ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Trabajo Fin de Grado

Desarrollo de un entorno educativo basado en videojuegos online

Development of an educative environment based on online videogames

AUTOR: ANDRÉS FRANCISCO SAAVEDRA CÁNOVAS

DIRECTOR: JUAN JOSÉ ALCARÁZ ESPÍN

FECHA DE PRESENTACIÓN: 07/10/2016

Autor	Andrés Francisco Saavedra Cánovas
E-mail del Autor	anfrasaaca@gmx.es
Director(es)	Juan José Alcaraz Espín
E-mail del Director	juan.alcaraz@upct.es
Codirector(es)	
Título del TFM	Desarrollo de un entorno educativo basado en videojuegos online
Título en Inglés	Development of an environment based on online videogames
Descriptores	Programación, videojuegos, aprendizaje, Minecraft, JavaScript, niños, aulas, entorno, herramienta, jugar, diversión, tecnología, ciencia, futuro, formación.

Resumen

A pesar de la importancia de la algorítmica y la programación en el mundo actual, estos contenidos no están suficientemente representados en el currículo de la educación obligatoria. Del mismo modo, los métodos docentes en estos niveles educativos siguen estando predominantemente basados en clases magistrales y pruebas de evaluación clásicas (exámenes) que otorgan gran relevancia a la competencia memorística. La adopción de metodologías basadas en proyectos, experimentación y fomento de la creatividad es muy lenta, y más aún la combinación de éstas con las nuevas tecnologías y herramientas innovadoras para motivar al alumnado como los videojuegos online. Un ejemplo de este tipo de videojuegos es Minecraft, convertido en fenómeno sociológico para toda la generación de niños y jóvenes en los distintos niveles de educación obligatoria. Este videojuego fue recientemente adquirido por Microsoft por 2500 millones de dólares, que identificó su potencial más allá del ocio y que recientemente lanzó un portal http://education.minecraft.net específicamente diseñado para ofrecer diferentes herramientas y tutoriales para profesores. En este proyecto se pretende explorar las posibilidades que ofrece Minecraft para el aprendizaje de la algorítmica y la programación entre los alumnos de primaria y secundaria.

Titulación	Grado de Ingeniería Telemática
Intensificación	Ingeniería Telemática
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	07/10/2016

ÍNDICE

1. INTRODUCCIÓN

La realización de este proyecto tiene como objetivo desarrollar las bases de un entorno educativo que acerque la enseñanza de los conceptos básicos del mundo de la programación a las aulas, permitiendo así que los niños y niñas comiencen a conocer estas nociones a edades más tempranas a través de un entorno que les es conocido, en el cuál, mediante pequeños juegos absorban esos conocimientos de una forma mucho más amena a la tradicional.

MOTIVACIÓN

El avance tecnológico que se ha formulado en los últimos años es innegable, pero aún más lo es el hecho de que esta progresión no se detendrá. En los años venideros el mundo de la tecnología será imprescindible y parte de un todo. Por este motivo, el desarrollo de los más pequeños en el ámbito de la ciencia y la ingeniería es primordial. Si bien, las matemáticas, la física o la química siempre han sido contenido habitual en las aulas, la programación no ha corrido la misma suerte. Probablemente, se deba a que esta última disciplina es relativamente reciente (último siglo) mientras que las demás siempre han estado presentes.

El hecho de que esta situación varíe es algo necesario y que debería realizarse con la mayor prontitud. Por supuesto, es una tarea ardua, ya que hay cuestiones que solo el ensayo y error sabrían resolver, detractores que se oponen y una necesaria renovación en la formación académica del profesorado.

Los conceptos básicos de la programación son muy abstractos, no son tangibles, por eso, es muy complicado que un niño los comprenda. De modo que se presenta la necesidad de introducir estas nociones aunque a la vez el modo de alcanzarlo luzca complicado. Este proyecto pretende colaborar en la realización de esta tarea. Para ello, su principal baza consiste en hacer uso de un entorno, representante de un mundo virtual, conocido por los alumnos. En el cual presentar estas ideas, añadiendo el valor de que ese entorno se trata de un videojuego con el que los niños se divierten y que despierta su interés.

Es de sobra conocido que el cerebro de los niños es como una esponja, que aprenden mucha información y de forma muy rápida. Con la llegada de las tecnologías este hecho no ha cambiado, es más, se ha visto incrementado a un gran nivel. Prueba de ello es un estudio realizado por AVG Internet Security, que revela que en la actualidad un 19% de los niños (2-5 años) saben hacer uso de aplicaciones sencillas de un teléfono inteligente mientras que solo el 9% sabe realizar una acción tan cotidiana como es atarse los cordones de los zapatos sin ayuda.

Es fácil elucubrar cuáles son las razones de este suceso. La facilidad de manejo de un dispositivo móvil simplifica la relación entre sujeto y objeto. Controlar la funcionalidad de una aplicación mediante el contacto de un solo dedo con la pantalla elimina barreras debido a la inmersión que suscita en el usuario. El otro factor es que precisamente debido a esta circunstancia de tener que mostrar toda la actividad posible en la pantalla hace que el diseño de la app requiera de ser muy intuitivo. Estas dos características desencadenan que los más pequeños dominen con gran habilidad un teléfono inteligente aunque no sean capaces de leer y no haya intervención alguna de ningún adulto.

Conseguir transmitir los conocimientos básicos sobre programación a los niños es la finalidad de este proyecto. Las principales adversidades son la abstracción de estas ideas y despertar el interés del alumno. La solución estudiada pasa por hacer de esta experiencia un juego en el que los niños se entretengan y diviertan mientras van adquiriendo estos conceptos, los cuáles aunque complejos de entender, al representarlos en un mundo, que aunque virtual representa algo muy parecido a la realidad, favorece la labor de comprensión que se presenta como uno de los principales problemas en la finalización de este propósito.

La situación actual respecto a la enseñanza del mundo de las TIC es que se enseña cómo trabajar con el ordenador pero no cómo trabaja un ordenador. Generalmente, la asignatura de informática suele centrarse en la utilización de los programas de escritorio

más comúnes (editor de textos, hojas de cálculo, navegadores, etc…). Ni siquiera se explican los sistemas binarios o hexadecimales. Por este motivo, la realización de este proyecto supone un aliciente por su posible futura implantación o al menos servir como primer paso para lograr que se enseñe a programar en las escuelas.

El hecho de acercar a los niños a los conocimientos de programación será motivo de celebración para la industria tecnológica. Supondrá una solución a largo plazo para demoler la incongruencia entre el número de personas cualificadas y el número de trabajos que el avance tecnológico está demandando.

Poco a poco se está prosperando respecto a esta tarea, Estonia es uno de los países de la UE que apoya con mayor intensidad este fin, casi cualquier trámite se puede realizar telemáticamente y esto deriva en su intención de introducir en las escuelas primarias la programación. En Reino Unido o Francia en el año 2015 se implantó esta asignatura en sus cursos académicos. La Comisión Europea ha estimado que para la segunda década de siglo unos 900.000 puestos relacionados con las TIC se habrán de cubrir. Por ello se están introduciendo estas asignaturas en los planes académicos gradualmente.

La vicepresidenta de la Comisión Europea y responsable de la Agenda Digital para Europa, Neelie Kroes, y la Comisionada de Educación, Cultura, Multilingüismo y Juventud, Androulla Vassiliou, enviaron una carta conjunta a los Ministros de Educación de la UE instándoles a promover la enseñanza de la programación informática en las escuelas pues se considera que “la programación es parte de la solución al desempleo juvenil en Europa”.

Por tanto, se manifiesta la necesidad de formar a los jóvenes, no solo cuando ya han decidido enfocar su vida laboral hacia una rama relacionada con la tecnología. Sino que la programación se considere una parte fundamental de la formación académica del alumno desde el inicio de la misma.

En lo referente a España, también se ha iniciado el proceso y por ejemplo, en Madrid el curso pasado, 130.000 alumnos de la ESO impartieron una asignatura denominada ‘Tecnología, Programación y Robótica’, en la que aprendieron como crear una página web o a manejar una impresora 3D entre otras dinámicas. Se establecerá como obligatoria (hasta 4º) de forma progresiva en los próximos años en todos los institutos tecnológicos y los centros públicos, privados y concertados de la comunidad autónoma.

No basta con añadir una asignatura en los colegios, es preciso reunir las condiciones necesarias para poder impartirla, veáse, tener el equipo necesario, que al tratarse de un ambiente tecnológico va a requerir de un desembolso mayor al habitual y consensuar un claustro capacitado.

El primer punto, no supone un gran problema. Si bien, el presupuesto dedicado a educación se ha visto reducido debido a la crisis económica, el impacto que resultó de

la aparición hace unos años de las llamadas 'pizarras inteligentes' tuvo como resultado que los centros se renovarán tecnológicamente, aunque esto iba destinado a cualquier tipo de aula, las dedicadas a tecnología se vieron reforzadas por derivación. Así que, las clases se inundaron de portátiles, proyectores, tablets, etc… El gasto en software es casi nulo. La mayoría de herramientas son open source y como se ha expuesto el hardware ya se posee.

El otro punto, es más escabroso. Cuando se anunció la introducción en las aulas madrileñas de la programación se plantearon dudas sobre si los docentes estarían cualificados. Ya que, en Reino Unido si se ha destinado presupuesto para este fin (cerca de 1,4 millones de libres además de capital de empresas privadas como Google o Microsoft). Por ahora, la solución dada en la implantación en la Comunidad de Madrid consiste en dejar este papel a los profesores que impartían hasta ahora la asignatura de Tecnología y la puesta en marcha de bastantes cursos para lograr profesores competentes, impartidos por empresas como IBM, bq o Telefónica.

OBJETIVOS DEL PROYECTO

Se pretende desarrollar una herramienta que sea útil, sencilla de implementar y que no requiera de una gran cantidad de recursos, cuya función principal sea permitir educar en los fundamentos de la programación a los más pequeños de modo fluido y divertido mediante pequeños desafíos y juegos que mantengan despierto el interés del alumno y faciliten la asimilación de estos conceptos tan abstractos al representarlos o ejecutarlos sobre entidades físicas, debido a que son conocidas por el receptor.

Lo que se ha realizado en este proyecto es el origen de lo que podría consolidarse como una gran herramienta educativa que sea utilizada en el futuro para diversas temáticas como podría ser la historia, las matemáticas o la educación vial, entre otras, gracias a la maleabilidad de su aspecto visual. Ya que, se hace uso de un entorno que fue creado con la finalidad de ser un lienzo en blanco en el que crear distintos universos.

En nuestro caso, se ha focalizado la atención didáctica sobre la programación, ya que se considera ineludible un futuro de los que ahora son niños en los que los dispositivos no estén presentes en cualquier parte. Debido al fenómeno bautizado como Internet de las cosas (IoT), esto ya se está dando y la trascendencia del desarrollo de la tecnología cada vez será más crucial. La programación es elemental en este ámbito como también lo son las matemáticas o la física, pero la enseñanza de éstas está estandarizada y se oferta en cualquier programa educativo. Son obligatorias hasta un determinado curso, de modo que ya están integradas en la formación académica básica de cualquier alumno. Por suerte, respecto a la programación esto también se está dando poco a poco pero a diferencia de las asignaturas antes mencionadas la metodología de ésta está todavía por establecer debido a la juventud y novedad de la temática. Por eso, se están buscando y probando distintas maneras de afrontar este asunto. Algo que se debe tener muy en cuenta es que el desarrollo de este proyecto se ha realizado sin ninguna cualificación en lo referente a la enseñanza y pedagogía. Por tanto, este aspecto será bastante débil y se asume que el resultado final es un pilar sobre el que otros podrán apoyarse y verter sus conocimientos para perfeccionarla y así poder conseguir y ampliar un material educativo que cubra todas las necesidades buscadas y enriquecerlo con un estudio constante.

Considerar tan positivamente las posibilidades que ofrece este proyecto se basa en las múltiples alternativas que se pueden perfeccionar a partir de la base que se muestra en este texto. Los límites que imponen los elementos empleados en este desarrollo existen, sin embargo, las posibilidades son inmensas. Más aún si se incluyera la experiencia de un pedagogo experto, lo cual ampliaría horizontes. La combinación de estos conocimientos con el trabajo de un programador daría lugar a interesantes resultados que exprimirían al máximo los beneficios que puede otorgar esta idea de aprender jugando. También es cierto que debido a encontrarse en una etapa temprana

probablemente sean necesarios varios intentos hasta poder madurar una idea que se pueda implementar en un aula de trabajo.

METODOLOGÍA

La idea de llevar a cabo este proyecto surgió a partir de un artículo publicado en el portal Github por un programador irlandés llamado Walter Higgings y que tiene por título: "La guía de programación en Minecraft para inexpertos". En él, se explican los pasos para preparar un pequeño servidor y efectuar una incursión en la programación, un 'Hola Mundo' realizado en Minecraft. Sin embargo, este texto no explica nada relacionado con incluirlo en el aula de un colegio o aplicarlo con niños de corta edad.

A partir de lo declarado en la publicación se elaboró una idea de cómo integrarlo en una clase de primaria y/o secundaria. Se planteó la visión técnica de la instalación de los componentes utilizados en este proyecto y se orientó todo el desarrollo en base a la visualización de su implantación en un aula. Se priorizó el punto de vista del educador, ya que se pretende mostrar lo que se requiere para poder hacer uso de ello en una clase. La opinión del alumno no ha estado tan presente, no se ha pretendido hacer la mejor y más divertida experiencia para aprender.

El propósito es ofertar al docente la mayor cantidad posible de alternativas para poder experimentar, hacer testeos con diferentes grupos de alumnos, permitir la inspección de todas las alternativas, lo que llevaría al perfeccionamiento de la funcionalidad del aprendizaje. Pero como punto de partida, sin que haya intervención por parte del profesor, se presenta un plan que pretende poseer cierto criterio educativo, dentro de las posibilidades.

Aunque se está tratando de crear un germen que pueda ser destinado a evolucionar y obtener un método didáctico estable, también podría ser aprovechado por un docente que carezca de los conocimientos sobre programación necesarios para ello o la otra vertiente, que un programador (carente de experiencia docente) trate de ir un nivel más allá y seguir incorporando funcionalidades. Por tanto, pese a que el máximo potencial se obtendría de la combinación de la pedagogía y la programación, cada uno por separado también puede desplegar sus propias ideas y lograr metas útiles y funcionales, si bien, no tan fructíferas como en conjunto.

Este trabajo ha madurado con la intención de utilizarlo en un aula, con un profesor y varios alumnos. Obviamente, no se ha podido probar esto en la vida real. No se disponía de una clase, ni material tanto humano como tecnológico. Por ello, gran parte de todo lo que se va a exponer es teórico. A lo largo del progreso del proyecto han aparecido diversas ideas que una vez analizadas se han descartado por límites técnicos o por las dificultades que podría suponer enfrentarlas a un niño. Todo lo figurado que se ha considerado factible es aquello que se explicará a continuación.

Los pasos que se han seguido han sido tres:

1. **Documentación de las propuestas existentes y estado de la técnica actual.** Se investigó qué se ha hecho hasta ahora en el tema que nos incumbe, cuál era la situación de la programación en las escuelas, qué métodos se han utilizado, las características de las mismas y los motivos que justifican su utilización frente a otros.
2. **Puesta en marcha de un servidor para programar extensiones de Minecraft con un intérprete para realizar scripts.** Descarga e instalación del software necesario. Siguiendo los pasos indicados en el artículo, crear y configurar un pequeño servidor sobre el que poder ejecutar las ideas que se pretenden plasmar en un aula y explorar las distintas opciones que brinda el servidor.
3. **Planteamiento y desarrollo de experiencias educativas orientadas a distintas edades.** Idear diferentes actividades con las que tratar de transmitir los conocimientos básicos de la programación a distintos niveles de dificultad. Fomentar la participación del alumno con una serie de desafíos y ejercicios que despierten su interés por aprender.

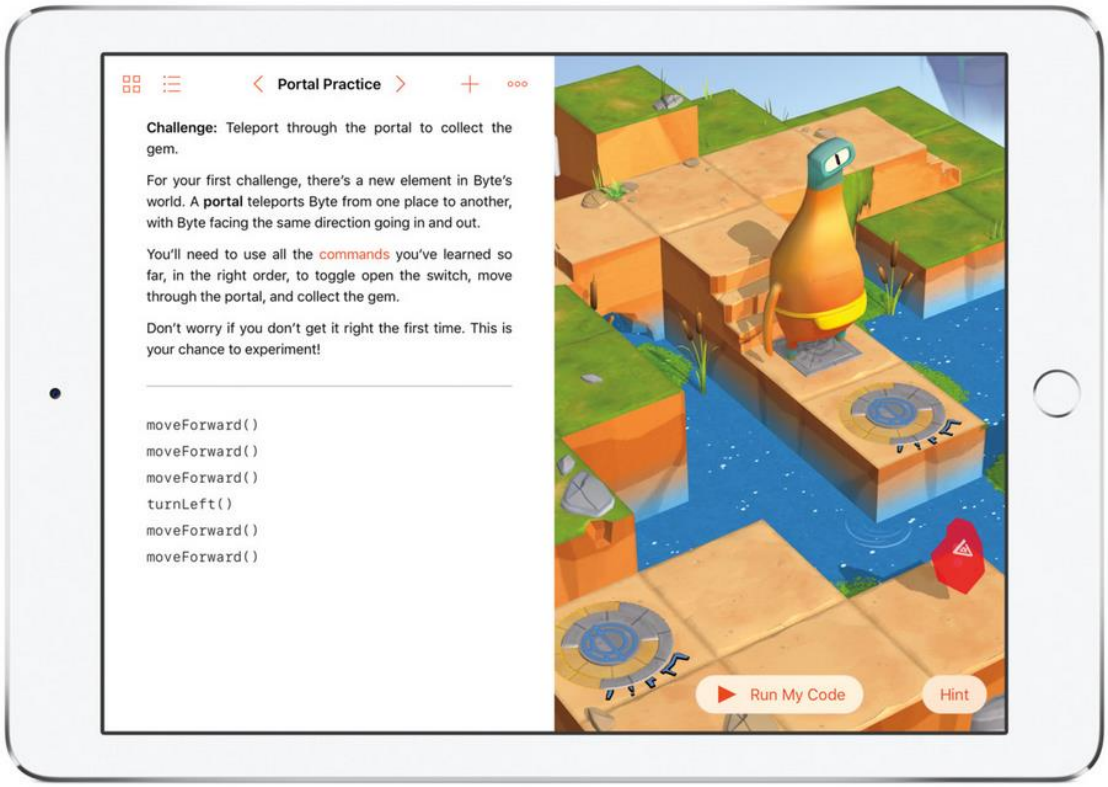
ESTADO DE LA TÉCNICA

En este apartado, se expondrá la situación actual de los métodos y herramientas utilizados para enseñar programación a los más pequeños. No solo las que se pueden usar en un aula sino también en el hogar. Al final se explicará un programa educativo propuesto por Complubot, un centro de robótica educativa.

Algunas de los métodos más interesantes a la hora de introducir a los niños en la programación son:

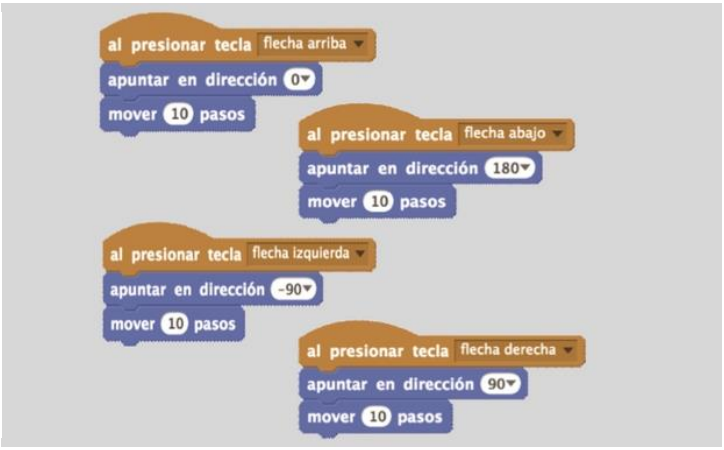
1) **Code-A-Pillar**. No se trata de un entorno de programación, en el que se escriba código con un teclado. Se trata de un juguete con forma de oruga formado por módulos que se iluminan, se interconectan entre sí y que realizan una única función (moverse hacia delante, izquierda, quedarse quieto…) Este gadget infantil ha sido desarrollado por la empresa Fisher-Price y está destinado para edades entre los 3 y 8 años. Permite la asimilación del concepto de programación de estructura secuencial. Se puede tratar de hacerle recorrer un circuito por la habitación. Cuenta con una app que plantea distintos desafíos y cuesta unos 50 dólares.

2) **Swift Playground**. Es la apuesta de Apple, en la que aprender Swift escribiendo código directamente en un iPad. Es una app en la que se muestran distintas opciones de código que se deben seleccionar para llevar a cabo el desafío que se puede visualizar en otra parte de la pantalla. Por lo que se facilita la comprensión de lo que se está haciendo, de este modo es accesible a niños, aunque también los adultos pueden aprender nociones gracias a esta aplicación.



3) **Scratch.** Es un lenguaje desarrollado por el MIT y financiado por empresas de gran importancia en el mundo tecnológico como Intel, Dell, Microsoft o Google. Es gratuito y está disponible en español, tiene versión de escritorio y web. Permite crear desde presentaciones hasta juegos. Debido a su atractivo es usado no solo por niños sino también por académicos, adultos… aunque su rango de edad objetivo es de 8-16 años.

   Se presentó en 2005 y no ha parado de crecer y extenderse debido a su fácil uso y versatilidad. De hecho, hasta han aparecido extensiones para utilizarlo con robots de Lego Mindstorms o placas Arduino. Se trata de un lenguaje visual y modular, en el que se realizan las funcionalidades sobre un personaje, por lo que facilísimo de utilizar y es muy llamativo. Así de simple es dirigir con las flechas del teclado a nuestro personaje:



   Además, profesores y educadores han creado un portal propio, ScratchEd, para compartir ideas e impresiones sobre la utilización de scratch en la docencia. Por último, mencionar **ScratchJr**, una variante más simple para niños de entre 5-7 años que consiste en una app para iOs y Android que facilita el acceso al utilizarse sobre pantalla táctil y ser necesario solo un dedo para controlarla en lugar de un ratón.

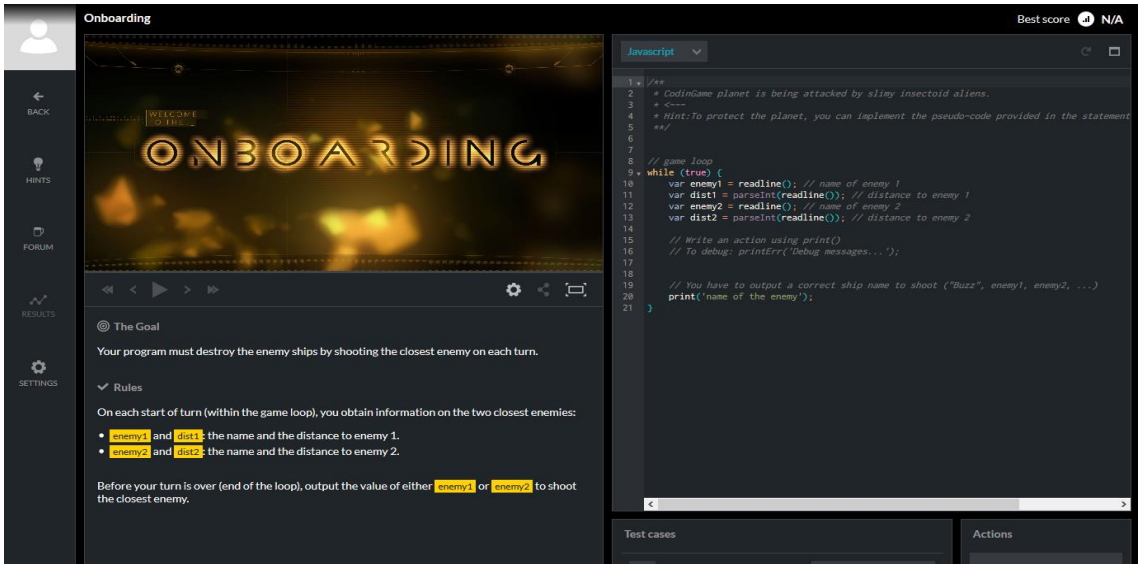
4) **WeDo**. Creada por LEGO Education, se trata de un kit de robótica que pretende educar en las nuevas tecnologías y los estándares de la ciencia puede ser la recolección de pruebas, la realización de investigaciones y el diseño de prototipos. Incluye un SmartHub, una unidad programable que hace uso de Bluuetooth, varios servomotores, sensores y 280 elementos de construcción.

   WeDo usa una interfaz muy sencilla basada en el concepto de arrastrar y soltar iconos para introducir los comandos. Permite una descarga gratuita para que los docentes puedan probarlo. Respecto a este punto, LEGO proporciona un plan de estudios de más de 40 horas con lecciones que

abordan las mecánicas más esenciales de la ingeniería. La intención de la compañía es lanzar una licencia para centros de estudio y así poder aprovechar WeDo, se calcula que costaría alrededor de los 2000 dólares equipar un aula.

5) **CodinGame.** Esta plataforma se aleja un poco del público objetivo ya que requiere de poseer cierto conocimiento previo pero se apoya en el concepto de aprender jugando. Consiste en un juego que presenta diversos ejercicios y desafíos de programación y ofrece una respuesta visual tras completarlos. Se puede elegir entre 23 tipos de lenguajes distintos y ayuda a comprender sobre la inteligencia artificial y otra baza interesante es que permite a los jugadores enfrentar su código en un entorno multijugador para comprobar cuál es más eficiente.

   Además, se accede a él con un simple navegador, el código se compila en la propia ventana y de esta forma en juego se visualiza en tiempo real mientras se programa. Hasta ahora cuenta con un gran impacto y más de 230000 personas ya hacen uso de esta plataforma online.

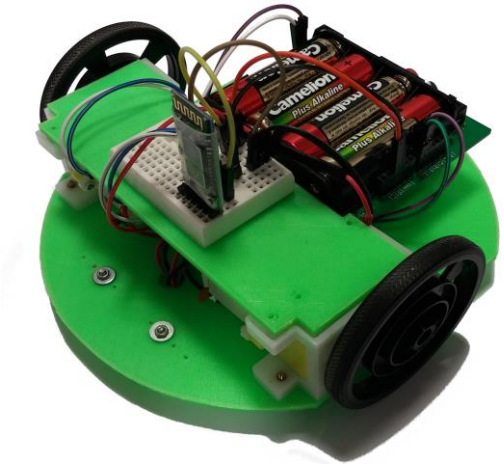


Se ha mencionado en varias ocasiones a LEGO y es que el apoyo que vierte esta empresa sobre los jóvenes para que se internen en la ciencia y la tecnología es enorme. Ejemplo de ello es la FIRST LEGO LEAGUE, una competición en la que se ponen a prueba el ingenio, la colaboración y el respeto. Cada año se propone un desafío con una temática central (ecología, protección de animales, energías renovables…) y además de un proyecto científico a defender ante un tribunal se plantea un juego en el que un robot diseñado (a partir de un kit de LEGO mindstorm) por los alumnos debe superar una serie de pruebas. Este robot consta de un armazón creado a partir de bricks, motores y sensores y los alumnos deben programarlo para conseguir la mayor puntuación en un circuito. Es una competición internacional y que cuenta con la UPCT como sede de celebración de una ronda provincial.

Como ya se ha mencionado, se ha creado una asignatura para instruir en programación implantada en los institutos de secundaria de la Comunidad de Madrid, ésta, al ser de reciente creación cuenta con un plan de estudios que no es estable y está sujeto a cambios en los próximos años. Por este motivo, para mostrar cómo podría estructurarse he decidido apoyarme en una propuesta educativa realizada por Complubot, un centro de enseñanza de robótica con 13 años de experiencia y que cuenta con ingenieros, informáticos, psicólogos y profesores. Aunque al principio su campo de trabajo se limitaba a actividades extraescolares, sus métodos se fueron introduciendo en centro educativos de España y Latinoamerica. Por tanto, esta propuesta, denominada Staring with Robotics, aunque no es oficial sí que cuenta con el respaldo de la experiencia de la compañía y sus trabajadores.

Para 1º de E.S.O (12-13 años), han creído conveniente que se inicie la formación con Scratch, que permite desarrollar la creatividad, el pensamiento lógico e introducirse en la programación. A parte, usar una serie de kits de robótica, que denominan para crear sencillos circuitos electrónicos programables que se conectan a un controlador. Esto llevaría a la producción de un pequeño robot como inicio y la creación de pequeñas modificaciones en el mismo que lo especialicen.

Su opción para 3º de E.S.O (14-15 años), pasa por los mismos objetivos pero apoyándose en Arduino y haciendo uso de sensores concluir la creación de un robot con una funcionalidad específica. En ambos casos, también ofrecen cursos de formación de profesorado.



RESUMEN DE LOS CONTENIDOS DE LA MEMORIA

El planteamiento de esta memoria sigue el mismo camino que se ha seguido durante la confección del proyecto. Primero se ha visto los motivos que han impulsado la realización de esta idea, cuál es la situación actual del escenario y lo que se pretende conseguir en el primer capítulo.

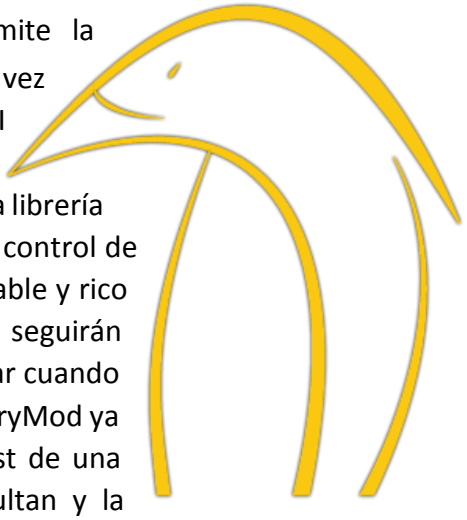
En el capítulo 2 se mostrará qué elementos han sido necesarios para llevarlo a cabo como poder hacer uso de los mismos. Pasaremos a los capítulos 3 y 4, donde se detallarán las herramientas utilizadas y se justificará qué se puede realizar con ellas, qué posibilidades ofrecen. En los capítulos 5 y 6, se expondrá una idea de visión de futuro de lo que se podría llegar a lograr con este ejercicio y el punto pedagógico que se ha tratado de aportar. Seguidamente, se detallarán algunas unas muestras de las actividades que se podrían plantear en una clase dentro del capítulo 7. Finalmente, en los capítulos 8 y 9 se concluirá esta memoria con las conclusiones obtenidas y la bibliografía que ha servido de apoyo.

2. DESCRIPCIÓN DE LAS TECNOLOGÍAS EMPLEADAS

Una de las principales ventajas del entorno que utilizamos en este proyecto consiste en su simplicidad. Hace uso de muy pocos elementos para funcionar que además son económicos y muy sencillos de obtener e instalar. Esto conlleva a un acceso muy asequible y que solo favorece que se amplíe el mercado que puede hacer uso de él.

MINECRAFT. Se trata del eje central de esta infraestructura. Es un videojuego 3-D, con vista en primera persona (aunque también permite la tercera) catalogado como 'sandbox' o de mundo abierto. Viviendo en un mundo en el que el apartado visual se critica tanto en busca de una resolución o un número de FPSs alto, el éxito cosechado por un juego como Minecraft basado en bloques es sorprendente. Según sus creadores, es un juego sobre: "colocar bloques e ir de aventuras. Explorar mundos generados aleatoriamente y construir lo que uno pueda imaginar". Uno de sus pilares es el 'crafteo', consistente en la exploración y recolección de recursos y la combinación de los mismos para obtener cada vez mejores utensilios, armas y estructuras que faciliten la superación del otro pilar del juego, la supervivencia ante los monstruos.

La simplicidad de su motor gráfico basado en los famosos cubos es lo que permite la construcción de cualquier cosa que se pase por la imaginación del jugador. El primer objetivo al iniciar el juego es conseguir un refugio en el que pasar la noche y así resguardarse y defenderse de los enemigos.



CANARYMOD. Software que permite la creación de un servidor que ejecute Minecraft y a la vez la inclusión de 'mods' en el mismo. Posee una API de código abierto y permite interactuar directamente con el servidor. Es un servidor con una librería de características de control de datos incrustada y control de permisos y grupos. Proporciona un framework estable y rico que asegura que los plugins escritos ahora seguirán funcionando en el futuro sin necesidad de actualizar cuando lo haga Minecraft. Por desgracia, el equipo de CanaryMod ya no se encuentra en activo y solo mantiene el host de una página web con un foro que los usuarios consultan y la sección de descargas (la más reciente de agosto del 2015).

SCRIPTCRAFT. Es el mod que nos permite una vez instalado en el servidor ejecutar código en lenguaje JavaScript. Uno de sus puntos fuertes es el objeto 'Drone', el cual ayuda a la hora de construir en el juego ya que es muy práctico al ser una referencia visual. Es un plugin, de código abierto, que permite a administradores, operadores y autores customizar el juego mediante JavaScript. Funciona con los siguientes servidores de Minecraft: SpigotMC, GlowStone, Bukkit y CanaryMod (el que usamos nosotros).

JAVA. Intérprete que permite la ejecución del servidor.

NOTEPAD++. Editor de textos que utilizaremos para escribir el código. Si bien, podemos ejecutar sentencias en la propia ventana de comandos del juego o en el servidor, contar con un editor de textos facilitará la labor cuando se trate de más de una línea de código. En realidad, sirve cualquier tipo de editor, por ejemplo, el famoso bloc de notas de Windows. Pero recomiendo éste debido a que reconoce muchísimos tipos de lenguaje de programación (entre ellos JavaScript) y resalta palabras clave lo cual es muy agradecido por cualquier programador. Es muy intuitivo, gratuito y fácil de instalar. Si utilizamos un equipo macOS, encontramos editores de texto como TextWrangler muy recomendables.

No necesitamos más que estos sencillos componentes para poner en marcha nuestra experiencia en Minecraft.

EXPLICACIÓN DE LAS TECNOLOGÍAS ELEGIDAS

La elección de los elementos arriba explicados se detalla a continuación según su importancia inversa. El editor de texto, podría ser cualquiera, pero se aconseja Notepad++ debido a su sencillez y opciones que ofrece. Java es necesario para poder compilar el paquete del servidor. La elección de CanaryMod como servidor se debe a que es el que se señala en el artículo de Github y porque el resto de opciones disponibles no incorporan ninguna ventaja y las indicaciones para la utilización de CanaryMod vienen detalladas en el artículo.

La elección de ScriptCraft como mod se debe a dos motivos. El primero es que como en el caso del servidor, el artículo que sirvió para inspirar este proyecto se basaba totalmente en ScriptCraft y estaba enfocado a su uso. La segunda razón y más significativa es porque permite la programación en JavaScript. Normalmente, los mods de Minecraft están escritos en Java, así que ¿por qué complicar todo añadiendo más elementos si el objetivo ya se puede alcanzar? El objetivo es transmitir los conceptos básicos de la programación y no saber programar en un lenguaje determinado. Para esto se podría haber escogido cualquiera de los ya existentes (el propio Java, JavaScript, C++, Python, Matlab, etc…). Pero la existencia de un plug-in que permite utilizar JavaScript y la consideración de que éste es mucho más sencillo de comprender que Java (por ejemplo, no hay que definir el tipo de variable al crear una) ocasionaron que fuera la elección final.

Por último, Minecraft, es el principal motivo para creer en el éxito de esta propuesta. Videojuego que desde su salida al mercado en noviembre de 2011 ha cosechado millones de ventas y captado a innumerables jugadores. La empresa que lo creó, Mojang AB vendió su estandarte a Microsoft en 2014 (3 años después de su lanzamiento) por 2.500 millones de dólares. A pesar del tiempo transcurrido, que normalmente es más que suficiente para que la comunidad de jugadores de un videojuego desaparezca, la empresa estadounidense consideró que ese desembolso sería rentable y no se equivocaba. A día de 2016, Minecraft sigue recibiendo actualizaciones, vendiendo merchandising y contando con el apoyo de modders que alargan la vida del juego.

El impacto de este videojuego es innegable, en la actualidad más de 350.000 personas lo juegan diariamente, en toda su historia cuenta con más de 100 millones de usuarios registrados. Uno de los motivos que produce estos números es su accesibilidad, ya que está disponible en casi todas las plataformas, incluso tras la compra por parte de Microsoft se lanzó al mercado para las consolas de sus máximos competidores en el mercado del entretenimiento digital, Sony y Nintendo.

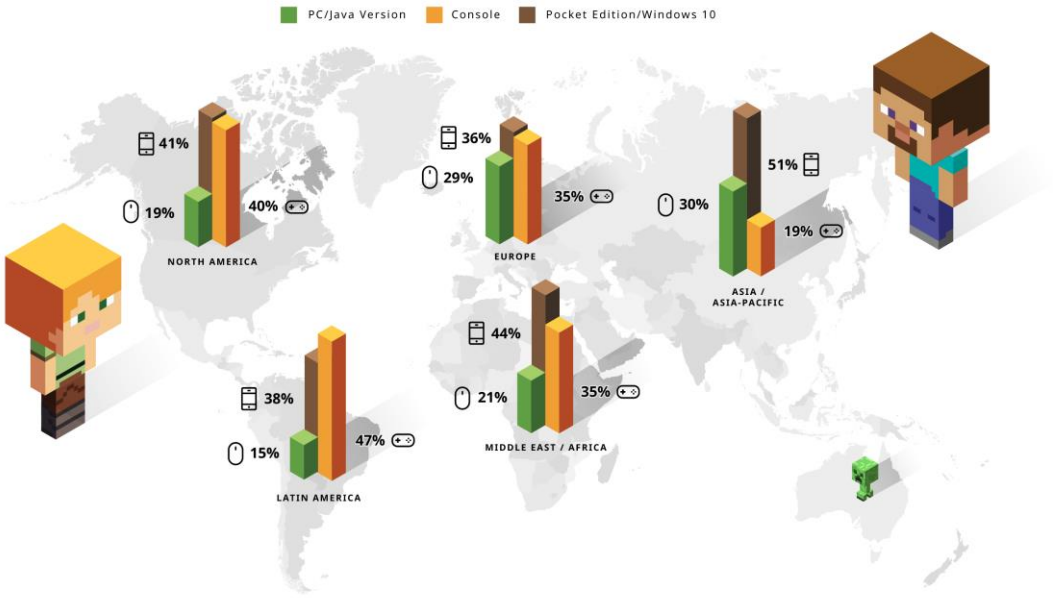
MINECRAFT

GET MINECRAFT FOR YOUR DEVICE

DESKTOPS	CONSOLES	DEVICES
Windows	Xbox One	iOS
Mac	Xbox 360	Android
Linux	PlayStation 4	Windows Phone
Windows 10	PlayStation 3	Kindle Fire
	PlayStation Vita	Gear VR

MINECRAFT

MAYBE IT'S TIME TO CHILL WITH THE MINECRAFTS.
WE'VE SOLD IT 100 000 000 TIMES ALREADY. ┐(‘∀‘)┌



MINECRAFT HAS SOLD MORE THAN 106,859,714 COPIES TO DATE

If each person that bought a copy formed a nation, it would be the 12th most populous in the world, behind Russia, Japan and Mexico.

1. China	1,382,323,332	**5.** Brazil	209,567,920	**9.** Russia	143,439,832
2. India	1,326,801,576	**6.** Pakistan	192,826,502	**10.** Mexico	128,632,004
3. U.S.	324,118,787	**7.** Nigeria	186,987,563	**11.** Japan	126,323,715
4. Indonesia	260,581,100	**8.** Bangladesh	162,910,864	**12.** Minecraft	106,859,714

SINCE THE BEGINNING OF 2016, MINECRAFT HAS AVERAGED OVER 53,000 COPIES SOLD PER DAY

The **Minecraft community** includes folks from every country and territory on the planet. There have even been 4 copies sold to crafters in **Antarctica**. We hope they enjoy the **polar bears** we're adding to the next PC / Mac update! (Yeah, we know they're from different poles.)

We're welcoming more players to the world of Minecraft now than ever before – **over 40 million people** every month spend time adventuring, exploring and building wondrous things. If everyone who played **Minecraft** on an average month were to join hands, they would be able to circle the **entire Earth** over one and a half times. But then they wouldn't have any hands free to play Minecraft. Nightmare!

Así que la razón más influyente por la que Minecraft es el núcleo de este proyecto es su repercusión. Prácticamente, cualquier niño conoce el juego, lo habrá jugado en alguna ocasión, es más, probablemente sea su juego favorito. Ante esta circunstancia, cuando se plantea un ejercicio educativo (de programación) pero que se realiza a través de una interfaz conocida, el interés del alumno es máximo y ese es uno de los propósitos de este proyecto, transmitir nociones al mismo tiempo que se está jugando. Una forma en la que aprendizaje y diversión no se encuentren en puntos opuestos.

¿QUÉ SON LOS 'MODS'?

El término viene de la abreviatura de 'modifications' y consisten en cualquier alteración del contenido de un videojuego de modo que se comporte de forma diferente a la original. Puede ir desde hacer que los personajes sean invisibles, que se disparen coches deportivos en lugar de balas o puede ser tan sutil como que los subtítulos estén en un idioma distinto a los que se establecieron para el juego primitivo.

El universo de los mods viene de lejos y siempre ha estado presente en la industria del videojuego, de hecho, en los últimos años es común que se editen juegos con las herramientas necesarias para que quién lo desee realice modificaciones. Muchos juegos se iniciaron como un mod de un juego anterior y acabaron lanzándose al mercado. Un caso famoso que puede evidenciar la relevancia que pueden llegar a conseguir es el caso de shooter táctico Arma II y su mod Day Z, éste cambiaba la ambientación y añadía un nuevo enemigo, los zombis. Era gratuito y las descargas se dispararon en pocos días, tras ganar tanto renombre se comercializó como juego independiente y a día de hoy supera en ventas a la continuación del juego que le dio origen, Arma III.

La fama de Minecraft unido a que no es complejo por su estructuración basada en bloques y a qué no hay eventos scriptados hace que la creación de mods para este juego sea una constante, la variedad es alta, desde emular la película de los juegos del hambre hasta jugar online a una mezcla de pictionary y ahorcado.

Podemos considerar a ScriptCraft como un mod creador de mods, la instalación de éste no realiza ninguna variación en el comportamiento habitual del juego, no es hasta que se ejecuta algún comando válido, gracias a la instalación de ScriptCraft, que se altera el comportamiento del juego. Veremos en un capítulo posterior algunas de las modificaciones que tendremos a nuestra disposición.

3. CONFIGURACIÓN Y PUESTA EN MARCHA DEL ENTORNO

DESCARGA E INSTALACIÓN DEL SOFTWARE

Como ya se ha dicho, la obtención de las herramientas y su instalación son muy sencillas y una de las ventajas de este proyecto.

MINECRAFT. El juego puede obtenerse de diversas formas ya que depende de la plataforma en la que lo vayamos a ejecutar, si fuera en Android desde Play Store, si fuera en Play Station 3 desde la Play Station Store o en una tienda de venta de videojuegos si buscamos el formato físico… El caso que nos ocupa es el de ordenador, la versión de PC no cuenta con formato físico así que el modo de obtenerlo es desde la página web oficial: https://minecraft.net/es/ y clickar en el apartado 'Get Game', es necesario poseer una cuenta en Mojang. Una vez logeados pasamos al apartado de pago de la copia del juego que tiene un valor de 23,95 €. Una vez se haya verificado la transacción económica (cuenta con la gran mayoría de métodos de pago) obtendremos el instalador necesario.

CANARYMOD. La página dónde se encuentran las distintas versiones es: https://canarymod.net/releases/ deberemos descargar la versión compatible con la versión de Minecraft que usemos. El nombre del archivo indica la última versión del juego con la que es compatible. Tenemos dos opciones para cada versión, el archivo comprimido (formato .zip) o el archivo puro (formato .jar). Si no se dispone de un programa de descompresión, descargar la segunda opción que llevará un poco más de tiempo, aunque no hay una gran diferencia. En ambos casos, tras el tiempo de descarga aparecerá en la carpeta de descargas elegida el archivo. Si es el formato comprimido, hacer click con el botón derecho del ratón y elegir la opción 'extraer aquí'. Ahora ya tenemos el archivo de CanaryMod.

Cada vez que arrancamos Minecraft se nos pide seleccionar que versión se quiere ejecutar, a septiembre de 2016 la última versión es la 1.10.2. Mientras que CanaryMod es compatible como máximo con la versión 1.8.0, esto se debe a que el equipo de CanaryMod abandonó el soporte y no ha creado el archivo compatible desde entonces. No es un problema, porque como se ha mencionado se puede seleccionar la versión de Minecraft a utilizar, no se obliga a hacer uso de la más reciente. De hecho, en la elaboración de este proyecto se ha usado la versión 1.7.10 sin inconveniente ninguno.

SCRIPTCRAFT. El link con la última versión (aquí si nos interesa estar completamente actualizados) es: http://scriptcraftjs.org/download/2014-11/20141114-v3.0.1/ solo hay que pinchar sobre el archivo y aceptar la descarga. Como en el caso anterior, lo encontraremos en la carpeta de descargas.

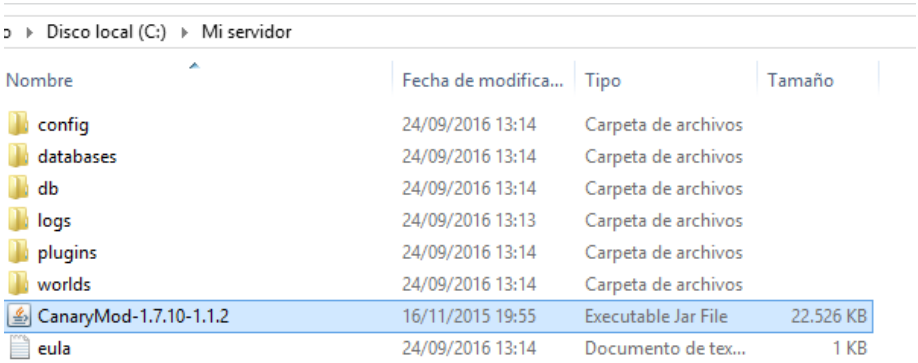
JAVA. Al tratarse de un componente utilizado en la mayoría de ordenadores es muy posible que no haya que descargarlo porque ya se encuentre instalado en el equipo. Se recomienda tener una versión superior a 1.8.0_25. Para comprobar que versión se encuentra instalada, abrir una ventana de comandos (sea cual sea el sistema operativo) y escribir lo siguiente: "java –version". Aparecerá un mensaje con la versión y en caso de que Java no esté instalado el mensaje dirá que no reconoce el comando java.

Si se da este último caso, la página de descarga es: https://www.java.com/es/download/ pinchar en el apartado 'Descarga gratuita de Java', en la nueva página que aparecerá pinchar en 'Aceptar e iniciar descarga gratuita' de este modo se nos descargará el instalador en nuestra carpeta de descargas.

NOTEPAD++. Al igual que antes, para descargar el instalador: https://notepad-plus-plus.org/download/v7.html y elegir la opción que se adecúe a nuestro equipo (32 ó 64 bits). Se puede elegir entre bajar el archivo sin o con compresión (zip o 7z). Si no queremos complicarnos pinchar en el botón verde que indica DOWNLOAD. Buscar el instalador en la carpeta de descargas.

Ahora pasamos a la instalación de cada uno de los componentes de los que hacemos uso. En el caso de Minecraft, Java y Notepad++ es suficiente con hacer doble click en el instalador y seguir las indicaciones. Sin embargo, el resto del software sigue otros pasos para poder poner en marcha un servidor.

Primeramente, tenemos que instalar CanaryMod para ello, creamos una carpeta con nombre indiferente, por ejemplo 'Mi servidor', es muy recomendable hacerlo en la raíz del disco duro. Seguidamente, movemos el archivo de CanaryMod que hemos descargado previamente y lo iniciamos con doble clic, tras unos segundos se crearán varias carpetas y un documento de texto.

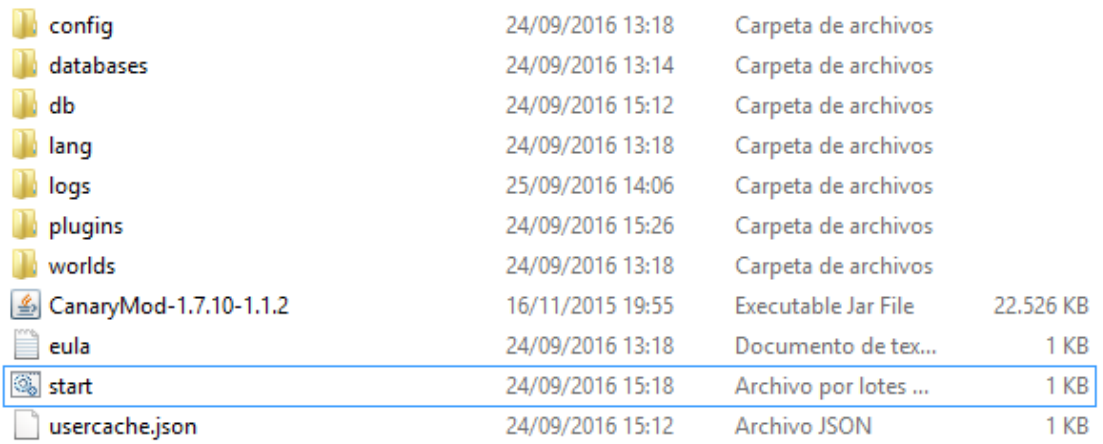


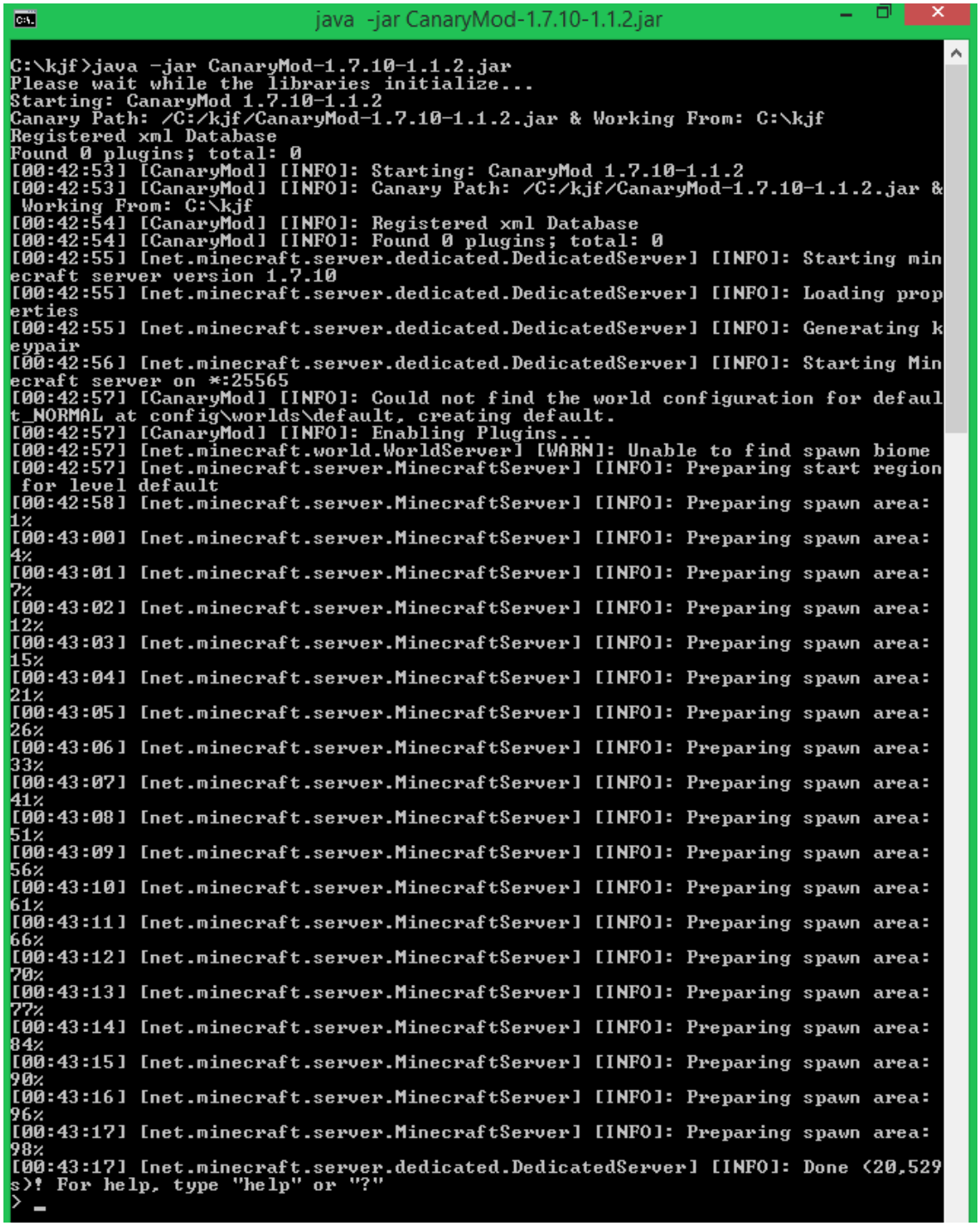
Puede que se haya abierto una ventana de comandos, si tras haberse creado los archivos arriba mostrados no se cierra automáticamente, escribir 'exit' y se cerrará. Ahora, en el archivo de texto debemos modificar la última línea y cambiarla por "eula=true", así aceptamos el acuerdo de licencia de usuario final, cuyas condiciones se encuentran en: https://account.mojang.com/documents/minecraft_eula

Posteriormente, iniciamos de nuevo el servidor. Para ello, podemos seguir haciendo uso del doble click o podemos ejecutarlo desde una ventana de comandos. Recomiendo esta última opción, ya que podemos ejecutar líneas de comandos en la misma, de la otra forma no es posible. Para lograrlo hay dos formas: Desde la propia ventana o creando un archivo que lo haga por nosotros.

Para acceder a la ventana, pulsar la tecla Windows y escribir 'cmd'. Una vez allí ubicarse en el directorio de nuestra carpeta (Mi Servidor) a través de los comandos 'cd' y 'cd..' que permiten subir y bajar a través de los directorios. Otro método es mantener pulsada la tecla shift y clicar con el botón derecho dentro de nuestra carpeta y elegir la opción 'Abrir ventana de comandos aquí', esto es más rápido pues ya nos ubica en el directorio deseado. Bien de una forma u otra, escribir "java –jar X.jar" donde X es el nombre del archivo de CanaryMod que nos descargamos de la página oficial y pulsar Enter. Para evitar complicaciones se recomienda copiar el nombre del archivo en lugar de transcribirlo directamente.

El otro método puede que sea incluso más sencillo porque una vez creado el archivo, solo tendremos que abrirlo cada vez que queramos arrancar el servidor, a diferencia del método previo el cual deberíamos completar en su totalidad. Haciendo uso de Notepad++ crear un archivo que contenga como texto: "java –jar X.jar", como en el caso anterior X debe sustituirse por el nombre del archivo CanaryMod. Podemos nombrar el archivo de cualquier forma pero por coherencia se recomienda algo como 'arranque', 'inicio' o 'start'. Lo que sí es influyente es la extensión (.bat), comprobar que a la hora de guardar la opción tipo sea 'todos los archivos'. De modo que se nos quedará algo parecido a start.bat Cada vez que abramos este archivo, se abrirá también la ventana de comandos automáticamente y ya tendremos nuestro servidor en funcionamiento.





Con esto, solo tenemos un servidor de Minecraft, nos falta añadir ScriptCraft. Es muy sencillo. Cerramos el servidor para que los cambios surtan efecto escribiendo 'shutdown'. Tras esto, solo tenemos que mover el archivo ScriptCraft que descargamos en la carpeta plugins que se encuentra dentro de nuestra carpeta principal (Mi servidor). Una vez completado esto, tenemos que arrancar de nuevo el servidor de una de las formas que ya se han indicado anteriormente. Para comprobar si todo se instaló correctamente escribir '/js 1+1' en el símbolo del sistema, debemos obtener un 2 como respuesta. Además aparecerá una carpeta llamada 'scriptcraft' dentro de 'Mi servidor'.

Con nuestro servidor en funcionamiento, arrancamos Minecraft. Nos pedirá que introduzcamos un nombre de usuario cada vez que ejecutamos el juego. Podemos cambiarlo siempre que queramos pero para que no haya que crear una lista de permisos en cada ocasión y tenga coherencia es recomendable no variar este nombre. Pinchar en el botón 'play'.

Nos aparecerá otra pantalla, en el menú Profile, dejar marcada la opción default y en el menú Version seleccionar una opción que sea igual o inferior a nuestra versión del archivo de CanaryMod. Aunque Minecraft ha seguido actualizándose hasta la versión 1.10.2, si seleccionamos ésta nuestro servidor no lo soportará. La versión más actualizada de CanaryMod es compatible hasta la actualización 1.8.0, así que es ésta la máxima versión de Minecraft que podemos ejecutar. Las novedades incorporadas a partir de esa versión no son influyentes ni provocan que la experiencia pierda calidad.

Ya estamos en el menú principal, para acceder a nuestro servidor, pulsar 'Multiplayer' y una vez en la selección de servidor en 'Añadir servidor'. En el nombre de servidor podemos poner cualquier texto, es solo un identificador para cuando tengamos varios servidores en memoria pero no designa el servidor que estemos ejecutando en el momento. En la dirección de servidor indicar 'localhost', así sí señalamos al servidor que hemos iniciado actualmente. Una vez creado, nuestro servidor aparecerá en el menú de selección de servidores y ya podemos acceder a él seleccionándolo. ¡Y ya está! Ya podemos entrar en un servidor en el que jugar a Minecraft y ejecutar nuestro código.

Hemos mostrado que nuestro servidor funciona pero todavía no podemos hacer efectivo nuestro código. Podemos escribir cualquier línea que si será comprendida pero no ejecutada por una sencilla razón, no tenemos permisos. Para otorgárnoslos, podemos escribir en el símbolo del sistema mientras se ejecuta nuestro servidor el siguiente comando: "op {nombre_de_usuario}" donde {nombre_de_usuario} es el identificador que escribimos al iniciar Minecraft. Otra forma de concedernos permisos es incluir ese nombre de usuario en el archivo 'op.txt' que aparece dentro de la carpeta 'config'. Una línea por cada usuario con permisos de operador.

CONFIGURACIÓN DE UN SERVIDOR

Las opciones que nos brinda CanaryMod para configurar nuestro servidor son amplísimas, aunque no podemos modificarlas in situ. Estos parámetros se consultan cuando se arranca el servidor por tanto para observar las modificaciones que realicemos en ellas tendremos que volver a iniciarlo. Dentro de la carpeta 'config', en el archivo 'server.txt' se encuentran todas las opciones sobre las que tenemos control.

- Announce-player-achievements – informar de la consecución de un logro a todos los jugadores.
- Ban-default-message – mensaje que se muestra a los jugadores que no tienen permitido jugar.
- Date_format – formato de la hora y fecha.
- Max-players - máximo número de jugadores online.
- Motd – subtítulo que aparece en el menú de servidores.
- Online-mode – autenticar jugadores conectados.
- Default-world-name – nombre del mundo cargado por defecto.
- View-distance – radio máximo de bloques cargados.
- Server-port – número de puerto usado.
- Rcon-enabled – permitir acceso remoto.
- Player-idle-timeout – tiempo de expulsión para un jugador inactivo.

Estas son alguna de las variables que controlan la configuración del servidor. No hay que crearlas cuando se inicia un servidor por primera vez, hay una configuración por defecto para ese caso.

También podemos establecer una configuración del mundo. En nuestro caso, vamos a elaborar labores de construcción de elementos con bastante frecuencia. Para ello, las siguientes opciones nos pueden facilitar la labor.

- world-type=FLAT – El mundo creado es plano, muy bueno para la construcción.
- spawn-villagers=false – No aparecen personajes.
- spawn-monsters=false – No aparecen enemigos.
- gamemode=1 – Modo de juego creativo.
- forceDefaultGameMode=true – Forzar modo de juego por defecto (modo creativo).

Como en el caso anterior, estos parámetros junto a todos los demás se encuentran dentro de un archivo de texto ('default_NORMAL.txt') que se encuentra en el directorio: Mi Servidor/config/worlds/default. También podemos hacer que no aparezcan animales, indicar qué animales y monstruos aparecen, aplicar una semilla que genere un mundo que haya sido creado otro usuario, habilitar el vuelo del personaje, especificar el nombre del mundo, indicar que tipos de bloque están habilitados…

Ahora hemos creado un mundo a nuestro gusto, con lo que al iniciar el juego obtendremos algo parecido a esto:

A partir de aquí, tecleando '/gamemode creative' en la ventana de texto tenemos todos los bloques de materiales y elementos del juego disponibles en el inventario para crear todo aquello que se nos ocurra. Aunque crearlos con líneas de código es bastante más rápido y eficiente.

PERMISOS

CanaryMod soporta un sistema de almacenamiento de permisos que evita que existan datos inútiles. Los permisos aparecen como mucho dos veces, una por jugador y otra por grupo. Para manejar estos permisos podemos hacer uso de la ventana de comando o hacerlo directamente desde el juego ya que los cambios se aplican directamente y no es necesario reiniciar el servidor.

Siempre que modifiquemos los parámetros de permisos debemos tener definido un grupo por defecto, que será el grupo que se adjudicará a nuevos jugadores y a jugadores que no se incluyan en la lista de usuarios.

Podemos comparar este apartado con una lista de control de acceso de un switch o router. Los grupos son conjuntos de permisos que podemos asignar a los usuarios y cada grupo puede tener un 'padre' de forma que si el grupo A hereda del grupo B, tendrá los permisos del grupo B más los propios. De esta forma, creamos estructuras de jerarquía: invitados, jugadores, moderadores y administradores, por ejemplo. Podemos crear, eliminar y renombrar grupos así como comprobar, asignar jugadores y editar los permisos de cada grupo.

La forma de asociar los permisos también es comparable al método que se utiliza en las ACLs. Los permisos se categorizan por conjuntos y podemos asignar cada permiso por separado o todos a la vez. Si solo queremos algunos permisos concretos, podemos escribirlos uno por uno o bien añadir todos y después eliminar los que no deseemos, hacerlo de una forma u otra depende del número de permisos que queremos conceder y la rapidez que conlleva un método u otro, pero funcionalmente son idénticos.

El comando para asignar un permiso es:

```
/groupmod permission add {group} {permission}
```

Donde {group} es el nombre del grupo a asignar el permiso {permission}. Para denegar el permiso cambiar 'add' por 'remove' o añadir tras {permission} ':false'. Para añadir todos los permisos que están por debajo de un nivel de jerarquía, añadir '.*':

```
/groupmod permission add Moderators canary.command.*
```

La variedad de permisos pasa desde prohibir el acceso a jugadores, expulsarlos del servidor durante la partida, permitir la modificación de grupos y sus permisos, permitir el uso de chat privado, matar a otros jugadores, uso de plugins, supersadministrador, etc… La lista completa de permisos se encuentra en: https://github.com/walterhiggins/canarymod-admin-guide#commands-1

Con estas nociones estamos preparados para controlar nuestro servidor, con nuestra configuración y limitando las funciones que cada jugador pueda ejecutar.

4. FUNCIONALIDADES DEL ENTORNO

ScriptCraft facilita muchísimas funciones que son muy útiles para construir estructuras de cualquier tamaño y que nos permite experimentar mientras jugamos. Según avancemos nuestro progreso y estudiemos las posibilidades que nos otorga este mod, seremos capaces que crear elementos más interesantes.

Para escribir nuestras líneas de código existen dos maneras: desde la ventana de chat del propio juego o en un archivo de texto externo al juego que incluiremos en el directorio: Mi Servidor/scriptcraft/plugins. Cada método tiene sus ventajas e inconvenientes. Mientras que desde la ventana de chat, a la que accedemos pulsando la tecla 'T' en cualquier momento, podemos ejecutar código sin necesidad de parar el juego. Igualmente, disponemos de una memoria con los últimos comandos escritos accesibles pulsando una simple tecla ('↑'), por el contrario, solo podemos escribir una línea de texto que además está limitada a 100 caracteres. Si hacemos uso de un archivo externo nuestro código estará más claro ya que no hay límite de caracteres y se puede comentar. Como desventajas cuenta con la necesidad de cambiar de programa entre el editor de texto y el juego cada vez que queramos alterar y ejecutar el código, y si modificamos el archivo hemos de volver al juego e introducir: '/js refresh()' para que las variaciones se carguen.

ScriptCraft proporciona muchos aspectos a controlar y funcionalidades, presentaremos algunos a continuación. Por seguir con la tradición del programador, lo primero que mostraremos será cómo realizar el clásico Hola Mundo. Para ello, solo necesitamos servirnos de la función 'echo'. /js echo('¡Hola Mundo!') producirá que nos aparezca en pantalla el mensaje ¡Hola mundo!, también podemos elegir quién verá este mensaje incluyéndolo como parámetro de la función: /js echo(usuario, ¡Hola Mundo!).

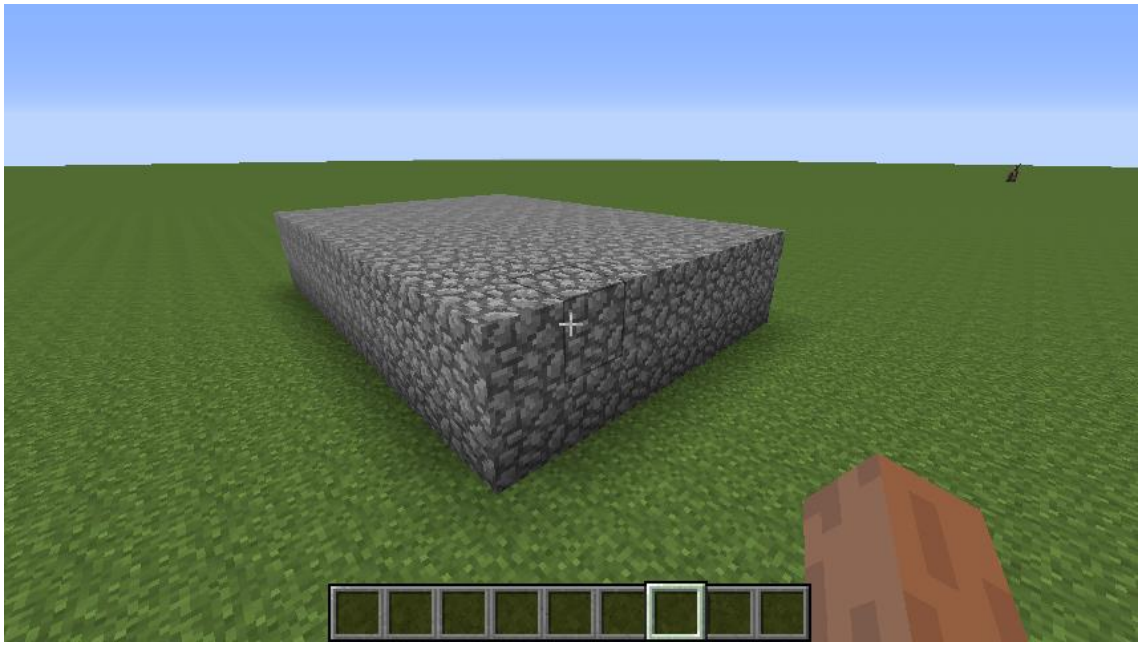
CONSTRUCCIÓN

Como hemos indicado, a la hora de construir el modo creativo es el más eficiente ya que disponemos de todos los tipos de bloques y para posicionarlos o eliminarlos solo requerimos de un click, del botón izquierdo o derecho respectivamente. Sin embargo, a pesar de esta condición cuando se trata de estructuras considerablemente grandes, ubicar bloque a bloque es una tarea muy ardua. Con ScriptCraft esto no ocurre ya que facilita varias funciones que se encargan de crear formas geométricas del tamaño y tipo de bloque que deseemos.

La función principal en este aspecto es 'box()'. Con ella, podemos crear ortoedros de tamaño y tipo de bloque cualesquiera. Por defecto, si no se indica las dimensiones, éstas tienen valor 1 por lo que ubicamos un solo cubo del material que indiquemos, este

parámetro no tiene un valor por defecto por lo que es absolutamente necesario señalarlo. Hay dos métodos para hacer esto: o bien mediante un número que identifica a cada tipo de material o bien mediante 'blocks.{nombre}' donde {nombre} es el nombre del material. Este segundo método es más laborioso porque se tarda menos en escribir un número.

Por ejemplo, un ortoedro de bloques de piedra con una altura de 7, anchura de 3 y largo de 10 bloques se crearía con el siguiente comando: '/js box(4, 7, 3, 10)' o bien con '/js box(blocks.stone, 7, 3, 10)'.



También contamos con otras funciones para crear formas simples, además con una versión de las mismas que sirven para construir la misma forma geométrica con la diferencia de que será hueca.

- box0(bloque, altura, anchura, largo). Crea un ortoedro hueco (sin bases).
- cylinder(bloque, radio, alto). Crea un cilindro.
- cylinder0(bloque, radio, alto). Crea un cilindro hueco (sin bases).
- prism(bloque, ancho largo). Crea un prisma triangular.
- prism0()bloque, ancho,largo). Crea un prisma triangular vacío.

Combinando estas funciones crear una casa en muy sencillo, las paredes son construidas rápidamente con box0 y el tejado con prism0, solo hay que adecuar dimensiones y eliminar un par de bloques en una de las paredes para colocar una puerta. Por supuesto, si queremos más detalles podríamos incluir una chimenea (cylinder0()), añadir ventanas o un jardín, pero eso lo veremos más adelante.

Otra virtud, es que podemos enlazar estas funciones, no hay necesidad de ejecutar una por una, es posible relacionarlas consecutivamente y ahorrar tiempo. Aunque si estamos desde la ventana de chat, fácilmente el número de caracteres permitidos nos limitaran la construcción y por eso cuando se trata de edificios grandes es muy recomendable ejecutar un código escrito en un archivo externo.

ScriptCraft también cuenta con funciones que construyen edificaciones predeterminadas, por ejemplo un castillo, ‘castle()’; una pirámide ‘pyramid()’, un fuerte ‘fort()’, un laberinto ‘maze()’ o una pequeña cabaña, ‘cottage()’, entre muchos otros.

MÓDULO DRONE

ScriptCraft nos proporciona un instrumento de trabajo muy útil a la hora de crear en Minecraft, el objeto 'drone', que es inmensamente útil ya que este es uno de los objetivos del juego, quese desarrolló con esta jugabilidad como base. Explicaremos este objeto como un punto de referencia remoto que podemos situar en la ubicación que nos apetezca a partir de la cual empezar a construir.

Para crear este tipo de objeto hay diversas maneras:

1. d = new Drone(self). Toma como parámetro al jugador que invoca la función y a partir de la ubicación de éste usa el bloque que se encuentre a dos cubos de distancia por delante del personaje en caso de que la cruceta que aparece en el centro de la pantalla no apunte a ningún bloque en el momento de invocar la función. Si esa misma cruceta si apunta hacia un bloque, la ubicación de éste será dónde el objeto drone se sitúe como punto de origen.
2. d = new Drone (x, y, z, dirección, mundo). Los parámetros de los que hace uso son x, coordenada del eje este-oeste; z, coordenada del eje norte-sur; y, coordenada del eje vertical; dirección, indica hacia a dónde apunta el objeto drone, puede tener 4 valores: 0 (este), 1(sur) , 2(oeste) , y 3(norte) y mundo, el mundo en el que se está jugando. Los dos últimos parámetros son opcionales y si no se indican los valores utilizados son la dirección en la que mira el jugador y mundo, mundo en el que se crea el objeto.
3. d = new Drone (localización). En este caso, se pasa la ubicación de un elemento, en la que ya se indican los parámetros que se tienen que indicar explícitamente en el caso 2.
4. Cada vez que se invoca uno de los métodos del objeto drone. Esta forma quizás sea la más efectiva y útil. Todos estos métodos devuelven 'this' por ese motivo es factible encadenar operaciones.

Podemos mover el drone según nuestro deseo con las siguientes funciones:

- up(número_de_bloques). Mueve el drone hacia arriba, tantos bloques como se indique.
- down (número_de_bloques). Mueve el drone hacia abajo tantos bloques como se indique.
- left(número_de_bloques). Mueve el drone hacia la izquierda tantos bloques como se indique.
- right(número_de_bloques). Mueve el drone hacia la derecha tantos bloques como se indique.
- fwd(número_de_bloques). Alejael drone del jugador tantos bloques como se indique.

- back(número_de_bloques). Acerca el drone al jugador tantos bloques como se indique.
- Turn(número_de_giros). Rota el drone en el sentido horario tantas veces como se indique. turn(), gira el drone 90 grados hacia la derecha.

Si no indicamos parámetro en estas funciones el valor por defecto es 1.

Otra gran utilidad de este objeto drone son los checkpoints, es decir, es posible guardar ubicaciones y regresar a ellas cuando lo deseemos. Esto es enormemente útil cuando se hacen grandes construcciones en las que el drone se traslada en numerosas ocasiones.

- Chkpt(nombre). Guarda la ubicación y la asocia con el nombre indicado.
- Move(nombre). Desplaza el drone a la ubicación que está asociada al nombre indicado.

MÓDULO INVENTARIO

Probablemente se trate de uno de los más atractivos, ya que tiene un impacto directo sobre el personaje. Permite manipular el inventario de los jugadores. Esencialmente cuenta con 3 funciones:

- add(item). Añade un ítemdel inventario.
- remove(item). Elimina un ítem del inventario.
- contains(item). Comprueba si el inventario contiene el ítem.

Utilizando este módulo conseguimos alterar el inventario de un jugador, por ejemplo:

```
var inventario = require (‘inventory’), items = require (‘items’);
// Añade 6 galletas y elimina 4 manzanas del inventario de un jugador
inventario(jugador).add(items.cookie(6)).remove(items.apple(4));
// Comprueba si el jugador tiene flechas
var tieneFlechas = inventario(jugador).contains(items.arrow(1));
```

Algunos de los ítems que se puede añadir en el juego pasan por todos los bloques de materiales, alimentos, armaduras, armamento, elementos decorativos, antorchas, yunques, libros, TNT, etc…

Mencionar que cuando se añade un ítem al inventario después se puede eliminar el número de ítems de ese tipo que tenga el jugador menos uno. Siempre queda al menos 1 ítem de cada tipo.

MÓDULO EVENT

Hasta ahora, todo lo que hemos visto eran funciones que decidíamos cuando tenían lugar con el comando '/js', es decir, el programador dictaminaba en qué momento el código era ejecutado pero si queremos que se materialice un determinado comportamiento cuando ocurra algún hecho en especial dentro del juego nos valemos de este módulo, que nos suministra una sencilla manera de escuchar eventos en Minecraft.

Una forma de definir esta herramienta es: "Haz esto cuando suceda aquello", donde esto es el evento que sucede durante el juego y aquello es una función que nosotros indicamos. Hay muchísimos (aproximadamente 160) eventos que se pueden rastrear en ScriptCraft, cuando un jugador rompe un bloque, cuando se une al servidor, cuando dispara una flecha, cuando se descarta un ítem del inventario, cuando muere un jugador, cuando respawnea, cuando un jugador mira una señal, cuando un jugador accede al inventario, cuando el clima cambia, etc… Para ver una lista con todos los eventos podemos escribir '/js events.' en la ventana de chat del juego y sin pulsar Enter, pulsar TAB si se tienen los permisos adecuados o consultar la página: https://github.com/walterhiggins/ScriptCraft/blob/master/docs/API-Reference.md.

Un ejemplo, cada vez que u jugador rompa un bloque recibirá el mensaje "Has roto un bloque":

```
function rompeBloques(evt){
        var jugador = evt.player;
        echo (jugador, "Has roto un bloque");
}
events.blockDestroy(rompeBloques);
```

En este caso cuando ejecutemos la función rompeBloques() no sucederá nada hasta el momento en el que un jugador rompa un bloque, de hecho, puede darse que si ningún bloque es roto no ocurra absolutamente nada.

Si ya no nos interesa un evento haremos uso de 'this.register()', que elimina la función de la lista de oyentes (listeners) del evento. La palabra clave 'this' se utiliza dentro de la función que maneja el evento y se refiere al objeto Listener que provee ScriptCraft. Suele ser útil cuando queremos que el manejador de eventos se ejecute en una única ocasión. En caso de querer cesar desde fuera de la función listener, crearemos una variable que sea la función oyente y después usaremos: 'variable.unregister()' cuando queramos detenerla.

El método 'on()' proporcionado por este módulo es muy importante ya que es en la que se basa el resto de métodos. Su finalidad es registrar listeners de eventos. El objeto event tiene funciones para registrar listeners para cada evento diferente. Cada vez que llamamos a un evento es como si aplicáramos este método 'on()' con nuestro

evento como parámetro. Lo mencionamos porque aunque llamar al evento directamente es más directo, también sirve para registrar eventos proporcionados por otros plugins que no sean CanaryMod. Los parámetros de este método son:

- eventType. El tipo de evento que se va a escuchar.
- Callback. La función que se invocará cuando salte el listener del evento.
- Priority. Es un parámetro opcional y que por defecto tiene valor critical. Indica la prioridad que tiene frente a listener del mismo tipo. Los posibles valores son: Critical, high, low, normal y passive.

MÓDULO CLASSROOM

Se trata de un conjunto de funciones que resultan muy prácticas para el entorno de una clase, facilitan a los profesores el conceder acceso a los estudiantes de ScriptCraft. Sigue siendo arriesgado otorgar acceso a ScritpCraft en un servidor compartido, los más avispados podrían abusar de ello y concederse a ellos mismos o a otros privilegios, pero es bastante menos arriesgado que conceder esos permisos de operador directamente para que puedan ejecutar comandos de javascript.

El propósito de este módulo no se centra tanto en asuntos de seguridad y privacidad sino en configurar un servidor compartido para acercar javascript a los alumnos. De aquí su principal propiedad, cada alumno que esté habilitado cuando se conecte al servidor tendrá un directorio dedicado donde almacenar sus archivos de texto con su propio código. Los scripts de estos directorios son cargados en una variable global con el nombre del usuario.

Imaginemos que estamos en una clase y se solicita la realización de un ejercicio que consta de crear un script que contenga un código que realice un objetivo X. Para simplificar la labor de corrección en un futuro, el profesor decide que esos scripts se llamarán ‘ejercicio1.js’ a fin de que cuando se corrijan todos los ejercicios tenerlos organizados por orden de realización. Si todos los alumnos denominan a sus scripts de la misma forma, sería un caos a la hora de invocarlos y guardarlos en el directorio, se estarían sobrescribiendo constantemente.

Este módulo permite que para ejecutar el script de un alumno solo haya que utilizar la variable antes mencionada. Por ejemplo, si un alumno tiene como usuario ANieto, únicamente hay que utilizar el comando: ‘/js ANieto.ejercicio1()’ Así cada alumno creará sus códigos sin tener colisiones por nombres. Para poder llevar a cabo esto, el directorio Mi Servidor/scripcraft/player debe ser una carpeta compartida para que se puedan guardar archivos en él desde otros equipos.

La activación de este módulo es muy sencilla y solo puede ser llevada a cabo por operadores del servidor. Con '/jsp classroom on' se activa el modo clase y se permite ejecutar código a todos los jugadores del servidor, 'jsp classroom off' desactiva el modo.

También es factible permitir (o no) el uso de ScriptCraft a jugadores que se unan al servidor o ya lo estén, por separado mediante la función 'classroom.allowScripting()' que solo los operadores pueden darle uso. Solamente necesitan como parámetro un booleano que indique si se concede permiso o no y el jugador afectado.

En cualquier momento, cuando se añada, modifique o elimine un archivo de la carpeta 'players' los contenidos de la misma se autorecargan evitando pérdidas de tiempo innecesarias.

Se han clarificado alguno de los módulos proporcionados por ScriptCraft, los más interesantes y que posibilitan una mayor experimentación pero hay otros módulos que conceden control sobre fuegos artificiales, carteles y señales con sus respectivos textos, sonidos, entre otros.

FUNCIONES EXTERNAS

Hemos mostrado muchas de las opciones que nos brinda ScriptCraft y lo más satisfactorio es ser capaces de disponer de ellas y emplearlas para combinarlas y sacar el máximo partido a Minecraft. Se ha mencionado con anterioridad que en el momento que manejamos varias líneas de código, la ventana de chat se convierte en insuficiente y tediosa. Por suerte, tenemos a nuestra disposición el uso de scripts, archivos de texto que no limitan el número de líneas de código y con las ventajas de realizar comentarios y resaltar palabras clave y reservadas, muy agradecido por los programadores.

Una vez tengamos un archivo con el código que queramos ejecutar solo hay que guardarlo en el directorio Mi Servidor/scriptcraft/plugins y tener la precaución de guardarlo como un archivo de extensión '.js'. En la ventana de chat, es tan sencillo como escribir '/js {nombre}()', donde {nombre} es el nombre de nuestro archivo sin extensión.

Muy probablemente, requeriremos de modificar nuestro código. No nos basta con guardar el archivo con las variaciones, tendremos que ejecutar el comando '/js refresh()' que detiene todos los plugins y los recarga, por tanto, ahora cuando invoquemos a nuestra función se cargará la última versión guardada.

Nota: para llamar tanto a las funciones propias como a 'refresh()' disponemos de la ventana de chat y de la consola del servidor, se recomienda la primera porque son idénticamente funcionales y de esta forma al ejecutar en el propio juego vemos en tiempo real los efectos del código y solo pasamos del editor de texto al juego y viceversa, sino involucramos también a la consola, perdiendo el tiempo pero esto es a gusto del programador.

5. PROPUESTA EDUCATIVA

Es innegable que se nos presenta un entorno muy potente, que ofrece infinitas posibilidades y que tiene un grandísimo potencial. Minecraft está diseñado para favorecer la aptitud imaginativa, es un gran puzzle que añade un componente de peligro y aleatoriedad para no pecar de tedioso. Es curioso, pero no cuenta con una historia argumentativa que progrese, su única premisa es la supervivencia. La muerte del personaje es la única manera de acabar una partida, ni siquiera hay un objetivo que sea construye tantos edificios y de unas determinadas características. No hay una lista de misiones, no se obliga a nada, ni siquiera se muestra un anuncio que diga: "Sobrevive".

El éxito de este videojuego está basado en la imaginación de los usuarios, no es atractivo por tener un guion sugestivo, unos gráficos estupendos o una banda sonora inmersiva. Este juego fue diseñado desde su inicio con este fin y eso ha favorecido casos como el de este proyecto. La comunidad de moders de Minecraft es inmensa y aumenta cada día.

Desde su salida al mercado han pasado varios años que usualmente suelen ser más de lo que dura la vida de la mayoría de videojuegos del negocio actual. Su público sigue creciendo y llegan nuevos jugadores que en general, suelen ser muy jóvenes. De hecho, muchos no han jugado antes a ningún juego (obviando, juegos simples como Fruit Ninja o Angry Birds). Por tanto, considero que la motivación de este proyecto será duradera y la elección de Minecraft como pilar de este trabajo ha sido acertada porque posiblemente pasen los años y siga siendo un fenómeno de masas.

Considero que este proyecto es atemporal, conseguirá tener éxito o no pero no pienso que dependa del momento. Sería igual de válido dentro de unos meses o de unos años, respecto a la audiencia que es uno de las razones en las que se basa este proyecto. Que existan otros conceptos similares es seguro, ya lo hay en este momento, se han explicado algunos en el primer capítulo de esta memoria. Pero opino que la coexistencia de todos es métodos es más que viable. Cada una tendrá sus ventajas e inconvenientes y siempre habrá una que se ajustará mejor a las condiciones de cada caso.

Otro aspecto a tener en cuenta, es la comunidad de consumidores de este concepto. Si se consiguiera llevar a la práctica y se observaran los beneficios que puede producir su utilización, se podría propagar y tener una retroalimentación de beneficios. Si los usuarios comparten sus casos prácticos y el estudio de su aplicación podrían analizarse para mantener una constante mejora de la implantación de esta herramienta en las aulas. En caso de que esto ocurriera sería de utilidad la creación de un foro, de forma que docentes, programadores, psicólogos, pedagogos, etc… pudieran realizar estudios, compartir conceptos y optimizar el uso de esta herramienta con la colaboración de expertos en las diversas materias que afectan a este instrumento y de

ese modo alcanzar una herramienta tan útil como cualquier otra técnica educativa y que lograra establecerse como standard en el sistema educativo.

VISIÓN DE FUTURO

Se han planteado alguna de las propiedades provechosas que proporciona este sistema, como un primer paso. Una aplicación directa se trata de la implantación en las aulas de los centros educativos, pero también es dirigible a actividades extraescolares, academias independientes, cursos de formación o incluso, en hogares particulares.

La trayectoria de este proyecto es altamente ampliable, el estudio del mismo por parte de expertos y el estudio de su implantación en un ambiente realista supondrían un perfeccionamiento de la herramienta que la elevaría a un nivel mucho mayor. Los beneficios que se podrían obtener serían inmensos, si es que no lo son ya. Supondría una intromisión en la estructura de la educación que está tratando de olvidar esa "vieja escuela" y potencia la involucración del alumno en las clases. Que el niño deje de ser un elemento pasivo que solo toma notas de lo que un docente comenta en la pizarra y se convierta en un componente activo y se empape de conocimiento al estar directamente en contacto con él.

En este trabajo se ha focalizado en educar a los más jóvenes en los principios de la programación, pero se podría extender fácilmente a avanzar un paso más e introducirse en el ámbito de la algoritmia, o directamente profundizar aún más en la programación, por ejemplo, control de Inteligencia Artificial. Pero no olvidemos que estamos ante una herramienta muy versátil y flexible, el uso que le hemos dado es debido a que permite utilizar JavaScript, que aunque ha sido escogido por resultar sencillo de aprender también es un lenguaje muy potente y nos entrega una manera de expandir los horizontes y abarcar otros temas de interés.

No es insensata la concepción de la idea de encaminar el uso de este entorno para la docencia en otros campos, véase: matemáticas, idiomas, geografía, educación vial y relaciones sociales, entre otros.

Imaginemos que el profesor plantea una operación matemática y varias posibles respuestas, asociadas cada una con un 'totém', los alumnos tras resolver el problema se dirigirían al totém que creen que indica la respuesta correcta. Si aciertan, ganan un punto, se podría incluir una tabla de resultados, para fomentar la competitividad o dificultar el proceso añadiendo un temporizador. Otra dirección, sería que los jugadores tuvieran que construir las banderas de los países (o el país de la capital) que el docente pide. Recrear un circuito de tráfico sería arduo pero no inverosímil. Siguiendo el formato tipo test, enseñar vocabulario de otros idiomas no sería complicado, o incluso ejercicios en los que los alumnos tengan que recrear un concepto que se les ha mostrado en otro lengua. Por último, el que más atrayente me parece, fomentar las relaciones sociales: Hay un premio que los alumnos deben alcanzar pero está situado a una distancia que no

es alcanzable, sin embargo si varios jugadores colaboran en la resolución de sencillos puzles si lo pueden obtener, así se alimenta la cooperación, la capacidad de resolución de problemas, la distribución espacial y la imaginación.

En este último campo puede que sea desproporcionado pero destinar este concepto para su utilización con niños con problemas de adaptación sería un servicio enormemente interesante, podrían ser niños que sufren de la barrera lingüística, o que conviven con el síndrome de Down, síndrome de Asperger o el autismo. En los casos comentados hasta ahora, la supervisión de un pedagogo era necesaria, si se llegará a destinar a estos grupos, pasaría a ser totalmente imprescindible.

Una vez se propagará el concepto que se está exponiendo en esta memoria, los casos de aplicación aumentarán y el tiempo transcurrido desencadenará de por sí multitud de diferentes ejercicios y resultados de los mismos, una mejora del sistema de uso de este entorno sería un proceso natural y podría incluso originar la estandarización de este método como fórmula educativa adjudicable a cualquier noción.

PÚBLICO OBJETIVO

El tratamiento que se le ha dado a este proyecto desde el inicio ha sido el empleo sobre niños en un rango de edad aproximado de entre 8 y 14 años, pero al igual que se podría aplicar para más ámbitos distintos a los de la programación o en niños con trastornos, se puede variar la edad de los receptores. Se puede modificar la concepción de la ideal global para adaptarla a cualquier edad. Personas mayores, jóvenes, maduros e incluso niños aún menores al rango de nuestro propósito.

Este último caso sería el más intrincado ya que partimos con un inconveniente principal y es que las capacidades lectoras y de escritura básica no están completamente desarrolladas. Se requeriría de la inclusión de otra capa entre el usuario y el software, como ya hemos visto en alguna de las herramientas actuales en el capítulo primero (quizás, mostrar opciones mediante botones con referencias visuales) que propiciara una explicación más intuitiva. Los conceptos que se ilustrarían serían aún más básicos, por ejemplo, la programación secuencial. El personaje puede que supere un obstáculo si: camina, salta y camina, pero chocaría si salta, camina, camina.

HISTORIA ARGUMENTATIVA

Aunque en un capítulo posterior se presentarán algunos ejercicios que se podrían trabajar en un aula, existe una fórmula mejor para exprimir el entorno utilizado, que está formado por personajes y elementos que facultan la creación de variados

universos. Crear un relato, en el que el personaje debe superar determinas pruebas para lograr un objetivo e ir enlazando todos los ejercicios a través de una aventura que mantendría la atención de los alumnos por completo. Una historia que sumergiría al niño en una mecánica en la que la diversión es un eje principal y la explicación de conceptos se convertiría en el medio que el alumno tiene para superar esos retos y su asimilación sería más natural al verlos aplicados y más importante aún, al haberlos aplicado él en primera persona. Figuremos que nuestro personaje Steve, aparece en un mundo en el que ese momento está diluviando, esto es controlable por el operador del servidor (profesor) y para que Steve no se resfríe ni se sienta incómodo, lo primero que debe conseguir es construirse un refugio, y se podría centrar en el uso del objeto drone y las funciones de construcción expuestas en el capítulo número cuatro.

La confección de este relato es una labor muy complicada pues se debe incrementar paralelamente la dificultad de los ejercicios/retos expuestos con la captación del interés del alumno. De nada sirve un cuento que tenga mucho sentido y que esté repleto de pruebas a superar si es aburrido. Una vez más, el apoyo proporcionado por un experto pedagogo sería inexorable.

CLASES ON-LINE

Existe la posibilidad de adaptar todo lo que se ha comentado hasta ahora a una clase a distancia. En cuanto a infraestructura siempre hemos trabajado con un servidor, al tratarse de un espacio físico cercano lo más cómodo sería utilizar una red en la que todos los alumnos estén conectados pero no hay necesidad de que el jugador se encuentre en la misma sala que el operador del servidor u otros jugadores.

La principal diferencia sería a la hora de buscar el servidor, en lugar de introducir 'localhost' o la IP de nuestra propia red, sería una dirección IP distinta, pero la concepción del propio juego es la de una experiencia multijugador en línea. Este concepto sería de los más sencillos de implementar.

COLABORACIÓN DE UN PEDAGOGO

A lo largo de esta memoria se ha indicado en numerosas ocasiones el hecho de que la colaboración de un pedagogo es esencial para conseguir obtener algún beneficio y sobre todo, no perjudicar a los alumnos. La cooperación con un programador provocará que este proyecto sea eficiente a la hora de introducirlo en una clase. Toda aportación por parte de un experto es bienvenida y debido a lo delicado del público de este caso es aún más significativo.

6. TÉCNICAS PEDAGÓGICAS

A pesar de carecer de cualquier conocimiento y experiencia en el campo pedagógico he trabajado como profesor de clases particulares con niños (y no tan niños) de las edades que nos conciernen, por supuesto, me he hallado en el lado de los alumnos y además en mi entorno familiar estoy rodeado de profesores y educadores. Esto ha podido originar que posea ciertos conceptos de este ámbito pero en ningún caso me da potestad alguna para considerar que tenga la certeza de que las técnicas que expondré a continuación vayan a ser efectivas. Solo la experiencia me ha otorgado esas posibles y endebles competencias, no he sido instruido y no estoy cualificado para desarrollar ningún aspecto, mi formación académica no ha estado orientada a la pedagogía.

En el capítulo siguiente, presentaré algunos posibles ejercicios que podrían plantearse a un alumno, acabar esta memoria sin exponer estas cuestiones prácticas sería desilusionante. Sin embargo, están faltos de cualquier característica que los autorizase a ser considerados como una manera adecuada de formar a un niño en los conceptos que nos atañen.

Recordemos uno de los fines de este trabajo, conseguir despertar en los alumnos el interés por descubrir y conocer la programación. De esta forma, se encontrarán más receptivos y el progreso de una clase será bastante fluido. Un mecanismo para alcanzar este planteamiento es el entretenimiento y la diversión. Confeccionar situaciones que produzcan un impacto en la mente de los pequeños, algo que recuerden y vinculen con las distintas nociones, de ahí el empleo de un videojuego que además les es conocido. Convertir las lecciones en un juego, no menospreciando la importancia de las mismas pero si canalizarlas hacia el concepto de asimilar conocimientos a través de un contacto directo, de una participación activa del alumno que manipula y se sumerge por completo en la experiencia y alejarlas del aprendizaje por mera observación y repetición.

HACIENDO TRAMPAS

En una partida normal de Minecraft, en la que no hay participación de ningún mod, los primeros minutos se concentran en encontrar comida, para que el personaje no fallezca de inanición y fabricar un refugio para esconderse de los monstruos que aparecen por la noche. Para avanzar en estas dos tareas se hace uso de la recolección y el crafteo. El jugador debe explorar el entorno que le rodea en busca de alimento y elementos que permitan construir un refugio. Con estos elementos en posesión, los podemos combinar para ‘craftear’ nuevos elementos que nos ayuden a recolectar mucho más rápido.

Empezamos sin ningún ítem y uno de los componentes más básicos es la madera, que se obtiene de los árboles (hay distintas especies), al inicio para talar estos árboles nos servimos de nuestros puños (por raro que suene, el juego es así) lo cual es bastante lento. Cuando obtenemos los suficientes bloques podemos combinarlos para elaborar una mesa de crafteo, que nos permite combinar muchos más elementos y obtener mejores herramientas. Con la madera recolectada podemos crear instrumentos como hachas, palas, picos o espadas con las que reducimos el tiempo que precisamos para talar un bloque de madera. Con un pico, acopiar piedra es sencillo (se podría hacer con las manos pero tarda muchísimo) y con este nuevo material podemos craftear un horno, que nos permite cocinar la comida, fabricar nuevas armas y utensilios o crear nuevos materiales como el carbón, el cual podemos combinar con madera para el crafteo de una antorcha, que otorgan puntos de visión en a noche. Las herramientas se deterioran con su empleo, es posible que nos hayamos alejado mucho de nuestro hogar y se nos rompa la espada y no podamos defendernos o cazar hasta que no volvamos a la mesa/horno y fabriquemos otra, según el material (no es lo mismo la madera que el diamante), resistirán durante más o menos tiempo.

En el mundo en el que aparecemos también conviven animales, vacas, cerdos y ovejas son los más comúnes, podemos matarlos para obtener carne o lana, que también se pueden combinar. La carne nos repone el medidor de hambre, que si llega a cero provoca que la vida del personaje empiece a reducirse, pero no nos restaura en la misma medida comer carne cruda que cocinada (haciendo uso del horno y madera). Con la lana y con madera podemos fabricar una cama.

Con estos componentes: comida, utensilios, luz y madera sobrevivir a la primera noche es sencillo, solo nos queda construir un pequeño refugio que nos separe de los enemigos durante la noche. Por ahora, situar bloques madera y una puerta (fabricada con madera, por supuesto) alrededor de la mesa de crafteo, el horno y la cama es lo máximo a lo que podemos aspirar. Un hogar no muy práctico pero que cumple su función defensora. Conforme avancemos en la aventura podremos mejorar nuestro equipo, crear un pequeño huerto, expandir nuestra vivienda, etc…

Todo este proceso, que es lo primero que procede cada vez que una partida nueva es comenzada, es muy familiar para el jugador habitual de Minecraft. Se trata de un procedimiento muy monótono y que puede llegar a ser aburrido porque consta casi en su totalidad de clickar el botón del ratón para talar o picar piedra. La duración de este ronda los 15 minutos, dependiendo del terreno que se ha generado aleatoriamente puede que se requiera de más tiempo porque los elementos y animales se encuentran alejados del punto de aparición del personaje.

Figuremos a un alumno delante del ordenador al que le pedimos que construya un refugio y después de transcurrir ese cuarto de hora aproximado nos sentamos en su silla y tras teclear ‘/js up().cottage()’ en la ventana de chat y pulsar enter (pulsar 21 teclas, unos 10 segundos como mucho) aparece esto en pantalla:

Una cabaña de piedra con tejado, puerta, ventanas, un cartel que reza “Home sweet home”, un horno y una cama. Opino que el alumno se quedará asombrado y expresará algo parecido a: “¡Hala!, ¿Cómo has hecho eso?”. Considero que este método en el que se solicita el cumplimiento de una labor (si puede ser aburrida y ardua, mejor) a un alumno y tras completarla, demostrarle que con unos sencillos comandos se puede realizar lo mismo (incluso, mejorarlo) es muy efectivo para lograr despertar el interés del alumno y que se mantenga atento a toda la explicación que prosiga.

En cuanto al aspecto de la comida y del crafteo de utensilios y armas también se puede resolver con unas líneas de código, utilizando el módulo ‘inventory’ que ya se ha expuesto en el cuarto capítulo. Llenar el inventario del alumno de zanahorias, manzanas, carne cocinada, tartas, armadura, espadas, hachas (del mejor material, diamante) o cualquier otro componente que deseemos sin tener que recolectar, picar piedra o talar árboles en ningún momento también puede provocar un gran impacto en el alumno.

CURVA DE DIFICULTAD

Es singular que este concepto sea característico tanto de los programas académicos como de los videojuegos y es infalible que debe nombrarse en nuestro caso. Se debe establecer un progreso en la dificultad de los ejercicios que se plantean en el aula, manteniendo un equilibrio para que el alumno no se enfrente a demasiados retos sencillos que le provoquen aburrimiento ni dar un gran salto en la dificultad de modo que al alumno se le planteen de improvisto pruebas muy complejas para las que no está acostumbrado.

Si lo que se les plantea es muy sencillo, los alumnos se acabarán aburriendo pero si es demasiado desafiante se acabarán frustrando, en cualquier caso perderemos el interés del alumno por seguir descubriendo nuevos conceptos, por tanto, diseñar una serie de ejercicios caracterizados por una progresión gradual en el nivel de desafío es vital.

Este avance de dificultad puede verse reflejado en el efecto que tengan los ejercicios en el mundo virtual. De forma que según escalemos esa progresión se nos pidan cosas que son más difíciles de conseguir en el juego. Focalizar esa graduación no solo en incrementar el número de líneas de código a completar o el número de variables a manejar al mismo tiempo sino que lo que reflejen esos comandos en el juego sea cada vez más sorprendente.

Se quiere hacer una diferenciación entre dificultad y complejidad. Puede que construir un castillo se tan fácil como invocar una función, que deriva en escribir un solo comando y que abrir una puerta dependa de varias variables y un condicional y que conste de varias líneas de código. Conseguir que el impacto visual de la resolución de un ejercicio y su dificultad es complicado y debe estudiarse con detenimiento.

MODO SUPERVIVENCIA VS MODO CREATIVO

Contamos con dos modos principales de juego. Modo supervivencia, el standard y en el que hay que sobrevivir mediante recolección de materiales y el modo creativo en el que desaparecen los stats de vitalidad y hambre y los enemigos no atacan, es decir, el personaje no tiene peligro que acaben con su vida, además están disponibles y de forma infinita todos los bloques de materiales del juego en el inventario junto con otros elementos como hornos, puertas, utensilios, armas, comida, en definitiva, todo está desbloqueado.

Plantear las actividades de clase en un modo u otro es algo que se debe analizar para obtener el máximo provecho. Volviendo al caso de construir un refugio, en el modo creación es mucho más sencillo pues la recolección de la madera no se encuentra. Aun así, lleva mucho más tiempo crear una casa en este modo que usando la función ‘cottage()’. Podría interponerse este caso entre la construcción de manera tradicional y con comandos para reforzar el concepto de que programando se puede agilizar los procesos.

El hecho de que no haya amenazas en el modo creación puede relajar el curso de una lección y que los monstruos no se conviertan en un incordio para el desarrollo de una clase puede ser beneficioso, de todas formas este aspecto se puede configurar en las opciones del servidor (archivo ‘default_NORMAL.txt’).

El modo creativo no es tan conocido por los alumnos y es posible que se pierda ese ingrediente de familiaridad con la herramienta que favorece el acercamiento entre la misma y el niño, sin embargo, abre las puertas por completo al enriquecimiento de la imaginación, aún más si lo compaginamos con un mundo plano, lo que equivaldría a poner frente al niño un lienzo en blanco y una paleta con toda la gama de colores.

COOPERACIÓN VS COMPETITIVIDAD

Este es un debate que ha permanecido constante en la educación. No hay duda de que fomentar la colaboración es esencial, el trabajo en equipo es una aptitud social indispensable en la experiencia vital de cualquier persona. Por otra parte, estamos en un momento de impulso de la igualdad/equidad, en la que se pretende tumbar cualquier diferenciación por rasgo. Por estos motivos, el ganador se aparenta claro, sin embargo la competitividad siempre ha estado presente en las aulas. Queramos o no, aunque sea de manera indirecta, las calificaciones de un examen crean ese clasismo entre vencedores y vencidos. En cualquier competición deportiva, siempre hay un ganador, un primer puesto, que se lleva la gloria aunque a los demás se les concedan galardones por participación.

Escuchar en un aula la frase: “Te he ganado, soy mejor que tú” no se manifiesta como algo a buscar, pero la competitividad puede acrecentar un valor de superación, que también es importante en la vida de cualquier individuo, para ser capaz de afrontar retos y no desistir ante adversidades. Otros argumentan que la única consecuencia es la aparición de la agresividad.

La controversia no tiene fin y en este proyecto no se pretende darle solución, solo manifestar que ambas opciones tienen beneficios y que la utilización de una u otra depende en gran medida de la opinión del docente.

Con la masificación que se está declarando en las aulas, el número de alumnos normalmente superará al de equipos, así que si se llevará a cabo este proyecto en un aula real se deberían realizar los ejercicios en parejas, con lo que cubrimos una de las competencias, enfrentar a las parejas entre sí induciría la otra, conseguir un equilibrio entre ambas sería lo más recomendable.

Al trabajar en equipos, la colaboración surge como algo natural al tratar de resolver problemas, cada miembro aporta sus ideas y se consensa para hallar la mejor solución. Para la competición, simplemente con añadir un marcador que estuviera presente en la interfaz gráfica constantemente se establecería una marca a batir en todo momento, tener más puntuación que los demás.

TÉCNICA ESFUERZO - RECOMPENSA

Puede que tras el desvanecimiento del primer impacto visual que produce crear construcciones complejas en cuestión de segundos la curiosidad del alumno se diluya. Para poder reavivar ese interés se presenta esta idea. Prometer una recompensa, cualquier concepto que el alumno pueda desear en favor de completar los ejercicios.

Con esto también se pretende eliminar la posible incertidumbre de una causa que justifique llevar a cabo los ejercicios, "¿qué gano haciendo esto?". Para evitar el aburrimiento y la desmotivación se presenta esta técnica. Premiar que el alumno realice las tareas. Existirán detractores de este concepto que aleguen como razón que un niño no debe ser estimulado con una recompensa por realizar una labor que se supone que es su única obligación, estudiar.

Vamos a comentar un posible planteamiento de esta técnica. Supongamos que en los minutos finales de una clase se detienen los ejercicios y se autoriza a los alumnos a disfrutar de un minijuego incluido en ScriptCraft, que se basa en eliminar al resto de jugadores lanzando bolas de nieve. Hasta aquí se podría utilizar esto como recompensa, aquellos alumnos que se han esforzado durante la clase pueden relajarse con este entretenimiento.

Es posible avanzar un grado más, convirtiendo ese minijuego como parte de la clase. Establecer que siempre se acabará la clase con una partida a ese juego, pero a lo largo de la clase y tras ir completando con éxito cada uno de los ejercicios el alumno se ha ganado una pieza de armadura. Se trata de una característica íntrinseca a la jugabilidad de Minecraft, existen 4 partes de armadura (casco, peto, grebas y botas) pueden ser de distintos materiales, cada uno mejor que el anterior. Aquél que haya completado todos los ejercicios poseerá una armadura completa de diamante (la mejor de todo el juego) mientras que otro puede que tenga un casco y unas botas de hierro, un peto de cuero y no tenga mallas.

Otorgar una pieza al inventario de un jugador es factible como ya se ha explicado en otros apartados y esa pieza se trata de un elemento casi tangible (es virtual, por supuesto) pero es visualizable y utilizable, además tiene una función, no es un mero galardón que solo puede exhibirse. En el minijuego, es más sencillo que quién se haya esforzado más durante la lección logré ganar porque podrá resistir más impactos de bolas de nieve. No es definitivo, un jugador con peor equipamiento puede ser el vencedor así que no se condena el no conseguir terminar los ejercicios.

Aquí se puede introducir de nuevo la competitividad que comentábamos anteriormente, al igual que el modo creativo, la activación de ese modo podría actuar como la propia recompensa, y se transformaría en un equivalente a ese dibujo/deporte libre que nos concedían de pequeños. Aquel alumno que acaba antes de tiempo puede usar ese modo de juego.

7. ACTIVIDADES PROPUESTAS

En este capítulo se expondrán una serie de ejercicios con los que se intenta mostrar los conceptos básicos de la programación. Una prueba de lo que podría llegar a implantarse en un aula. El planteamiento a la hora de idear estas actividades ha sido el de tratar los fundamentos de la programación de forma individual y gradual, respetando esa curva de dificultad mencionada previamente.

Así, se han imaginado retos que sirven para una visualización de las utilidades de variables, condicionales, bucles, arrays y funciones. Se considera que la noción de 'objeto' puede ser demasiado compleja como para que un niño la comprenda, una aproximación podría ser hacer uso del objeto drone pues al poder identificarlo en la pantalla de juego como el bloque al que apunta el jugador puede ser más fácil de representar en la mente del alumno. Por otra parte, podría ser contraproducente pues la sensación transmitida puede ser que el 'objeto' es el bloque y esto es completamente erróneo.

VARIABLES

Uno de los motivos por los que se escogió JavaScript como lenguaje utilizado para estos primeros pasos en el mundo de la programación. Se trata de uno de los conceptos más complicados de explicar debido a que es una idea muy abstracta y la primera que se enseña. Por tanto, no hay nada que respalde su explicación. Es una serie de caracteres (si es una palabra representativa, mejor) que se refieren a un conjunto de caracteres, un número o un booleano. El caso del booleano no es conocido por los niños, pero si el concepto de verdadero y falso, no debería ser muy complicada la compresión de esta idea.

Sin embargo, imaginemos que vamos a calcular el área de un rectángulo y creamos las variables ancho y alto de valores 4 y 10. Para alumnos que se están iniciando en las matemáticas, explicarles que al ejecutar 'ancho * alto', obtendrán 40 puede ser confuso. Lo primero que destaca es que se están multiplicando dos palabras y además el resultado que se imprime por pantalla no es otra palabra (área) sino un número.

Debido a esta complejidad, se ha optado por utilizar directamente la ventana de chat del Minecraft ya que incluir el editor de textos y estar pasando de un programa a otro favorecería el desconcierto.

Al no tener que indicar el tipo de variable (char, string, integer, double, boolean, etc…) declarar una variable es más sencillo que, por ejemplo, en Java. Se ha enfocado

esta explicación a mostrar al alumno que con el escribiendo lo mismo ocurren fenómenos diferentes si la variable tiene valores distintos.

Supongamos que queremos que se nos muestre por pantalla el nombre del alumno. Si ejecutamos '/js echo(ANieto, 'Alberto')' al alumno le aparece el mensaje en pantalla pero siempre aparecerá el mismo texto. Pero si creamos la variable 'var nombre = Alberto' y ejecutamos '/js echo (ANieto, nombre)' no solo no se mostrará en pantalla 'nombre' sino que además pondrá 'Alberto', cuando ni siquiera hemos escrito eso en los comandos. Si modificamos el valor de la variable, ejecutando el mismo comando obtenemos resultados diferentes y así demostrar al niño la utilidad de las variables.

Para convertirlo en algo más simple crearemos una función, que además de saludar y escribirse en castellano, no necesite de la variable que señala a que jugador le aparecerá el mensaje. Aunque no se haya escrito implícitamente el nombre de Alberto, sí que aparece una referencia al alumno pues se usa su nombre de usuario y esto puede colisionar con la idea de sorprender al alumno produciendo que el juego le salude por su nombre sin habérselo indicado en el comando.

```
exports.saludar = function(name){
     echo(self, '¡Hola ' + name +'!' );
}
```

De este modo, el alumno debe escribir '/js saludar(nombre)' y se mostrará '¡Hola Alberto!'.

Con esto hemos conseguido un añadido que se trata de empezar a acostumbrar al niño a la sintaxis de un lenguaje de programación. En caso de usar la ventana de chat del juego se debe añadir siempre '/js' antes de cualquier línea de código o el hecho de que hemos provocado que aparezca el valor de la variable en pantalla pero no han aparecido las comillas simples que se indican al crear la variable nombre de tipo string.

Siguiendo con el procedimiento de mostrar distintos resultados para entradas idénticas, pasamos ahora a las variables numéricas. Queremos crear un vecindario, para ello necesitamos saber cuántas casas hay que construir. Así que creamos una variable casas que nos lo indique. '/js var casas = 8'. Invocaremos la función 'vecindario()' con esta variable como parámetro de entrada.

Ahora variamos la variable casas, le asignamos el valor 5 y ejecutamos el mismo comando que antes '/js vecindario(casas)' y observamos que aunque si aparecen casas y un pasillo de flores, el número de casas es distinto a pesar de utilizar el mismo texto.

```
exports.vecindario = function(casas){

var d = new Drone(self),casasNew;
casas%2==1 ? casasNew = Math.floor(casas/2): casasNew
= casas/2;
d.up(1);
d.chkpt('start');
d.turn(3);

      for (var i = 0; i < casas/2; i++){
      d.cottage()
      d.right (9);
      }
      d.back(10);
      d.left(3);
```

```
        d.turn(2);
        for (var i = 0; i < casasNew; i++){
        d.cottage()
        d.right (9);
        }
        d.move('start');
        d.right(1);
        d.box(37,9,1, Math.ceil(casas/2)*9 - 2);
}
```

Las viviendas que se construyen constan de un cartel, haremos uso de él para reforzar el concepto de la variable string. Apoyándonos en una función 'cartel()' que requiere de un parámetro de entrada conseguiremos que en la señal se pueda leer 'Casa de Alberto'.

Debemos hacer que el personaje esté mirando directamente a la señal que queremos modificar y entonces ejecutar el comando. De modo que ahora los efectos que provocamos se llevan a cabo dentro del propio universo y no en una ventana que por definición muestra mensajes.

```
exports.cartel = function(name) {
var d = new Drone(self);
d.wallsign(['Casa de', name, '']);
}
```

Con este ejercicio el alumno dispondrá de varias casas y es natural que aspire a cambiar los carteles del resto de casas, con los nombres de sus amigos, padres, hermanos, mascotas, etc… Así se incentiva la reiteración de esta acción probando con distintos nombres. Asimismo, el personaje debe moverse físicamente entre los diferentes carteles, lo que supone una mayor inmersión.

En el caso de las variables booleanas vamos a formular una pregunta cerrada, que solo tenga como posibles respuestas sí o no. Empezaremos cuestionándonos si en nuestro mundo está lloviendo (condición manipulable por el operador del servidor). Invocamos a la función 'lluvia()' y obtendremos una de las dos posibles respuestas en función de la condición climatológica.

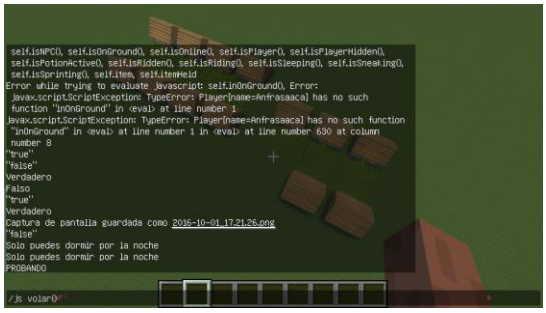
```
exports.lluvia = function() {
var i = self.world.isRaining();
echo(self, (i ? 'Verdadero' : 'Falso'));
}
```

Se ha decidido devolver la respuesta en castellano para no desviar la atención del asunto principal, comprender el principio básico del álgebra de Boole. Quizá esto se complete cuando se lleve a cabo el siguiente ejercicio en el que la pregunta cerrada es ¿Está NO lloviendo?

```
exports.noLluvia = function() {
var i = self.world.isRaining();
echo(self, (i ? 'Falso' : 'Verdadero'));
}
```

Estos ejercicios no son tan impactantes para el alumno porque no hay una gran reacción a los comandos, para hacer más inmersiva la experiencia podemos plantear una nueva pregunta: ¿Está volando el personaje? En el modo creativo, el jugador puede activar un modo vuelo pulsando dos veces seguidas la barra espaciadora. Por lo que se puede dejar al alumno experimentar y que decida en qué momento su personaje vuela y que se percate de que la respuesta cambia ante hechos que ha decidido él mismo.

```
exports.volar = function() {
var i = self.isOnGround();
echo(self, (i ? 'Falso' : 'Verdadero'));
}
```



Recordamos que en las propuestas de este apartado el alumno todavía sigue usando la ventana de chat, en el siguiente apartado ya podremos presentar código de varias líneas en las que la ventana se queda insuficiente y es necesario utilizar un editor de texto para visualizar el código de forma clara.

CONDICIONALES

En este apartado también se ha pensado en atraer la atención del alumno, manipulando las estadísticas de vida y hambre del personaje. En una partida normal, estos stas están presentes en todo momento en pantalla y son vigilados por los usuarios en todo momento pues son indicadores de cómo se está desarrollando la partida. Buscar que estos valores se encuentren en sus niveles máximos es una de las prioridades para los jugadores, por eso ser capaces de modificarlos a nuestro antojo puede ser muy llamativo.

Sirviéndonos de una acción conocida por los alumnos, como es mostrar mensajes por pantalla, vamos a evaluar estos datos, restaurarlos al máximo (si no son elevados) e informar a los jugadores sobre estas cantidades.

```
exports.salud = function() {
 salud = self.health, hambre = self.hunger;
     if(salud < 10) {
          echo(self, '¡Estás cerca de morir, cúrate!');
          self.setHealth(20);
           }
     else{
          echo(self, 'Todavía tienes salud suficiente
para sobrevivir');
          }
}
```

```
exports.comida = function() {
 hambre = self.hunger;

     if(hambre < 14) {
          echo( self, '¡Tienes mucha hambre, come!');
          self.setHunger(20);
      }
     else{
     echo( self, 'Todavía no estás hambriento.');
      }
}
```

Puede resultar un poco caótico para el alumno ver tanto texto que no comprende por eso existe unas versiones de estos códigos en los que las sentencias condicionadas se sustituyen por una función de nombre castellano y definitoria de su funcionalidad. Es decisión del profesor hacer uso de una u otra en función de las características de los

alumnos. Si bien al declarar las funciones se siguen visualizando ese (para el niño) galimatías, que incluso se incrementa al ntroducir más líneas de código, se puede distanciar visualmente del área de acción (condicionales) que es donde se centra la atención. Al desvincular ese texto con este alejamiento, no interfiere en tanta medida y evita confusiones.

```
exports.saludC = function() {
 salud = self.health;
     if(salud < 14) {
           curar();
           }
     else{
            noCurar()
            }
}

/*** Crear un gap, que evite distracciones y
desconciertos***/


var curar = function (){
     echo( self, '¡Estás cerca de morir, cúrate!');
     self.setHealth(20);
}

var noCurar = function (){
     echo( self, 'Todavía tienes vida suficiente para
sobrevivir.');
}
```

Además de estas versiones con la misma estructura se puede avanzar un poco más apoyándose en ellas e introducir la noción del else if. Simplemente, no situaremos un límite de salud o hambre sino dos.

```
exports.saludE = function() {
 salud = self.health;
     if(salud < 7) {
          echo(self, '¡Estás cerca de morir, cúrate!');
          self.setHealth(20);       }
     else if(salud < 14){
          echo(self, 'Revisa tu salud.');
          }
     else{
          echo(self, 'Todavía tienes suficiente vida
para sobrevivir.');
          }
}
```

Como hemos indicado antes, el código con esta nueva incorporación, también tiene las versiones antes mostradas: para comprobar la salud o el hambre y mediante llamadas a funciones en las sentencias condicionadas o no. También es viable, combinar todo. Un código en el que se testeen ambas estadísticas, se establezcan varios límites y en una condición invocar a una función y en otra no invocar. Además también se podría construir una versión en la que se haga uso de la sentencia switch. Al reiterar y combinar ideas, evitaremos ilustrar estos códigos.

BUCLES

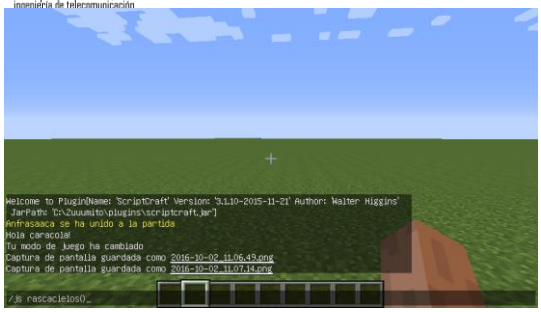
Probablemente, este concepto sea más sencillo de explicar que los anteriores, resumiéndolo a la mínima expresión: "en un bucle se hace lo mismo tantas veces como quieras". Es fácil asociarlo a una máquina, a un ordenador.

Se propone un ejercicio de gran impacto visual y que difícilmente el alumno vaya a olvidar: construir un rascacielos. En este caso, solicitar al alumno que previo código y con las condiciones habituales del videojuego trate de crear el edificio. Tardaría varias horas y además se aburriría porque está repitiendo la misma acción continuamente. Si se le demuestra que construir una planta de la edificación es muy sencillo y posteriormente que mediante un bucle podemos crear esa misma planta cuántas veces nos apetezca y obtener el rascacielos en cuestión de segundos se fascinará y será más receptivo a comprender que se necesita crear una variable de conteo y de qué forma inicializarla, evaluarla y variarla.

```
exports.rascacielos = function ( pisos ) {
  if ( typeof pisos == 'undefined' ) {
    pisos = 10;
  }
  var d = new Drone(self);
  d.up();

  for ( var i = 0; i < pisos; i++ ) {
   d
   .box(blocks.iron,20,1,20)
   .up()
   .box0(blocks.glass_pane,20,3,20)
   .up(3);
  }
}
```

Existen variaciones de este código en las que incluir los checkpoints del objeto drone pero alejan la atención del bucle for y debemos evitarlo. También podríamos incluir como parámetros la altura de cada planta o el material con el que están construidas.



Pero un bucle for no tiene por qué producir el mismo resultado en cada iteración. Aunque en el ejercicio de rascacielos esto si se da (se crea una planta pero a distintas alturas) la apreciación del alumno puede ser que lo que hace el bucle es construir cada piso, que siempre hace lo mismo construir pisos iguales.

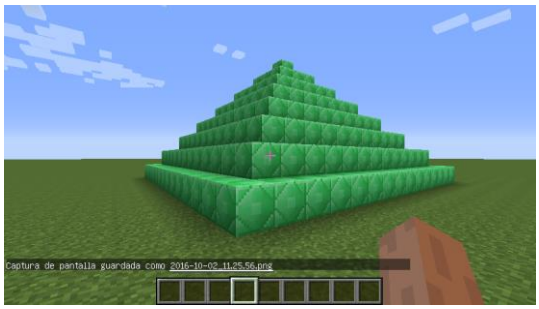
Para reforzar el concepto de reiteración pero no de obtener el mismo resultado el siguiente ejercicio consiste en construir una pirámide, como antes en cada iteración se crea un nivel de la pirámide pero estos niveles son distintos (difieren en tamaño) visualmente a diferencia de los pisos del rascacielos.

```
exports.piramide = function (base){
var d = new Drone(self);
d.up();

for( var i = base; i > 0; i = i-2){
     d.box(133, i, 1, i);
     d.up().fwd().right();
}

}
```

Como antes, podemos complicar la función añadiendo más parámetros, como el material o pedir que sea una pirámide hueca. Además aquí se da a conocer un nuevo tipo de incremento (en este caso, decremento) que no se realiza por pasos de unidad (++ / --) así que introducimos nueva sintaxis.



ARRAYS

Para este ejercicio, el diseño basado en bloques de Minecraft es de gran ayuda. Podemos visualizar un array unidimensional como un conjunto de variables estructurado. El fin de esta actividad es ordenar un array que contiene los colores del arcoíris en el orden incorrecto. Cuando finalice el ejercicio podremos observar dentro del propio juego la entrada y salida de la función (arrays ordenado y desordenado).

Crearemos una variable de tipo Array con los siete colores del arcoíris (rojo, naranja, amarillo, verde, azul claro, azul oscuro o índigo y violeta). Después analizaremos el array para encontrar algún fallo al introducir los colores, al tratarse de strings es fácil que se escape algún carácter o se dupliquen o haya alguna equivocación respecto las mayúsculas. Tras esto, asociaremos cada color con un número del 0 al 6, para introducir que las posiciones de los arrays empiezan en el cero, por convenio. Posteriormente, ordenaremos este array de números de menor a mayor y finalmente transformaremos el array de números en uno con los colores correspondientes para acabar "imprimiéndolo" en pantalla, valiéndonos de bloques de colores.

Mientras realizamos este ejercicio, se repasan las nociones de apartados anteriores, la estructura switch, variables y bucles for. Se considera un ejercicio muy completo y de gran utilidad. Ya que representa visualmente un array, dentro del propio juego, aumentando la inmersión en la programación.

Este código es susceptible a variar de complejidad, se podrían evitar nombrar directamente los colores y asociarlos directamente a números, pero introducir una codificación entre el código y el alumno conllevaría a confusión. Aunque se trate de mucho texto ya empieza a ser familiar para los alumnos.

```
exports.arcoiris = function(desorden){

// 'Imprimimos el arcoirir desordenado'

var d = new Drone (self);
for(i = 0; i < desorden.length; i++){
     switch(desorden[i]){
      case 0:
          d.box(blocks.wool.red,1,1,12).right();
          break;
      case 1:
          d.box(blocks.wool.orange,1,1,12).right();
          break;
      case 2:
          d.box(blocks.wool.yellow,1,1,12).right();
          break;
      case 3:
          d.box(blocks.wool.lime,1,1,12).right();
          break;
      case 4:
          d.box(blocks.wool.lightblue,1,1,12).right();
          break;
      case 5:
          d.box(blocks.wool.blue,1,1,12).right();
          break;
      case 6:
          d.box(blocks.wool.purple,1,1,12).right();
          break;
      default:
          d.box(11,1,1,12).right();

     }
}

// Codificación color-número
for(i = 0; i < desorden.length; i++){
     switch(desorden[i]){
      case "Rojo":
          desorden[i] = 0;
          break;
      case "Naranja":
          desorden [i] = 1;
          break;
      case "Amarillo":
          desorden [i] = 2;
          break;
      case "Verde":
          desorden [i] = 3;
          break;
      case "Turquesa":
          desorden [i] = 4;
```

```
            break;
        case "Azul":
            desorden [i] = 5;
            break;
        case "Violeta":
            desorden [i] = 6;
            break;
        default:
            desorden[i] = 84;

      }
}

//Ordenamos array
 var ordenados = desorden.sort();

//Reubicamos el drone
 d.fwd (13).left (7);

// 'Imprimimos el arcoiris ordenado
for(j = 0; j < ordenados.length; j++){
     switch(ordenados[j]){
      case 0:
          d.box(blocks.wool.red,1,1,12).right();
          break;
      case 1:
          d.box(blocks.wool.orange,1,1,12).right();
          break;
      case 2:
          d.box(blocks.wool.yellow,1,1,12).right();
          break;
      case 3:
          d.box(blocks.wool.lime,1,1,12).right();
          break;
      case 4:
          d.box(blocks.wool.lightblue,1,1,12).right();
          break;
      case 5:
          d.box(blocks.wool.blue,1,1,12).right();
          break;
      case 6:
          d.box(blocks.wool.purple,1,1,12).right();
          break;
      default:
          d.box(11,1,1,12).right();
     }
}

}
```

Esta actividad también serviría para introducir la algoritmia, se complicaría bastante el ejercicio pero valdría con suplantar la función 'sort()' por un código puro que efectúe el algoritmo de la burbuja. Es decir, sustituir la línea en la que se utiliza por:

```
//Creamos un array idéntico para no machacar el original
 var ordenado = desorden;

//recorreremos todos los elementos hasta n-1
 for(i=0; i < (ordenado.length - 1); i++){

    //recorreremos todos los elementos hasta n-i, tomar en cuenta los ultimos no tiene caso ya que ya están acomodados.
    for(j=0; j < (ordenado.length - i); j++){

        //comparamos
        if(ordenado[j] > ordenado[j+1]){
            //guardamos el numero mayor en el auxiliar
            aux=ordenado[j];

            //guardamos el numero menor en el lugar correspondiente
            ordenado[j]=ordenado[j+1];

            //asignamos el auxiliar en el lugar correspondiente
            ordenado[j+1]=aux;

  }   }   }
```

FUNCIONES

A lo largo de todos los ejercicios mostrados se han estado invocando funciones constantemente. Es más, para comprobar si el código creado es correcto se llama a una función que lo ejecuta enteramente. Por tanto, el alumno ha estado haciendo uso de funciones aunque desconociera de su existencia. Apoyarse en todos estos ejercicios previos, ya familiares para los alumnos para explicar qué es y cómo trabaja una función debería ser sencillo.

Por eso, repetir los ejercicios anteriores sustituyendo código por funciones parece una forma de enseñar este concepto en una situación conocida por los alumnos. Por ejemplo, en la actividad que trata los arrays. La función 'sort()' que ha sido sustituida por el algoritmo de la burbuja o el propio algoritmo podría ser invocado por una función 'ordenar()'.

OBJETOS

Al inicio, se descartó introducirse tanto en los conceptos de la programación y esta noción no iba a ser tratada pero la disponibilidad del objeto drone, podría facilitar esta labor. Se trata de una referencia visual, es un punto que se sitúa en el bloque al que mira el personaje, que podemos trasladar a nuestro antojo y ubicar bloques de un material a nuestra elección para crear nuestras propias referencias espaciales.

Por eso, solicitar a los alumnos la creación de pequeños edificios como una casa, construida a su gusto, a diferencia de la función 'cottage()' antes mencionada que siempre construye una cabaña con la misma distribución, sería un buen ejercicio práctico.

Haciendo uso de las funciones del objeto drone, que ya se expusieron en el capítulo cuarto se pueden realizar construcciones simples (como una casa sencilla) con pocas líneas de código.

```
/js
up().box0(4,9,5,12).right(4).door().left(4).up(5).prism(5,9,12)
```

PLANTEAMIENTO

Todos los ejercicios expuestos en este capítulo toleran diversas modificaciones, en función de cómo esté orientada la clase, el profesor será quien decida cómo plantear estas actividades. El alumno puede empezar con el código casi completo a falta de alguna línea que deba completar, que tenga que escribir todo el código (al tratarse de nociones tan simples, la mayoría son pocas líneas). Se puede practicar la sintaxis previamente en la ventana de chat para posteriormente pasar al editor de texto. Otra opción, es plantear la problemática a la clase y los alumnos aportando sus ideas vayan descubriendo por sí solos que necesidades necesitan cubrir y después demostrar que herramientas pueden ser útiles.

La concepción de cómo están dirigidos estos ejercicios queda completamente a cargo de un experto pedagogo. Por tanto, no nos embarcamos en esa tarea, nos limitamos a mostrar pruebas que pueden servir como excusa para empezar a escribir de la programación.

8. CONCLUSIÓN

El mundo tecnológico avanza a pasos agigantados y en un futuro muy próximo la cualificación de la mano de obra en este ámbito será mandatoria para cubrir el número de puestos de trabajo que se requerirán. Comenzar esta formación cuanto antes favorecerá que el aprendizaje de estos conocimientos sea más fluido.

Enseñar a los más pequeños a programar puede ser muy complicado debido a la complejidad y abstracción de los conceptos que rodean a la programación. Conseguir que esas ideas puedan visualizarse en un mundo 'real' ayuda enormemente a su comprensión. Por eso, hacer uso de un videojuego como Minecraft es muy acertado pues los alumnos lo asocian con diversión y lograr un impacto en los niños es sencillo con lo que se despierta su interés y sus ganas de aprender cómo piensa una máquina.

Como resumen de este trabajo, se ha instalado y configurado un entorno basado en el famoso videojuego, Minecraft. Se han analizado, evaluado y presentado muchas de las funcionalidades proporcionadas por el sistema.

Por otra parte, se ha ofrecido una propuesta educativa, constituida por un grupo de procedimientos pedagógicos que exprimen el potencial educativo de este entorno de programación. Además, para concluir se presentan varias actividades que favorecen la adquisición de competencias en el campo de la programación.

9. BIBLIOGRAFÍA

- https://www.theguardian.com/technology/2014/aug/07/ofcom-children-digital-technology-better-than-adults
- http://www.independent.co.uk/life-style/gadgets-and-tech/features/are-children-naturally-better-with-computers-than-their-parents-8628034.html
- http://blogthinkbig.com/el-colegio-del-futuro-pasa-por-la-programacion-temprana/
- http://tecnologia.elpais.com/tecnologia/2014/11/19/actualidad/1416417300_241820.html
- https://issuu.com/gruposiena/docs/n__104_padres/13?e=8701546/32932857
- https://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming
- http://www.eldiario.es/turing/Ninos-programadores-ensenanza-programacion-escuelas_0_293970921.html
- http://www.elmundo.es/madrid/2015/09/02/55e62d89e2704efc378b45aa.html
- http://ccaa.elpais.com/ccaa/2014/09/03/madrid/1409772225_352560.html
- http://gestiondgmejora.educa.madrid.org/
- http://blogthinkbig.com/aprender-programar-futuro-educacion/
- http://www.juntadeandalucia.es/educacion/webportal/web/cecd/prensa/notas-de-prensa/-/noticia/detalle/educacion-destina-cerca-de-2-5-millones-de-euros-a-modernizar-cuatro-centros-docentes-publicos-de-1
- http://gestiondgmejora.educa.madrid.org/bqopenscad/
- http://gestiondgmejora.educa.madrid.org/bqarduino/
- http://complubot.com/inicio/recursos/recursos-educacion/tecnologia-programacion-y-robotica/
- http://complubot.com/inicio/quienes-somos/
- http://walterhiggins.net/blog/
- https://github.com/walterhiggins
- https://twitter.com/walter?lang=es
- http://gestiondgmejora.educa.madrid.org/tecnoeducacion/
- http://www.desarrolloweb.com/articulos/2477.php
- http://www.xataka.com/makers/como-empezar-a-aprender-programacion-consejos-y-recursos-para-hacerlo-de-adulto
- http://computerhoy.com/noticias/software/juguete-fisher-price-ensena-ninos-programar-38861?utm_content=bufferc6379&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer
- http://www.firstlegoleague.es/que-es-first-lego-league/

- http://www.applesfera.com/aplicaciones-ios-1/swift-playgrounds-una-nueva-app-para-aprender-swift-directo-desde-el-ipad
- https://developer.apple.com/videos/play/wwdc2016/408/
- http://thenextweb.com/apple/2016/07/14/apple-swift-playgrounds-preview/#gref
- http://www.elmundo.es/tecnologia/2016/09/22/57e3fde0468aeb2b0b8b4682.html
- http://www.cnet.com/special-reports/minecraft/mindcraft-helping-students-learn
- http://www.educaciontrespuntocero.com/novedades2/software2/minecraft-education-edition-el-juego-de-moda-ahora-mas-cerca-del-mundo-educativo-que-nunca/32046.html
- http://www.educaciontrespuntocero.com/noticias/que-es-minecraft-educacion/32274.html
- http://www.stuffaboutcode.com/2014/10/minecraft-raspberryjuice-and-canarymod.html
- https://canarymod.net/index.php
- https://github.com/CanaryModTeam/CanaryMod
- http://es.ign.com/minecraft-pc/104056/news/minecraft-sobrepasa-los-100-millones-en-ventas
- https://minecraft-statistic.net/en/players_list/
- http://www.gamespot.com/articles/minecraft-passes-100-million-registered-users-14-3-million-sales-on-pc/1100-6417972/
- http://neurogadget.net/2016/03/08/minecraft-convention-location-and-date-has-been-revealed/25669
- https://i.ytimg.com/vi/pHNbYqILJfw/hqdefault.jpg
- http://www.portalprogramas.com/hunger-games/
- https://basicoyfacil.wordpress.com/2012/01/04/que-es-eula/
- https://github.com/walterhiggins/canarymod-admin-guide#permissions-and-groups-1
- https://www.youtube.com/watch?v=BZTgJXT-3Rc&context=C346e6baADOEgsToPDskLInCYbp7b9CrmmO79KsVUb
- https://www.youtube.com/watch?v=AIAvAfK7RbM

Nota: a lo largo de toda la memoria se ha utilizado expresiones como el alumno, el profesor, los niños, el pedagogo, etc… Siempre en masculino, por supuesto no se insinúa que el sexo femenino no esté contemplado en esta idea. Es una simple cuestión lingüística en la que se usa el masculino por comodidad.