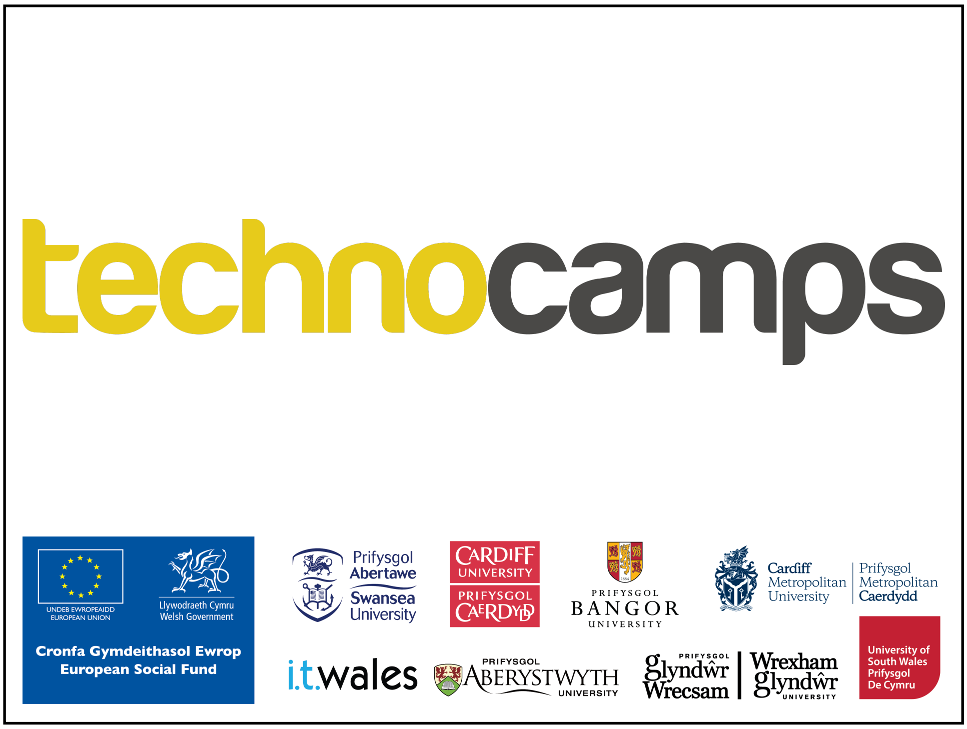
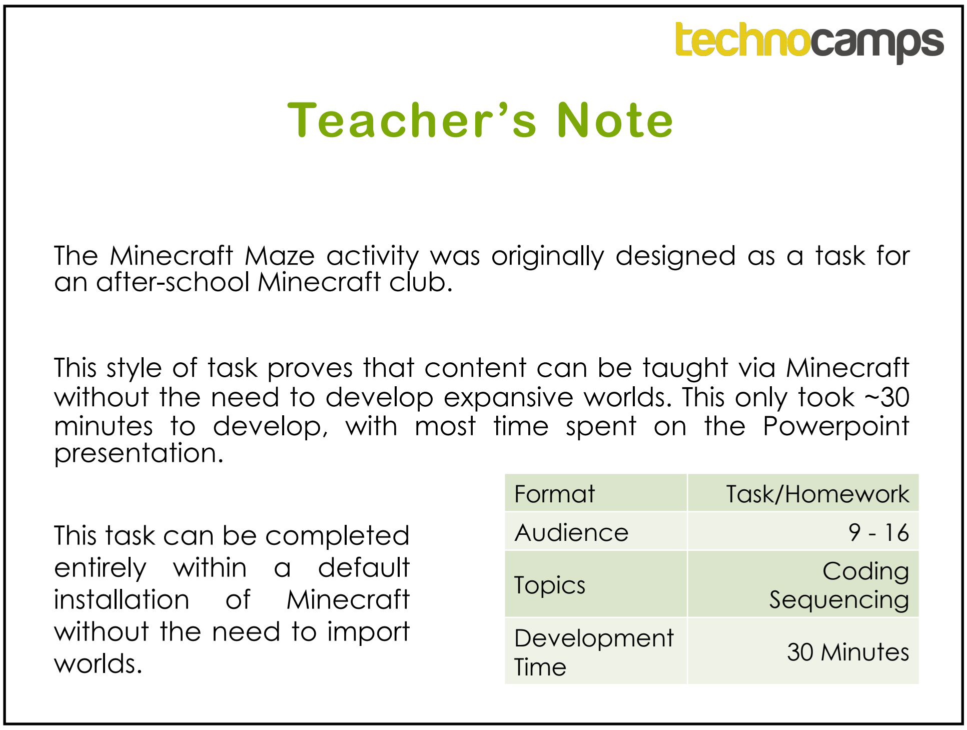


technocamps

UNDEB EWROPEAIDD EUROPEAN UNION
Llywodraeth Cymru Welsh Government
Cronfa Gymdeithasol Ewrop
European Social Fund

Prifysgol Abertawe Swansea University
CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD
PRIFYSGOL BANGOR UNIVERSITY
Cardiff Metropolitan University | Prifysgol Metropolitan Caerdydd
i.t.wales
PRIFYSGOL ABERYSTWYTH UNIVERSITY
PRIFYSGOL Glyndŵr Wrecsam | Wrexham Glyndŵr UNIVERSITY
University of South Wales Prifysgol De Cymru



Teacher's Note

The Minecraft Maze activity was originally designed as a task for an after-school Minecraft club.

This style of task proves that content can be taught via Minecraft without the need to develop expansive worlds. This only took ~30 minutes to develop, with most time spent on the Powerpoint presentation.

This task can be completed entirely within a default installation of Minecraft without the need to import worlds.

Format	Task/Homework
Audience	9 - 16
Topics	Coding Sequencing
Development Time	30 Minutes

Mazes!

Mazes make for good Minecraft challenge maps, but can take a long time to build.

If we generate them **Automatically** we can avoid most of the work.

There are many different **Algorithms** for generating mazes.

We're going to use the Agent to implement one of the basic algorithms.

Getting Started

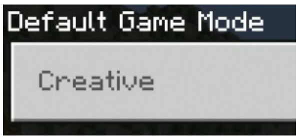
1. In the **Play** menu, click **Create New** and then **New**

2. Give your world a name and change the game mode to **Creative**

3. Scroll down until you find **World Type**

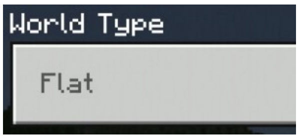
4. Change World Type to **Flat**

5. Click Play



Default Game Mode

Creative



World Type

Flat

technocamps

Getting Started

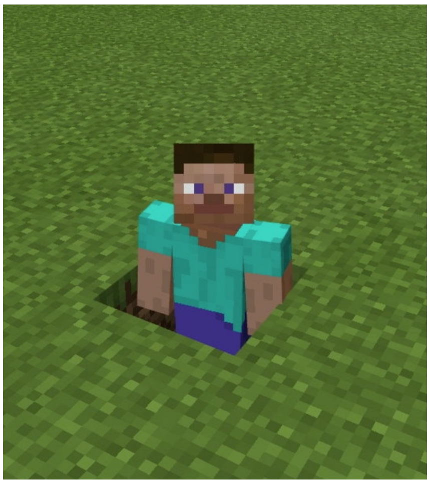
7. Dig one block down

8. Stand in the hole

9. Open the Code Builder

   (Press '**C**' on the keyboard or Agent icon on iPad)

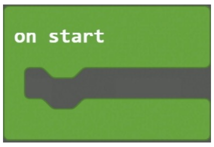
10. In the Code Builder add an **Event Block** that will trigger your code



technocamps

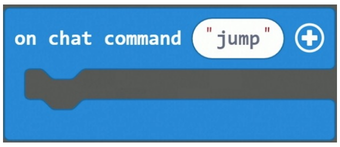
Events

Events are special blocks with a **flat top**. The code inside these blocks runs when the event is triggered.



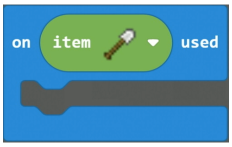
```
on start
```

Code runs when **Play** button is pressed.



```
on chat command "jump" ⊕
```

Code runs when the player types 'jump' in chat



```
on item ▾ used
```

Code runs when player uses an **Iron Shovel**

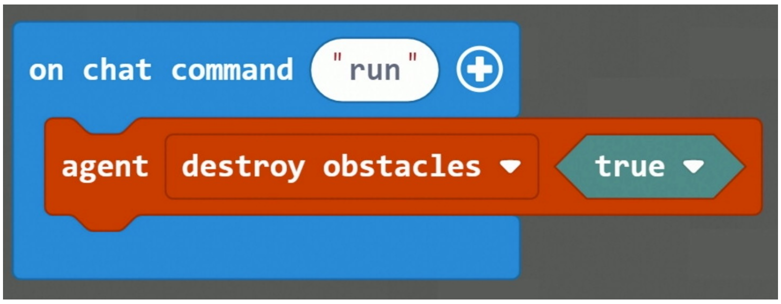
Algorithm

Inside your event you'll have to **implement** this algorithm. For a challenge, have a go coding this on your own.

If not, carry on to the next slide.

```
01    Tell the Agent to destroy any block in its way
02    Teleport the Agent to the Player
03    If there is a block in front of the agent:
04          Move forward 2
05    Turn a random direction
06    Repeat steps 3 to 5 as many times as you like
```

1. Tell the Agent to destroy any block in its way



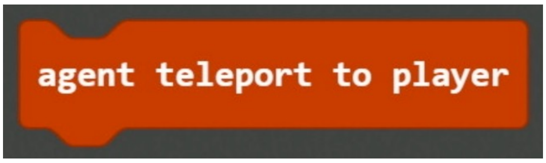
```
on chat command "run" ⊕
    agent destroy obstacles ▾ true ▾
```

(**hint:** If you can't find this block, take a close look at `agent place on move false`

If you click the drop-down menu you'll find what you're looking for.)

technocamps

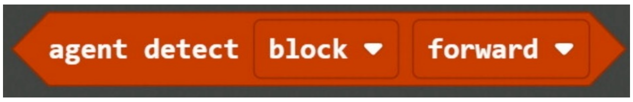
2. Teleport the Agent to the Player



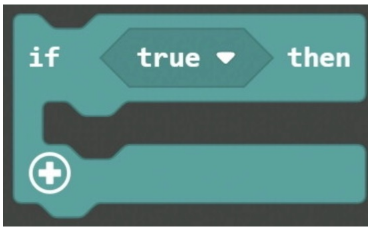
```
agent teleport to player
```

technocamps

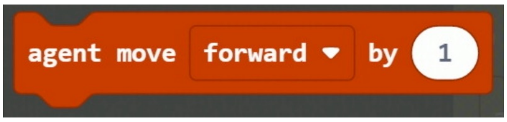
3. If there is a block in front of the agent: Move Forward 2



```
agent detect block ▾ forward ▾
```



```
if true ▾ then
```



```
agent move forward ▾ by 1
```

4. Turn a random direction

This step is the trickiest. We need to break some of the rules of MakeCode to get it to work.

Usually the agent turn left ▾ block only accepts the words **Left** or **Right**.

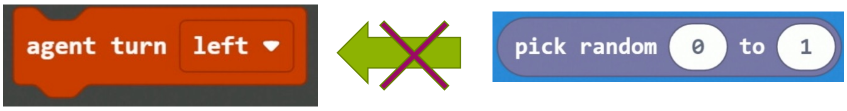
Under the bonnet, Minecraft says that **Left is 0** and **Right is 1**.

So all we have to do is randomly choose between 0 & 1 and put it in the **agent turn** block.

Easy Right?

Not Easy!

MakeCode won't let us! In typed programming languages this is very common. But here we can't put **pick random 0 to 1** in **agent turn**.



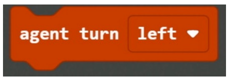
agent turn left ▾

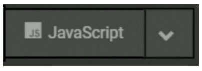
pick random 0 to 1

Even though we definitely should be allowed to because **Left = 0** and **Right = 1.**

Instead we have to trick MakeCode into letting that happen by changing into JavaScript.

Being Tricksy

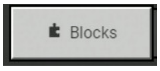
1. Put down an  block under your **if** block.

2. Click  to change into JavaScript

3. Find **`agent.turn(LEFT_TURN);`**
4. Delete `LEFT_TURN`
5. Type **`randint(0, 3)`** where LEFT_TURN used to be
6. Check your code

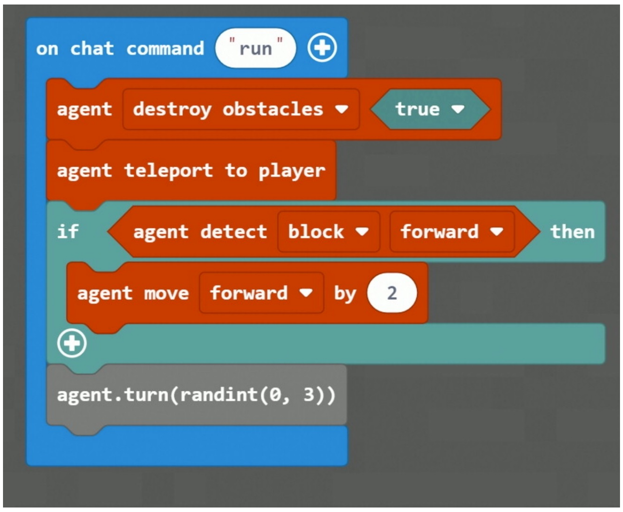
The Code

```
agent.turn(randint(0, 3))
```

If that's all good, we can go back to 

technocamps

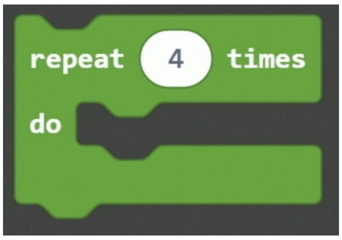
Code Check



```
on chat command "run" ⊕
    agent destroy obstacles ▾ true ▾
    agent teleport to player
    if agent detect block ▾ forward ▾ then
        agent move forward ▾ by 2
    ⊕
    agent.turn(randint(0, 3))
```

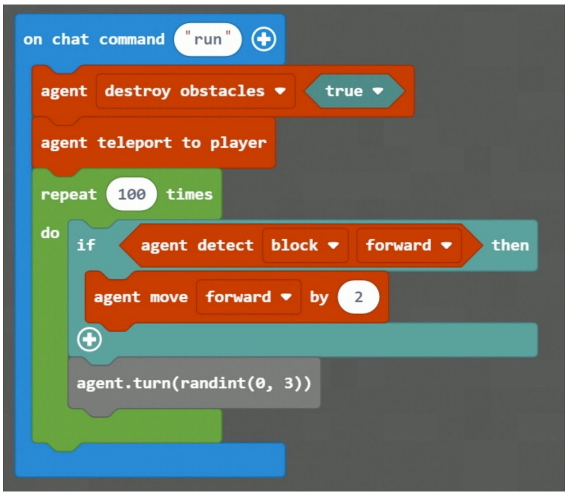
technocamps

5. Repeat steps 3 to 5 as many times as you like



```
repeat 4 times
do
```

Final Code

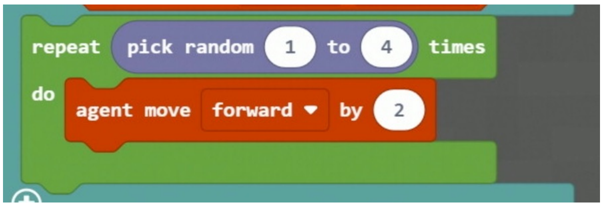


```
on chat command "run"
    agent destroy obstacles true
    agent teleport to player
    repeat 100 times
    do
        if agent detect block forward then
            agent move forward by 2
        agent.turn(randint(0, 3))
```

Extension 1

At the moment our Agent will always move forward 2 spaces. This makes a very twisty maze.

If we want a maze that's a bit more random we can randomly change **how many times** we go forward 2.



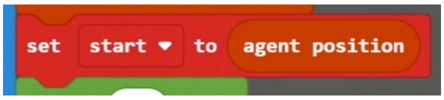
```
repeat pick random 1 to 4 times
do
    agent move forward by 2
```

Extension 2

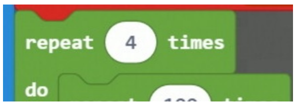
To improve our Maze even more, we need **Branching Paths**.

To do that, we can run our algorithm again and again, starting from where we began.

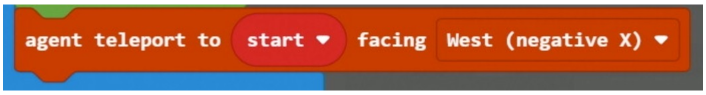
For that, we'll need to remember where we started.



```
set start ▾ to agent position
```



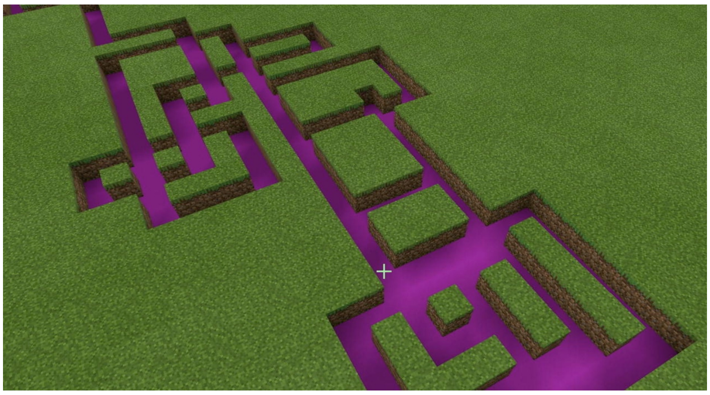
```
repeat 4 times
do
```



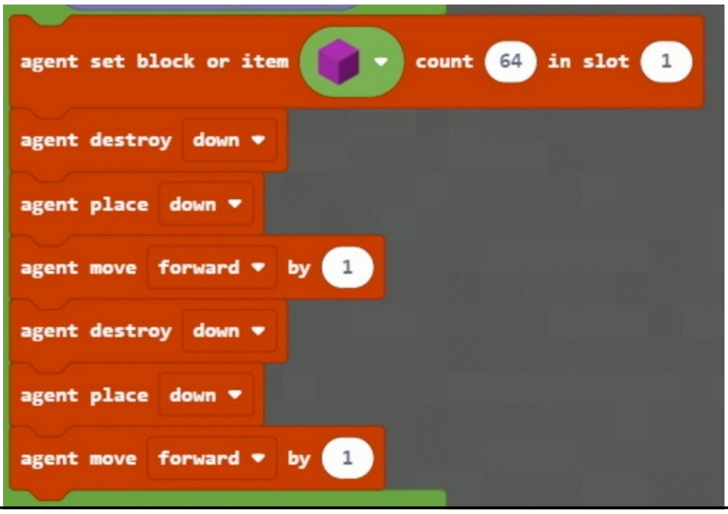
```
agent teleport to start ▾ facing West (negative X) ▾
```

Extension 3

Why not get the Agent to place some blocks as it creates the maze so we can line it?



Extension 3



```
agent set block or item ▾ count 64 in slot 1
agent destroy down ▾
agent place down ▾
agent move forward ▾ by 1
agent destroy down ▾
agent place down ▾
agent move forward ▾ by 1
```

Extension 4

Take a look at this website, watch the animations and answer these questions:

https://professor-l.github.io/mazes/

1. Which algorithm is your favourite (i.e. looks the coolest)?
2. Which maze looks the most random?
3. Which mazes look like a human has made them?
4. Which mazes look like a computer has made them?

Now take a look at the video below and try implementing that Algorithm.

https://www.youtube.com/watch?v=UGIWiyEieso