



# Práctica guiada 6. Tanque de agua


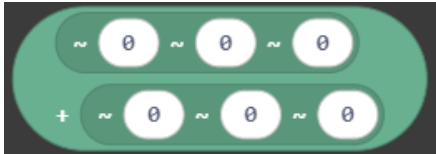


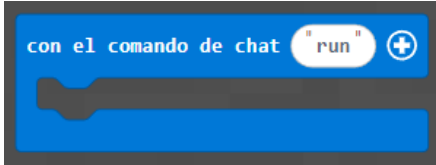
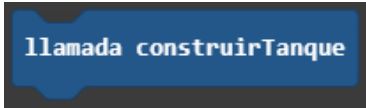
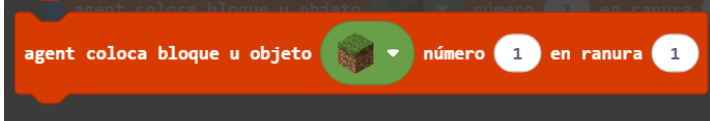
En esta práctica aprenderemos a diseñar y programar un minijuego de tipo “Tanque de agua” dentro de Minecraft Education. A lo largo de las siguientes actividades, construiremos una estructura interactiva en la que el jugador deberá activar un mecanismo con precisión para hacer caer agua sobre un objetivo. Para ello, programaremos eventos personalizados, gestionaremos condiciones de acierto, y utilizaremos bloques de detección, funciones y temporización. Además, trabajaremos aspectos como la creación de efectos visuales, el control de flujo del programa y la retroalimentación para el jugador, con el objetivo de desarrollar una experiencia de juego divertida y completa.

## Mecánicas de Minecraft



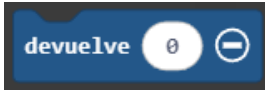


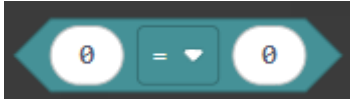
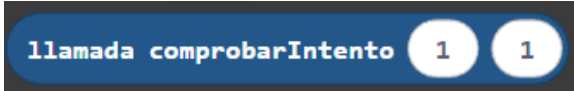


Mecánicas	Para ordenador
Abrir el editor de código	C
Abrir chat	T
Abrir el menú de pausa; cerrar los menús de juego	ESC
Minar un bloque	Clic Izquierdo
Interactuar con botones y con personajes	Clic derecho
Saltar	Barra espaciadora
Movimiento del jugador (Adelante/Atrás/Izquierda/Derecha)	W/S/A/D
Dirección del jugador	Movimiento del ratón


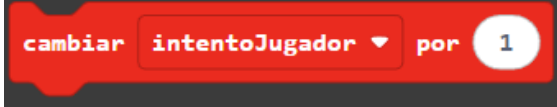

## Bloques para esta práctica

<b>Función</b> 	Bloque de código que realiza una tarea en específico.
<b>Agente desplazarse</b> 	Mueve al agente en la dirección especificada el número de veces indicado.

<p>Círculo</p> 	<p>Rellena un círculo de bloques desde una posición central.</p>
<p>Suma de coordenadas</p> 	<p>Suma dos posiciones.</p>
<p>Posición del agente</p> 	<p>Devuelve el valor de posición del agente. Es decir, sus coordenadas en el mundo.</p>
<p>Repetir</p> 	<p>Ejecuta la parte del programa contenida el número de veces que se le indique.</p>
<p>Con el comando de chat</p> 	<p>Evento que se activa al escribir una palabra en concreto en el chat.</p>
<p>Llamada a una función</p> 	<p>De este modo se puede ejecutar el código de una función.</p>
<p>Agente coloca bloque u objeto</p> 	<p>Coloca el número especificado de bloques de minecraft/objetos en la ranura especificada del inventario del agente.</p>

<p>Establecer ranura activa</p> 	<p>Establece que ranura del inventario va a utilizar el agente.</p>
<p>Agente colocar</p> 	<p>Coloca un objeto en el mundo desde la ranura seleccionada anteriormente del inventario del agente.</p>
<p>Huevo de ...</p> 	<p>Hace aparecer un huevo de mob (criatura) de Minecraft en las coordenadas indicadas.</p>
<p>Fijar</p> 	<p>Inicializa una variable, es decir, le da un valor inicial.</p>
<p>Escoge al azar</p> 	<p>Selecciona al azar un valor entre el intervalo introducido.</p>
<p>Falso</p> 	<p>Devuelve que un valor es falso.</p>
<p>Mostrar</p> 	<p>Muestra un texto en pantalla al jugador seleccionado, con la opción de poner un título y un subtítulo.</p>
<p>Parámetros de una función</p> 	<p>Los parámetros en una función son valores que se pasan a la función desde fuera y que la función puede usar y procesar</p>

<p>Si / si no</p> 	<p>Permite ejecutar acciones diferentes según si se cumple o no una condición.</p>
<p>Comparador mayor / menor</p> 	<p>Devuelve verdadero si la condición de mayor que o menor que se cumple. En caso contrario devuelve falso.</p>
<p>Devuelve</p> 	<p>Permite a una función devolver información.</p>
<p>Valor de texto</p> 	<p>Devuelve un texto.</p>
<p>No</p> 	<p>Invierte una condición lógica.</p>
<p>Comparador igual</p> 	<p>Devuelve verdadero si los dos valores de la condición son iguales.</p>
<p>Llamada a función con parámetros de salida y entrada</p> 	<p>Llama a una función que requiere unos parámetros de entrada y obtiene su valor de salida.</p>
<p>Verdadero</p> 	<p>Devuelve que un valor es verdadero.</p>
<p>Decir</p> 	<p>Pone el mensaje especificado en el chat de juego.</p>

<p>Agente destruir</p> 	<p>Manda al agente la orden de destruir el bloque que se encuentre en la dirección indicada.</p>
<p>Cambiar constante por</p> 	<p>Cambiar el valor de una constante por lo indicado. En caso de ser constante numérica suma o resta según el valor puesto.</p>
<p>Unir texto</p> 	<p>Une diferentes valores y los convierte en un solo valor de texto.</p>

## Parte 1. Construyendo el escenario

En este juego, el agente escogerá al azar un número secreto, y el jugador deberá adivinar que número es en 6 intentos o menos, si no lo acierta, un pobre aldeano caerá en un tanque de agua. Para que te hagas una idea, el escenario final será el siguiente, y por cada intento, el agente quitará un bloque de debajo del aldeano.



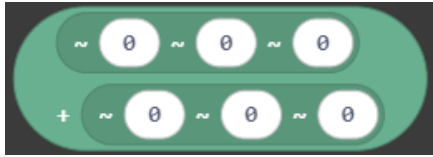
Para poder hacer este juego, lo primero que deberemos hacer es crear la zona de juego. Comenzamos por el mismo paso de siempre, generando un nuevo proyecto. Y con el proyecto ya creado, generaremos una nueva función a la que llamaremos “construirTanque”.



Podríamos pedirle a nuestro agente que construyese bloque por bloque la estructura, pero eso nos llevaría mucho código y tiempo, y hay un código en AVANZADO que podríamos utilizar para esta ocasión, lo podemos encontrar en el apartado FORMAS, como nuestro tanque va a ser un cilindro, podríamos gastar para crearlo el bloque “círculo”. Pero antes, le pediremos que el agente se mueva un bloque hacia arriba, para que no choque con la nueva construcción.



Deberemos cambiar algunos parámetros de nuestro círculo, el primero es que queremos que sea de cristal, por lo que deberemos cambiar el tipo de bloque de construcción, como punto central, queremos que sea justo debajo de nuestro agente, así que deberemos usar el bloque de sumar posiciones, y sumarle a la posición del agente un “-1” en el eje y, es decir, el segundo valor de las coordenadas, de esta forma la posición será justo debajo del agente.



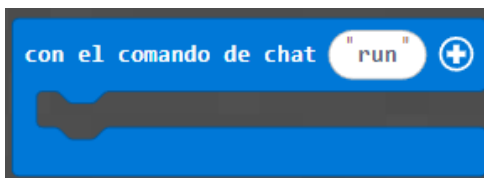
posición de agent

También deberemos cambiar el radio, que lo haremos de 3, esta es la distancia en bloques que tomará el círculo desde el centro para construir. Como queremos que el círculo sea plano en el suelo, deberemos cambiar la orientación a Y, y el último parámetro lo podemos dejar como está. Después de que se construya, deberemos rellenarlo con agua, y para hacerlo debemos crear un nuevo círculo dentro de ese círculo, pero usando como material “agua”. Para ello solo tenemos que duplicar el bloque del círculo, cambiar el material por “agua” y reducir el radio de 3 a 2.

A continuación, como queremos que sea un tanque profundo, vamos a añadir un bucle que repita todo lo que hemos hecho, pero 3 veces, de este modo tendremos un tanque de tres bloques de alto.



Para poder probar si se construye nuestro tanque, deberemos llamar a nuestra función, como hemos hecho en anteriores prácticas. Lo vamos a llamar desde el bloque “con el comando de chat”, creando un nuevo comando de chat que será, como otras veces, “jugar”.

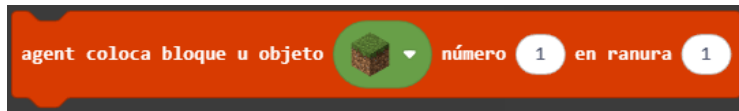


llamada construirTanque

Al probarlo, deberíamos conseguir un tanque de tres bloques de alto, una vez testado el código podemos devolver al agente a su posición inicial reseteando el mapa, como hemos hecho otras veces, desde la máquina recreativa.

Con el tanque ya preparado, vamos a acabar nuestra estructura añadiendo los bloques necesarios y a nuestro aldeano.

Justo después de crear el tanque vamos a darle al agente 6 de “bloque de hierro”, que conformarán la torre sobre la que estará el aldeano.



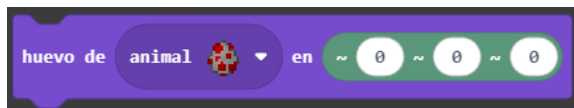
Recordad asegurarnos de que el agente tenga activa la ranura del inventario correcta.



Para construir la torre el agente deberá moverse un bloque arriba y colocar el bloque justo debajo suyo, y deberá hacerlo 6 veces, por lo que podemos utilizar un bucle. El bloque de desplazarse y el de bucle ya los hemos usado en esta práctica, pero recordemos cual es el de colocar.



Una vez termine la torre, es decir, después del bucle, vamos a mover una última vez hacia arriba al agente, para poder colocar al aldeano justo arriba de la torre, para ello utilizaremos el bloque del apartado MOBS llamado “huevo de ... en ...”, y seleccionaremos el huevo de aldeano, dando como coordenadas la misma que la del agente.



Finalmente, para terminar nuestra zona de juego, queremos colocar a nuestro agente en la posición adecuada para comenzar a jugar. Para conseguir esto lo haremos moverse hacia atrás en un bloque y bajar 7 bloques.

Una vez tengamos esto, podemos probar el código para ver si todo queda en su posición. Deberías conseguir algo como lo siguiente, con el agente mirando a los bloques de hierro, preparado para romperlos.





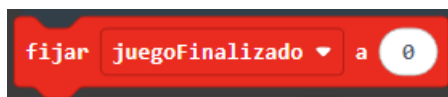
## Parte 2. Iniciando las variables

Ahora que ya tenemos completa la variable de construir el tanque, podemos recogerla para que ocupe menos espacio y no nos moleste.

Durante el juego hay diferentes variables que van a ir cambiando, como el número de oportunidades que le quedan al jugador para adivinar el número, el número aleatorio que ha escogido el agente, y si el juego ha terminado. Para cada uno de estos valores iniciaremos una variable:

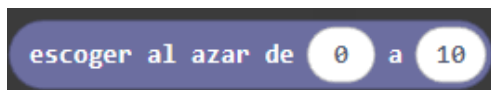
- “oportunidadesRestantes”: la iniciaremos con el número total de oportunidades que tendrá el jugador para adivinar el número e iremos restando cada vez que este haga un intento.
- “numeroAgente”: guardará el número aleatorio que ha escogido el agente y nos servirá para compararlo con los números del jugador.
- “juegoFinalizado”: guardará si el juego ha terminado o no.

Una vez creadas vamos a inicializarlas justo debajo de la llamada a la función anterior, en el bloque “con el comando de chat”.



“oportunidadesRestantes” la iniciaremos con el número total de intentos que tendrá el jugador para adivinar el número, es decir, 6.

“numeroAgente” la iniciaremos con un número aleatorio entre 1 y 50, podéis escoger el rango que queráis, pero que sea asequible para tratar de adivinar el número.

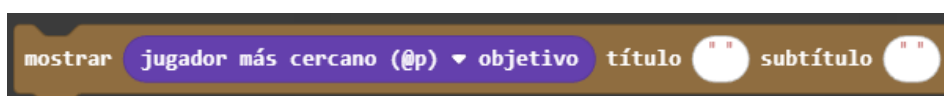


“juegoFinalizado” la iniciaremos como una condición de falso, que pondremos a verdadero cuando el juego termine.



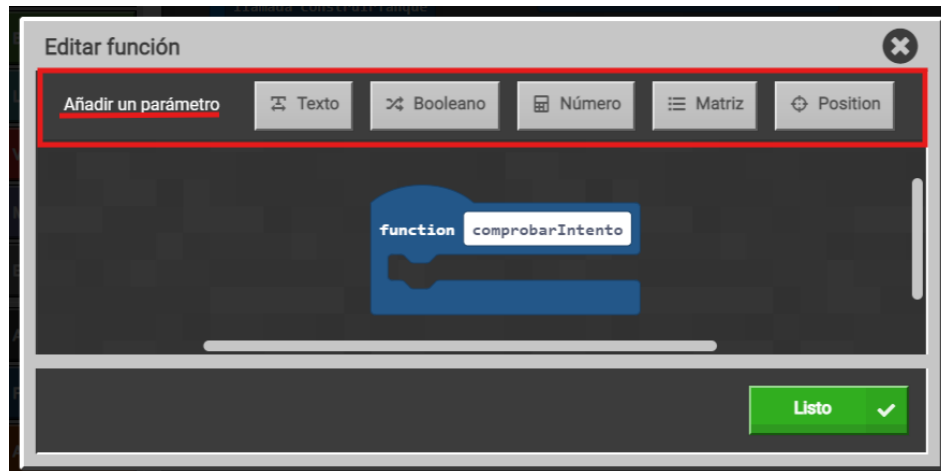
## Parte 3. Calculando los resultados

Con las variables ya creadas y antes de pasar definitivamente a programar el funcionamiento del juego, vamos a colocar un bloque “mostrar” justo debajo de la inicialización de las variables, para decirle al jugador que comience a jugar, puedes poner el título y el subtítulo que quieras, pero recuerda decirle al jugador lo que debe hacer, es decir, ¿adivinar un número entre qué intervalo?

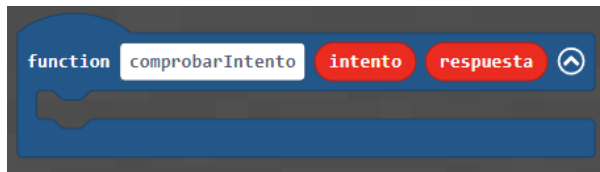


Una vez ya hemos mostrado el cartel de comenzar a jugar al jugador, pasamos con la programación de la jugabilidad.

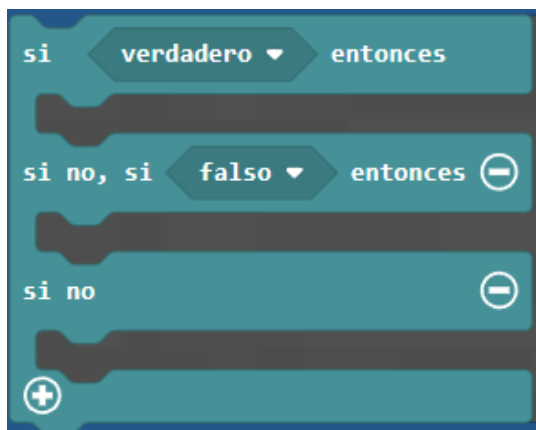
Vamos a utilizar de nuevo una función, a la que llamaremos “comprobarIntento”, pero esta va a ser algo diferente a las que ya hemos hecho, puesto que vamos a tener que darle información externa, es decir, le vamos a pasar parámetros, como hemos visto en teoría. Esta información la añadimos desde el siguiente apartado dentro de funciones:



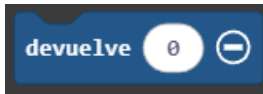
Nosotros vamos a añadir dos parámetros nuevos de tipo número, al primero lo llamaremos “intento” y al segundo lo llamaremos “respuesta”. El primero contendrá el valor que el jugador trate de adivinar y la respuesta contendrá el número aleatorio del agente, pero eso lo programaremos más adelante, de momento vamos simplemente a añadirlos.



Estos parámetros son valores que llegan a la función desde fuera, y ahora nosotros queremos que la función devuelva unos resultados después de haber procesado esos valores, en este caso devolverá “si el número de intento es mayor que el de respuesta”, “si el intento es menor a la respuesta” o “si el intento es igual a la respuesta”. Para ello utilizaremos un bloque de condición si / si no, si/ si no y el comparador de “mayor y menor que”.



En caso de que el intento sea mayor que la respuesta, queremos devolverle al programa la información de que el número intentado es demasiado alto, para poder devolver información a partir de una función necesitaremos usar el bloque devuelve.



Como queremos devolver un texto en el que ponga “MUY ALTO”, deberemos ir al apartado TEXTO y coger el primer bloque de valor de texto y ponerlo en el devuelve.



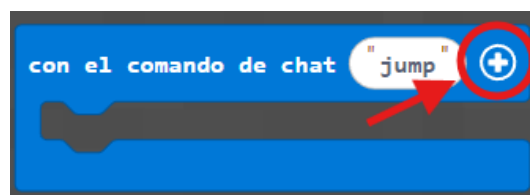
Ahora, para la siguiente condición vamos a duplicar el comparador anterior y cambiarlo de mayor que, a menor que. Y esta vez el texto que devolvamos será un “MUY BAJO”.

La última opción que nos queda es que el jugador acierte el número, en cuyo caso, habrá ganado, así que vamos a añadirle otro devuelve con texto y esta vez escribiremos un “¡HAS ACERTADO!”.

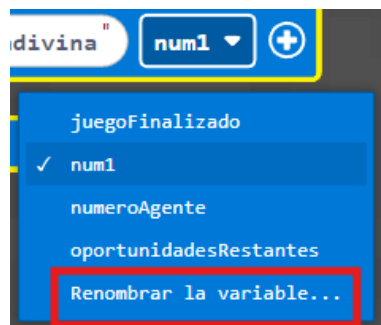
## Parte 4. Comprobando los resultados

Una vez tenemos hecha la función que comprueba los resultados, vamos a revisar que repuestas nos devuelve la función y a hacer que el agente actúe dependiendo del resultado del intento.

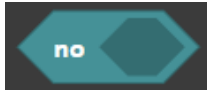
Lo primero que necesitaremos para comprobar los resultados es que haya un número que comprobar, así que vamos a programar una forma de recibir por chat el número que el jugador quiere adivinar. Para ello vamos a utilizar un bloque de “con el comando de chat” y le vamos a poner como comando “adivina”, pero no solo eso, si no que queremos que el jugador ponga “adivina” y seguido el número que quiere adivinar, como “adivina 25”, para poder hacer eso vamos a darle al botón de más del bloque “con el comando de chat”.



Al añadir una variable con el más, nos aparecerá una nueva variable llamada “num1”, nosotros vamos a renombrarla, para hacerlo le daremos al desplegable de la variable y seleccionaremos “renombrar”, la llamaremos “intentoJugador”.

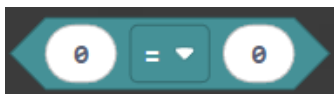


Una vez el jugador introduzca “adivina ...” en el chat, este comando se activará, pero solo queremos que compruebe si el número es el correcto en caso de que el jugador siga teniendo intentos o en caso de que no haya ganado ya la partida, así que, lo primero que vamos a añadir es un “sí”, con un “no”, para comprobar nuestra variable de “juegoFinalizado”. Usamos el no porque en caso de que el juego Sí haya finalizado, no queremos que el código haga nada.

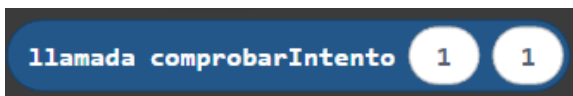


Si el juego sigue en marcha, queremos comprobar si el jugador ha acertado el número o no, así que vamos a añadir otro condicional “sí /si no”, dentro del otro.

Como primera opción, vamos a comprobar si el jugador ha acertado la respuesta, utilizando la comparación de igual.



Para cada valor de la comparación utilizaremos las variables de “intentoJugador” y “numeroAgente”. En caso de que sean iguales, le mostraremos al jugador un mensaje con “mostrar”, pero para el título, en lugar de poner nosotros un texto, pondremos el texto que viene de la función “comprobarIntento”. Para eso, en el hueco del título pondremos el bloque de valor que llama a la función.



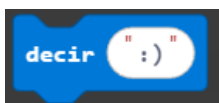
Recordad que debemos pasarle dos valores, el primero sería la variable del intento del jugador, y el segundo sería el número del agente. Esos dos valores, y siguiendo ese mismo orden, son los que en la función llamábamos “intento” y “respuesta”.

Como en este caso la respuesta era la correcta, también queremos marcar la variable de “juegoFinalizado” a verdadero, utilizaremos el bloque de “fijar” y el de “verdadero”.



¿Qué pasa si el intento del jugador no es correcto?

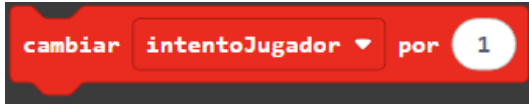
Lo primero que queremos que ocurra es que el jugador vea un mensaje en el chat indicándole si su acierto es muy alto o muy bajo, para ello vamos a llamar de nuevo a nuestra función de “comprobarIntento”, que puedes duplicar del bloque “mostrar” anterior, pero esta vez, en lugar de mostrarlo en pantalla vamos a ponerlo en el chat con un bloque de “decir”.



Y como el jugador a obtenido una respuesta incorrecta, queremos que el agente destruya uno de los bloques de la torre del aldeano, que tendrá justo enfrente, y suba un bloque hacia arriba, puesto que irá destruyendo los bloques desde abajo hacia arriba.



Además, reduciremos el número de intentos del jugador en uno, usaremos el bloque “cambiar” para darle “-1” a nuestra variable de “oportunidadesRestantes”.



Por último, después de restar un intento al jugador, vamos a comprobar con un “sí”, si los intentos del jugador son iguales a 0.

En caso de que así sea y el jugador haya acabado con todas sus oportunidades, lo primero que queremos hacer es utilizar el bloque “mostrar” para indicar al jugador que se le han acabado las oportunidades y ha perdido. Puedes poner el título y subtítulo que prefieras.

A continuación, mostraremos en el chat, con el bloque “decir” cual era el resultado correcto. Para ello necesitaremos de nuevo el bloque de “unir” del apartado de TEXTO, para poder poner “RESULTADO: “+ “numeroAgente” (refiriéndose a la variable).



Finalmente, queremos marcar nuestro “juegoFinalizado” como verdadero, así que podemos simplemente duplicar el bloque correspondiente que hemos hecho antes y añadirlo después de decirle el resultado al jugador.

Y ya hemos acabado nuestro juego, ya puedes lanzar el código y comprobar si funciona correctamente tratando de adivinar el número correcto sin mojar al aldeano. Y piensa, ¿De qué modos podrías mejorar el juego?

## Entrega

Una vez hayas finalizado el juego, asegúrate de:

1. Guardar el mundo correctamente.
2. Guarda el proyecto y realiza una captura de pantalla del código y del circuito



3. Presenta de forma oral el proyecto o haz un documento con las capturas y la explicación del código realizado.
4. Sube el archivo del proyecto y el documento, si es el caso, a Moodle.

**ANTES DE LA PRÓXIMA PRÁCTICA.**