puede variar porque el punto de observación de Minecraft y ConExp-FX no tienen por qué coincidir) y se respeta el número de conceptos (10 en total).

Seguidamente se procederá a examinar la información (extensión e intensión) de los conceptos. Se sabe que el concepto situado en lo más alto del retículo (color blanco) es superconcepto de todos los demás conceptos formales, por lo que su extensión deberá ser el total de los objetos del contexto formal. En la siguiente figura se puede observar que el resultado es correcto:

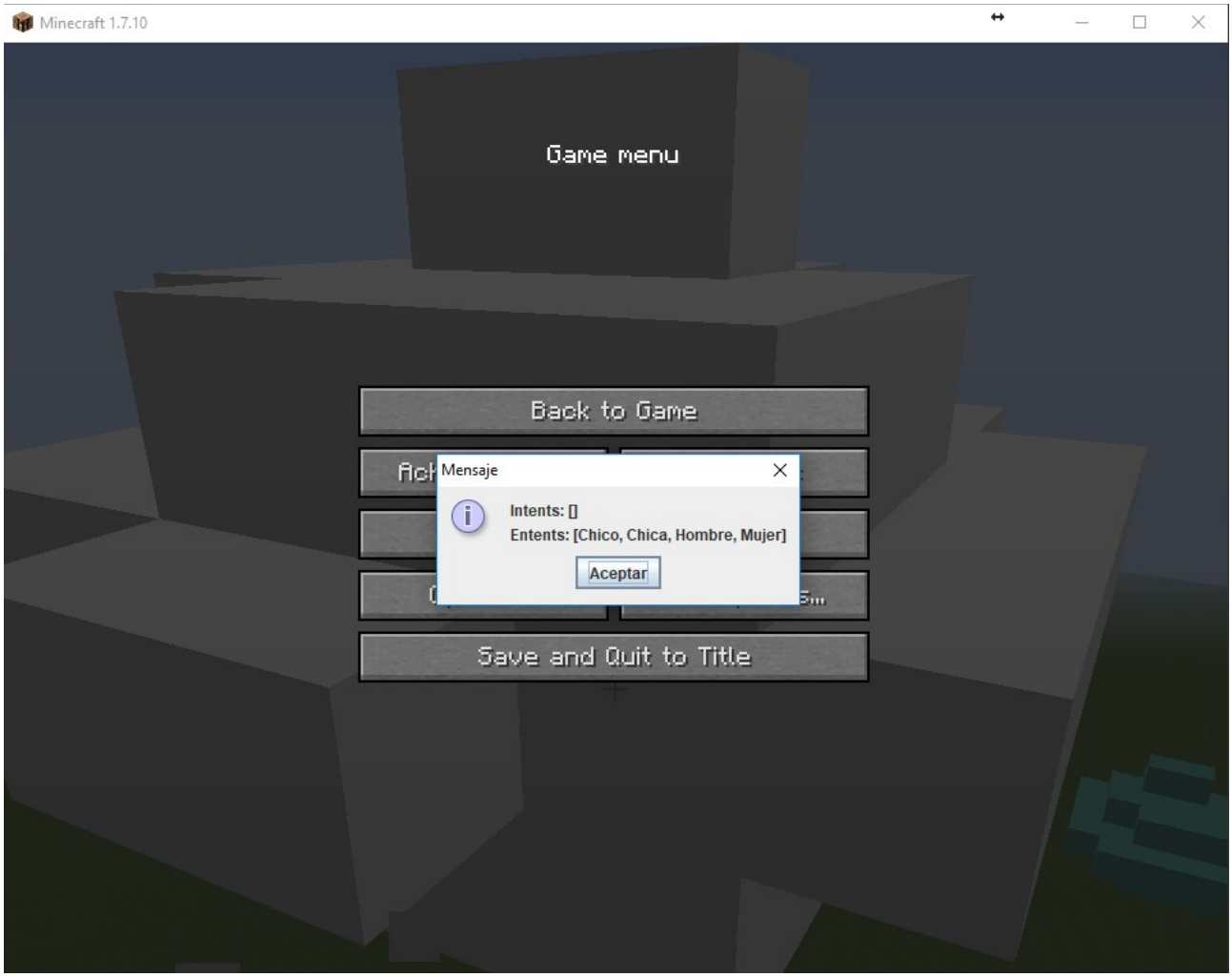

Ilustración 37: Superconcepto del contexto formal de prueba.

Del mismo modo, el contexto más inferior (subconcepto de todos) deberá contener en su intensión todos los atributos.

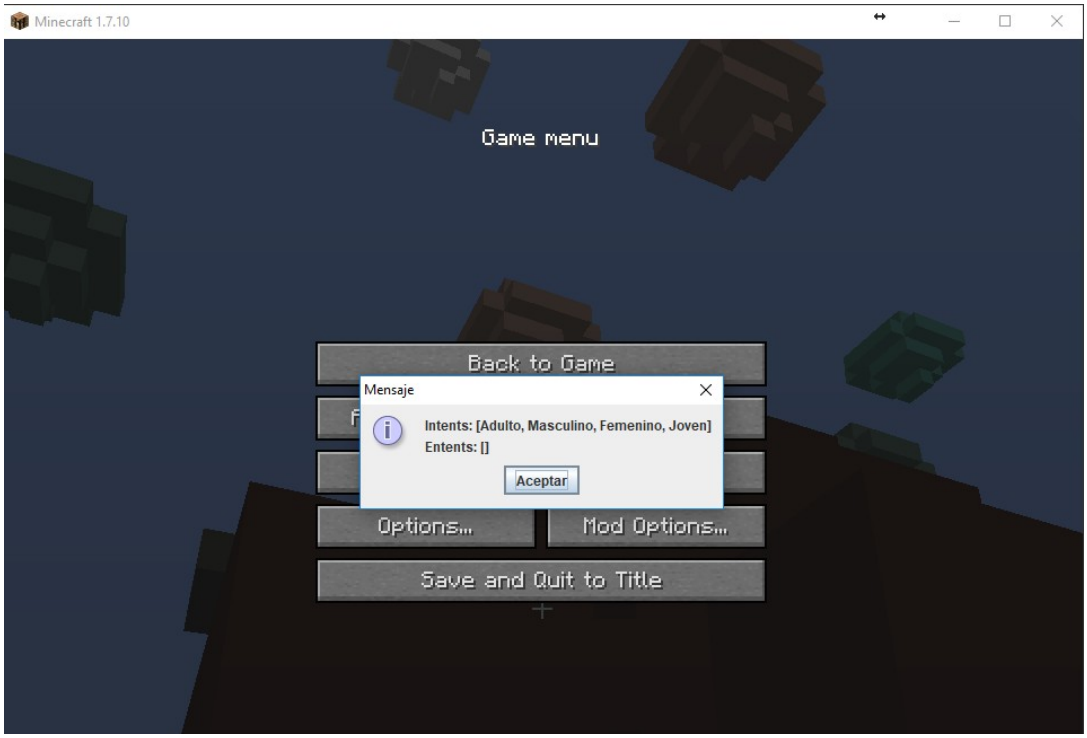

Ilustración 38: Subconcepto del contexto formal de prueba.

El resto de concepto deberá tener una determinada intensión y extensión en función de su posición en el retículo.

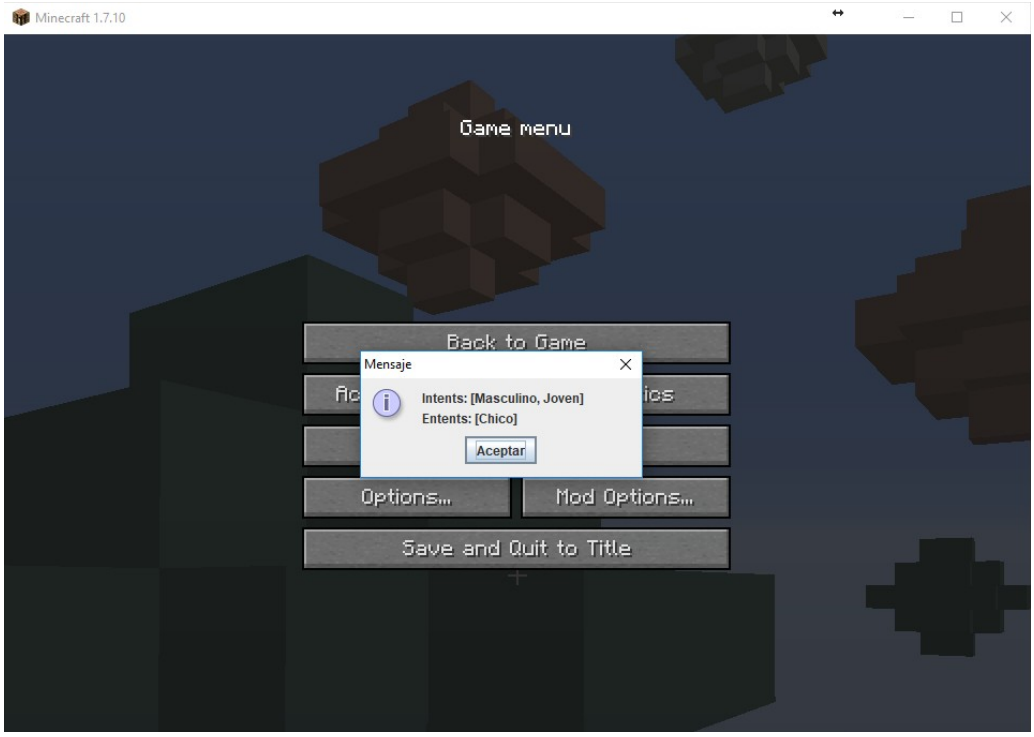

Ilustración 39: Ejemplo de concepto formal de pueba

A la vista de los resultados, se puede decir que la representación en Minecraft es fiel a la representación del retículo en Conexp-FX y que se respeta la información de los conceptos, por lo que el resultado de las pruebas es satisfactorio.

A continuación se muestran otras pruebas para ver el correcto funcionamiento del programa.

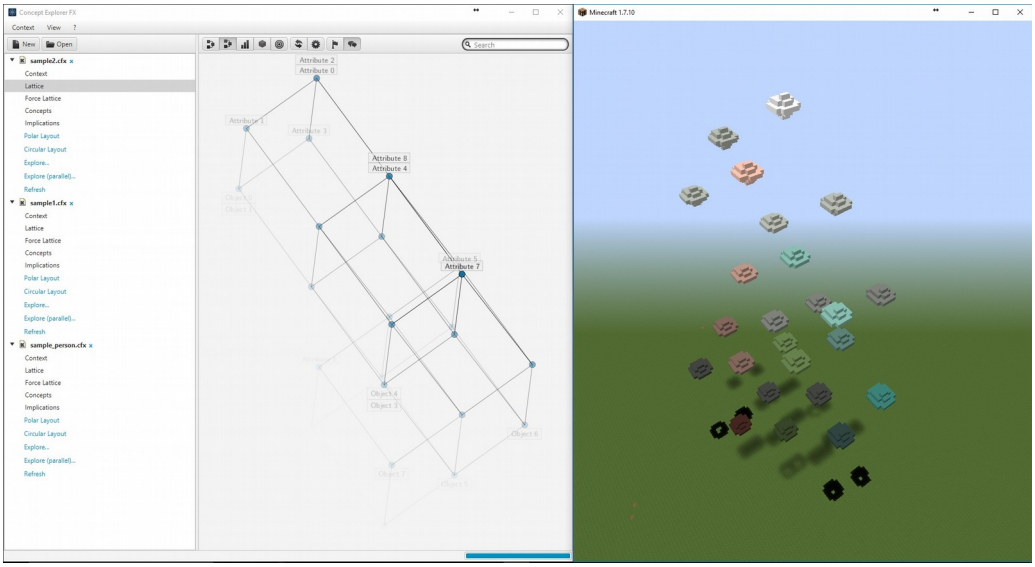

Ilustración 40: Ejemplo 2 de representación de contexto formal.

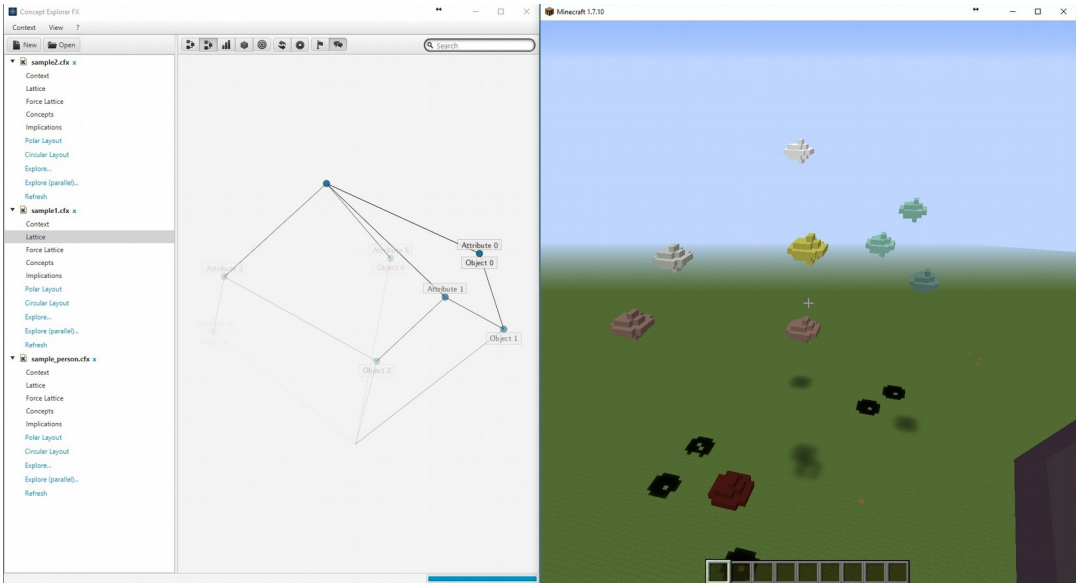

Ilustración 41: Ejemplo 3 de representación de contexto formal.

Capítulo 4: Marco regulador y entorno socio-económico.

4.1: Marco regulador.

El uso del software desarrollado en este proyecto estará limitado al uso privado o educacional, ya que han sido utilizadas herramientas sujetas a derechos de copyright, y su comercialización no está permitida sin el consentimiento del autor como se establece en el Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual.

4.1.1: Conexp-FX.

El software Conexp-FX[10] está sujeto a copyright, perteneciendo los derechos a su autor Francesco Kriegel. Sin embargo, en lo términos de uso se especifica que el software puede ser utilizado para propósitos privados o educativos.

4.1.2: Minecraft.

Minecraft es un software propiedad de la empresa Mojang AB, y para su uso es necesario adquirir una licencia oficial de Minecraft[11]. Entre los términos de uso se establecen unas condiciones para el desarrollo de herramientas basadas en Minecraft. En ellos se establece que se permite el desarrollo siempre y cuando no se utilicen mecanismos que hagan parecer que la herramienta es oficial (como, por ejemplo, el uso del logo oficial de Minecraft). Tampoco se permite el uso de la modificación con propósitos comerciales sin el consentimiento de los compañía.

4.2: Entorno socio-económico.

Como se ha explicado en el apartado anterior, esta herramienta no puede ser comercializada, pero si se puede utilizar con propósitos educacionales. La herramienta resulta práctica para realizar el FCA de una manera diferente al resto de alternativas de software de FCA, por lo que puede ser de gran ayuda en entornos académicos como las universidades.

[10]Enlace al GitHub de Conexp-FX donde se muestran los derechos de copyright de la plataforma: https://github.com/francesco-kriegel/conexp-fx
[11]Enlace a los términos de uso de Minecraft: https://account.mojang.com/terms?ref=ft

El presupuesto de FcaMc sólo constará de la licencia de Minecraft que ha supuesto el único coste económico al desarrollo de la aplicación.

Artículo	Unidades	Coste/unidad (€/ud)	Coste Imputable
Licencia Minecraft	1	23,95	23,95 €
		Total	**23,95 €**

Capítulo 5: Discusión y futuras mejoras.

5.1: Discusión.

El objetivo de este proyecto era realizar una visualización del FCA y para ello se utilizó la plataforma Minecraft como entorno para esa visualización y la herramienta ConExp-FX para la generación del contexto formal. Como resultado se tiene una representación tridimensional del retículo por el que se puede navegar libremente y extraer información de cada uno de sus conceptos.

El mod FcaMc cumple con estos objetivos, cubriendo así, un espacio vacío en el entorno del software de FCA. Hasta entonces, las representaciones eran bidimensionales o la representación tridimensional que ofrecían era una proyección en un plano de dos dimensiones. FcaMc proporciona una navegación inmersiva, sabiendo en todo momento en que parte del retículo nos encontramos, permitiendo así una visualización espacial más natural e intuitiva, además de obtener rápidamente la información de cada uno de los conceptos.

5.2:Futuras mejoras.

El código de FcaMc estará alojado en la plataforma GitHub para que tengan acceso futuros desarrolladores. Éstas son unas de las posibles mejoras para FcaMc:

5.2.3:Implementación de las líneas de unión entre conceptos relacionados.

Existe la posibilidad de trazar líneas en Minecraft entre dos puntos del mundo. Estas líneas se querían implementar para trazar las líneas de unión entre los conceptos formales que guarden relación.

Esto es posible utilizando las librerías del software de gráficos OpenGL[12].Sin embargo, debido a una incompatibilidad entre OpenGL y el controlador gráfico utilizado en la máquina de desarrollo, esta función no se pudo implementar.

[12]Enlace a la página de OpenGl: https://www.opengl.org/

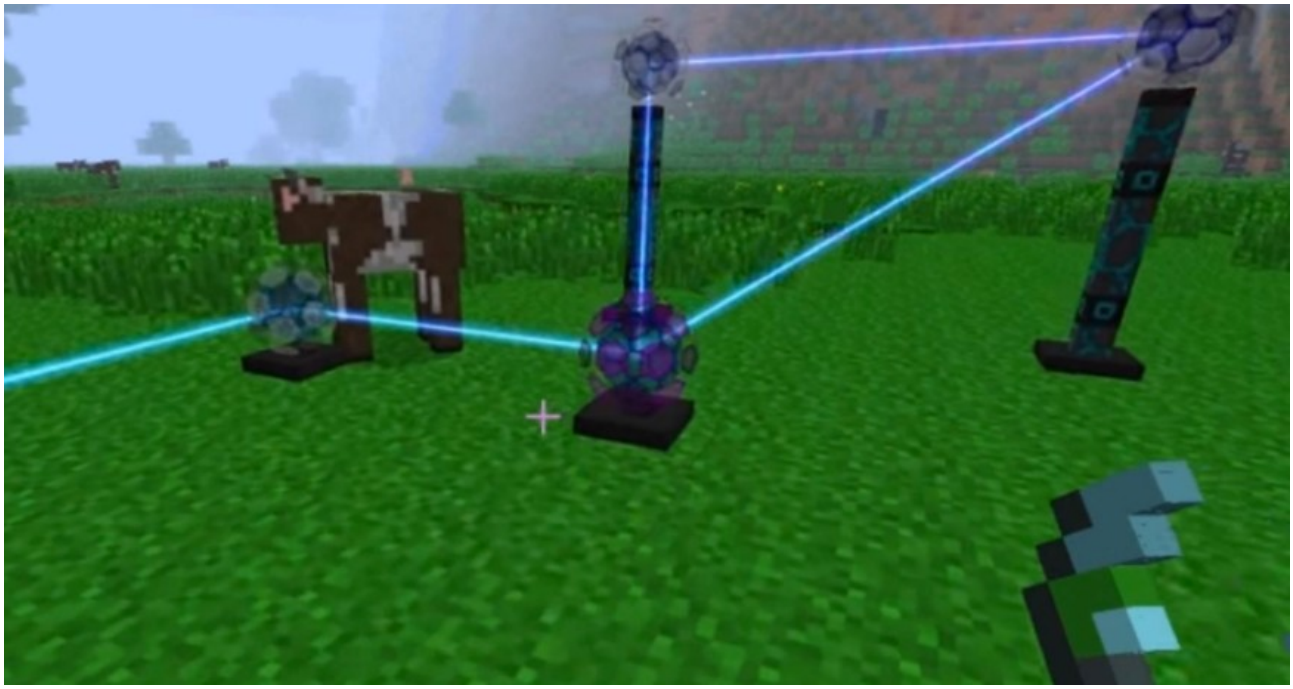

Ilustración 42: Ejemplo de lineas implementadas con OpenGL.

5.2.3: Soporte de realidad virtual.

Minecraft ya tiene soporte para el uso de realidad virtual con las gafas Oculus Rift y Samsung Gear VR. Una posible mejora sería ver el funcionamiento de FcaMc utilizando las gafas y comprobar si el funcionamiento es el correcto o es necesaria alguna implementación adicional.

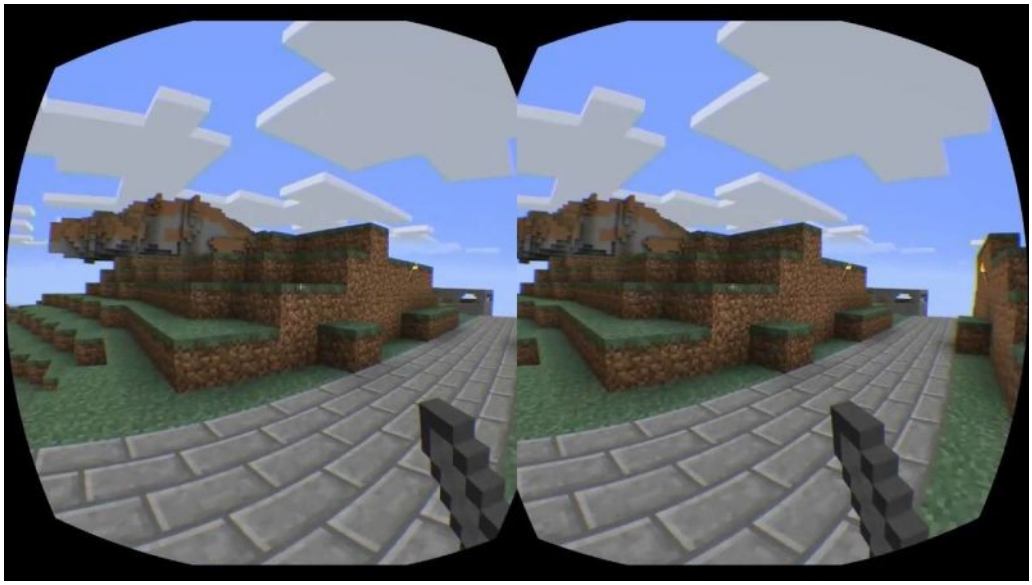

Ilustración 43: Interfaz de Minecraft con realidad virtual.

Capítulo 6: Bibliografía:

1. Sébastien Ferré, Sebastian Rudolph. (2009). Formal Concept Analysis. Alemania: Springer.
2. Bernhard Ganter, Gerd Stumme. (2005). Formal Concept Analysis: Foundations and Applications. Alemania: Springer Science & Business Media.
3. Uta Priss (2008). FcaStone - FCA file format conversion and interoperability software-
4. Francesco Kriegel (2017). ConExp-FX JavaDoc. - https://lat.inf.tu-dresden.de/~francesco/conexp-fx/apidocs/index.html
5. Análisis Formal de Conceptos (22 de marzo de 2016) - https://es.wikipedia.org/wiki/Discusi%C3%B3n:An%C3%A1lisis_formal_de_conceptos
6. Retículo(matemátcas) (22 de mato de 2012) - https://es.wikipedia.org/wiki/Ret%C3%Adculo_(matem%C3%A1ticas)
7. María Naranjo López (2016). Análisis Formal de Conceptos desde el punto de vista de la programación funcional.
8. Uta Priss (2008). FCA Software Interoperability.
9. Bedrock Miner Moding Tutorials - https://bedrockminer.jimdo.com/modding-tutorials/