Crab Maze

Program Aim Create a maze game where the crab sprite steers through a maze, collecting coins and score until it reaches the maze end which takes it to another level. If it hits the wall the game ends

Computer Science Concepts

-Decomposition

-Variable to collect score

-Loops

-Conditional selection

Maths Concepts

-Decimal fractions

–Angles

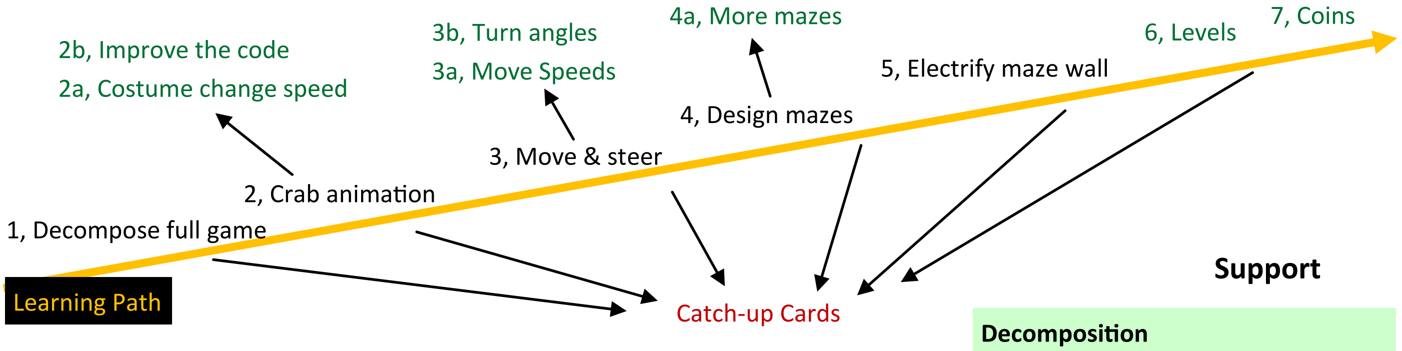
-X & Y Coordinates

Scratch 1.4 or Scratch 2.0

This planning is written in 1.4 but all the blocks are the same as 2.0. I always start with 1.4 as I think it is easier especially in the drawing aspects

Differentiation and Assessment for Learning This planning should be used alongside the *catch-up cards* supporting SEN pupils and the *extension cards* supporting the more able, although in this module a lot of the extension tasks can be given orally. At the beginning of each session the *learning intention sheet* is shared and the learning journey expanded through success criteria. Pupils feed their progress back to the teacher through annotating this sheet with smiley faces at the end of the lesson. Teachers can also annotate the sheet to indicate those who need more or less help in future lessons. These extra resources can be found on the code-it.co.uk website.

Extension



2b, Improve the code

2a, Costume change speed

3b, Turn angles

3a, Move Speeds

4a, More mazes

7, Coins

6, Levels

5, Electrify maze wall

4, Design mazes

3, Move & steer

2, Crab animation

1, Decompose full game

Learning Path

Catch-up Cards

Support

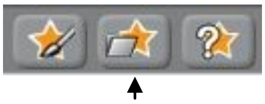
Decomposition

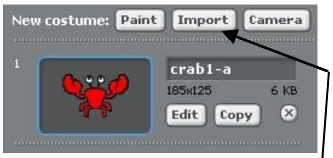
Is breaking a program up into smaller chucks and solving these separately.

1, Decompose the full game

Open a modern web browser and navigate to http://scratch.mit.edu/projects/22804830/ Play the game whist pupils watch (don’t show them the code) ask them to work in pairs to list all the things that they will need to make and all the things they will need to make it do. After pupil pairs discussion, collect and list these things. Explain that we have decomposed the game, broken it up into smaller chunks to solve them separately. Explain that this is something programmers do to help make problem solving manageable and sometimes so that large groups of programmers can work on the same project at the same time. Refer to their list as you guide them through creating the game.

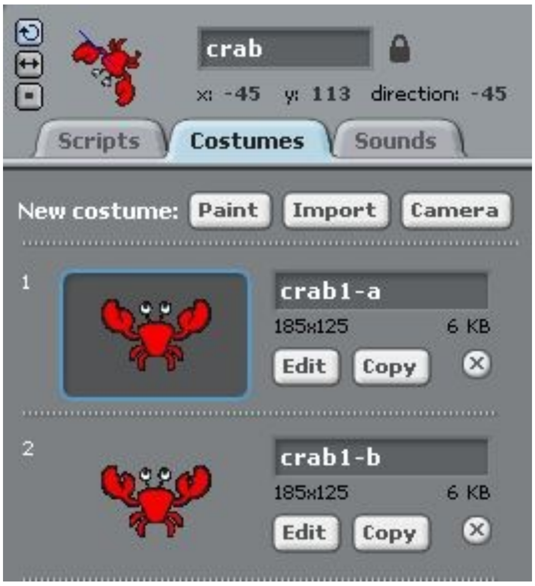
2, Crab Animation





Open Scratch, right click on the cat and delete the cat. Click on the choose new sprite from file button and navigate to the animals folder. Inside select the crab1-a costume and click ok. Now navigate to the costumes tab inside the crab sprite and import the crab1-b costume.

Crab Maze P2



crab

x: -45 y: 113 direction: -45

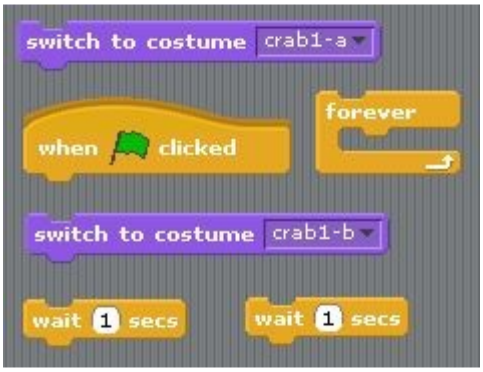
1 crab1-a 185x125 6 KB

2 crab1-b 185x125 6 KB

2 Crab animation continued

Click between the two pictures inside the crab sprite. Ask pupils what they can see? (crab claws open and close) Explain that we would like to automate that process so that the crab opens and closes its claws as it moves.

Drag out all these blocks and ask pupils to snap them together to make the crab pictures change. Make sure pupils know the link between the names of the costumes and the names on the switch to costume block. Explain that if it doesn't change over and over again they have got the order wrong. Don't help anyone as struggle is good. If someone drags out the wrong blocks get them to compare theirs to yours on the board or on the catch-up card. Can they spot which block is wrong? Make a fuss of them when the fix it as this is debugging their code.

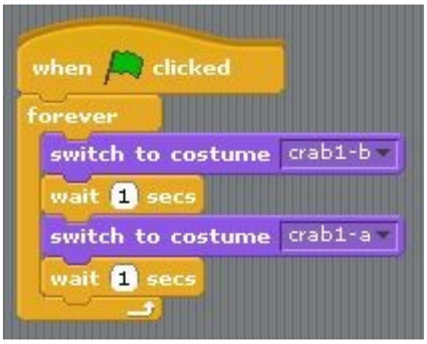


```
switch to costume crab1-a

when 🏴 clicked          forever

switch to costume crab1-b

wait 1 secs          wait 1 secs
```



```
when 🏴 clicked
forever
    switch to costume crab1-b
    wait 1 secs
    switch to costume crab1-a
    wait 1 secs
```

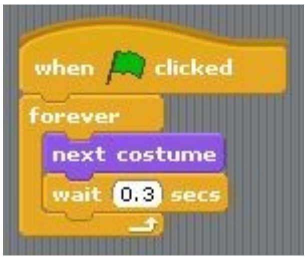
The forever block could start with a wait

2a, Costume change speed

Challenge pupils to make the claw costumes change faster without using a 0. They can then investigate how using a decimal fraction will speed the change up.

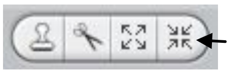
2b, Improve the code

Can they improve on the code by using just one costume block and one wait block?



```
when 🏴 clicked
forever
    next costume
    wait 0.3 secs
```

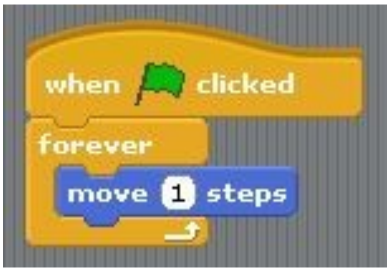
One block solution



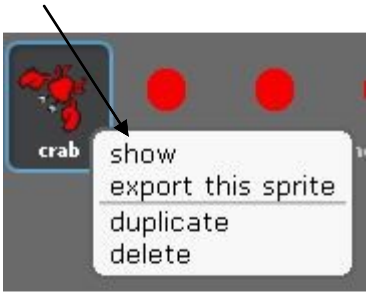
Use shrink key to reduce size of crab

3, Move and Steer

Drag out these blocks and ask pupils to snap these together, unconnected to their last blocks, to make the crab move. Show them how they can right click on the crab and select **show** to return it to the middle if it goes out of sight. Demonstrate this by walking one step over and over in a loop.



```
when 🏴 clicked
forever
    move 1 steps
```

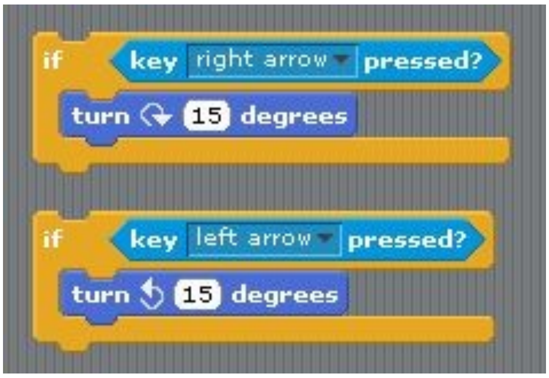


3a, Challenge them to slow the crab down

They can do this by reducing the move to less than 1 without using 0.

Crab Maze P3

3 Move and steer continued
Explain that we now want the computer to turn right if the right arrow keyboard input is pressed and left if the keyboard left arrow key is pressed. Some children might suggest the **when space key is pressed** block. Explain that this is quite slow to use within games.

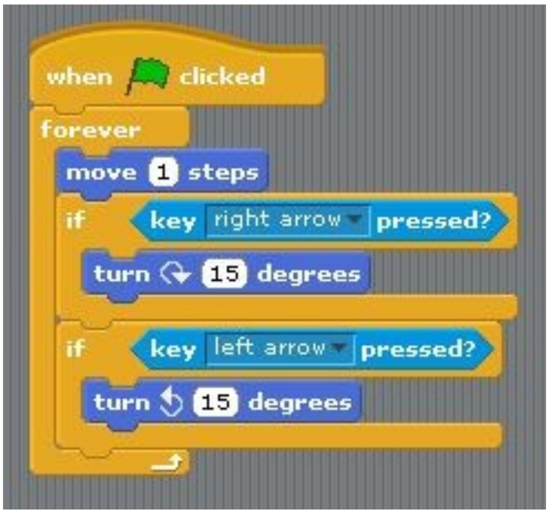


```
if key right arrow pressed?
    turn ↻ 15 degrees

if key left arrow pressed?
    turn ↺ 15 degrees
```

Introduce the idea of conditional selection
If you haven't introduced this idea before you can use these real examples before using the coding ones.
http://code-it.co.uk/unplugged/ConditionalSelection.pdf

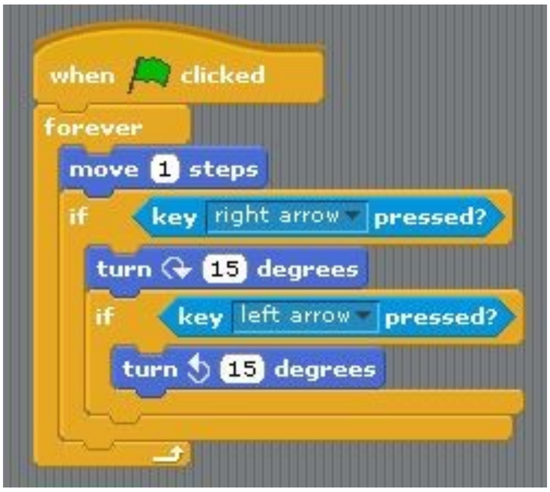
Now demonstrate how to make two steering blocks but before pupils go and make these themselves ask them how we can get the program to check if the condition is met over and over again. If you have emphasised the forever loop enough before pupils will suggest using a forever loop. Drag the blocks into the forever loop with the move block. You can ask why we wouldn't use the costume change block? (Answer: The time wait will slow all the blocks down)



```
when 🏁 clicked
forever
    move 1 steps
    if key right arrow pressed?
        turn ↻ 15 degrees
    if key left arrow pressed?
        turn ↺ 15 degrees
```

Common Errors

The most likely error is to drag an if block inside another one. Ask pupils to check to see if their **if** blocks are lined up underneath each other if you spot this error. In this example the left arrow condition will only be checked if the right arrow has been pressed.



```
when 🏁 clicked
forever
    move 1 steps
    if key right arrow pressed?
        turn ↻ 15 degrees
        if key left arrow pressed?
            turn ↺ 15 degrees
```

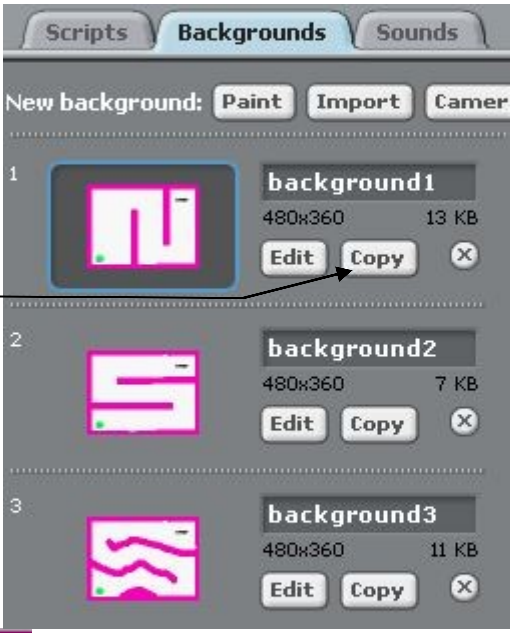
3b, Investigate angles other than 15 degrees
Ask pupils if 15 degrees is the best turn amount. Get them to investigate other turn angles.

4, Design Mazes
Click on **stage** and **backgrounds** and demonstrate creating a maze background using **only** one colour for the walls. Either use a different colour for the start place and another for the end of the maze or use text for the start. Make sure you include a border all round the edges so the crab cannot leave the maze. Make sure pupils know that the one colour is so that code will run later not just your whim.

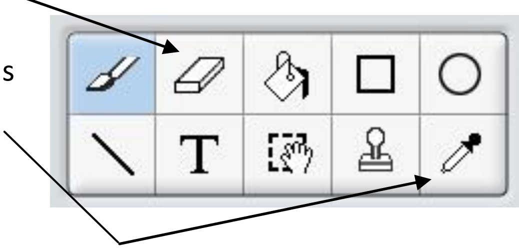
Crab Maze P4

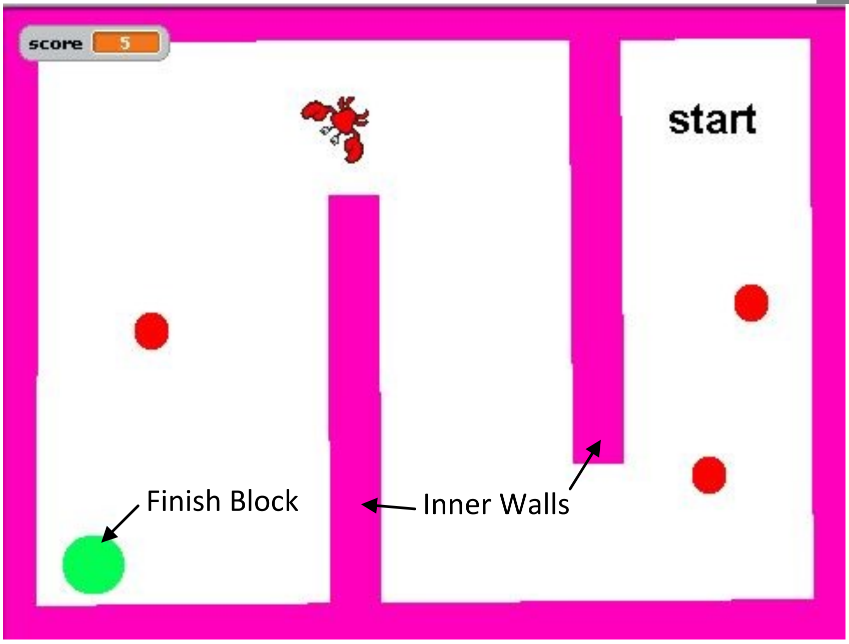
4, Design Mazes Continued

Once pupils have created a maze show them how they can copy and edit a maze to change the inner walls but leave the outer walls and start and finish place the same.

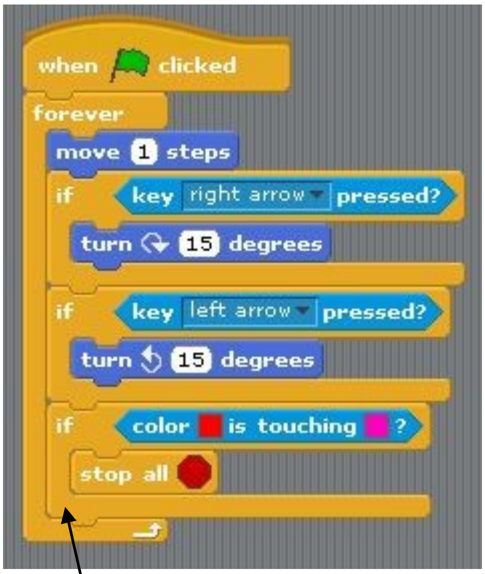


Use the eraser tool to rub out inner walls and the eyedropper tool to select the same wall colour





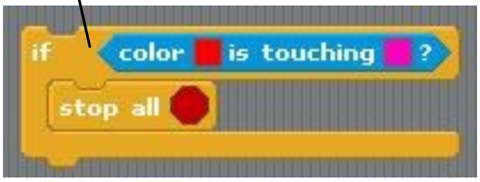
Allow pupils plenty of time to create mazes. Check that they use the same colour walls, start and finish blocks on every maze.



```
when 🏁 clicked
forever
    move 1 steps
    if key right arrow pressed?
        turn ↻ 15 degrees
    if key left arrow pressed?
        turn ↺ 15 degrees
    if color 🟥 is touching 🟪 ?
        stop all ⬣
```

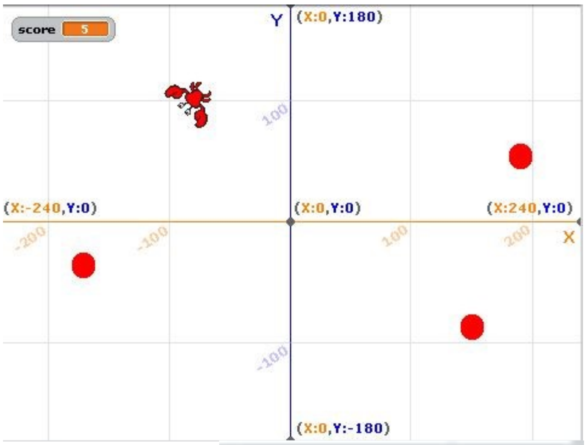
5, Electrify Wall Maze

Remind pupils of their decomposed task to end the game when the crab touches the walls. Drag out an if block and talk pupils through creating the code as shown. Make sure you show them how to change the colour by clicking in the squares and the selecting the colours using an colour eyedropper tool. The first colour is the crabs followed by the colour of the wall. Finally ask pupils where we can place this so it will be checked over and over again? Incidentally this could be its own forever loop but we use less code this way.



```
if color 🟥 is touching 🟪 ?
    stop all ⬣
```

Crab Maze P5

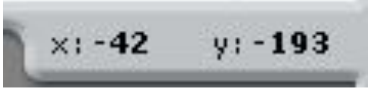


6, Levels Spawn Point

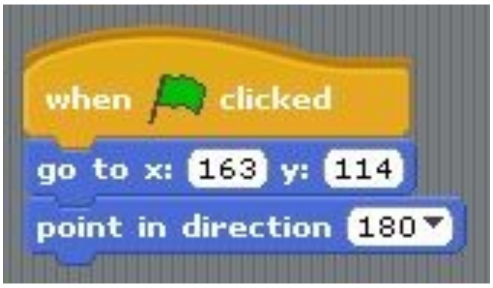
Start by asking pupils what is the problem with ending our game by touching the walls at the moment? Pupils should identify that the crab is left touching the wall so when the game is restarted it ends straight away. Explain that we can create a spawn point so that the crab will always start the game from the same place.

Stage, backgrounds, import xy-grid

Point out that everywhere that we position the cursor on the background has a grid reference. Point out the changes on the x and y indicator just to the bottom right of the screen. Start in the centre of the screen and move up, ask pupils to note what is happening to the Y axis. Now go back to 0 in the middle and ask them to predict what will happen if we go down? (negative numbers) relate to other negative numbers like temperature. Repeat with the X axis, moving right first. Explain that this is good to know but that there is an easier way to find the X and Y position of everything on the screen. Open the motion blocks and ask them to watch the **go to x and y** block. Now move the crab and double click on it. The go to x and y block will change. Do this lots of times until every child has seen this change. Now use this block and the point in direction block to create a spawn point.



x: -42 y: -193



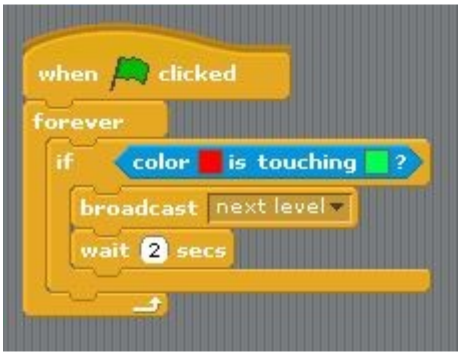
```
when [flag] clicked
go to x: 163 y: 114
point in direction 180
```

Common bugs

Errors at this level nearly always come down to pupils not seeing the X and Y block change when the crab is double clicked. Some pupils will copy your X and Y otherwise.

6, Levels

Explain that we can now use the X and Y and the if colour is touching to change levels. Gather all those ready to move on and whisper to two children that when you say **change level** they must wave their hands in the air. Explain to all the pupils that we are going to use a broadcast signal to change the level. Make sure you include the word **change level** lots of times. Create code that says if colour red of crab touches colour of end of level then broadcast. Show them that is doesn't do anything. Explain that this is like them at the moment they can receive the change level broadcast but they don't have any code to know what to do with it. The two children who you whispered to by contrast will have been programmed to wave when we broadcast change level.

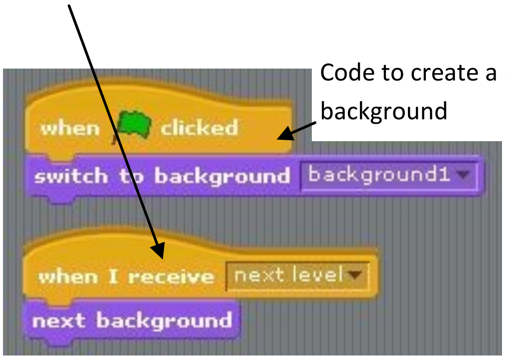


```
when [flag] clicked
forever
  if color 🟥 is touching 🟩 ?
    broadcast next level
    wait 2 secs
```

Crab Maze P6

6, Levels Continued

Now show the pupils how they can pick up the broadcast in the stage area to change the background. Then show them how we can also use the X and Y code to receive the broadcast and pick up the change level and return the crab to the start position.

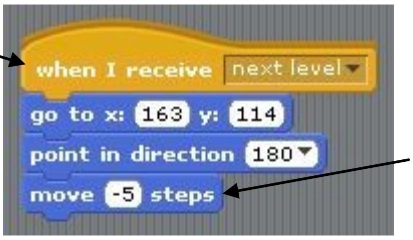


```
when 🟩 clicked
switch to background background1

when I receive next level
next background
```

Code to create a starting background

Code in the stage scripts area



```
when I receive next level
go to x: 163 y: 114
point in direction 180
move -5 steps
```

Extra code to move crab back, slightly slowing its entrance

Code in the crab scripts area

Letting pupils look at my example on the computer and click between stage and crab helps

7, Coins

See the coins sheet to help pupils create coins that can be eaten by the crab to increase its score.

7a, More adaptations

This game can be easily adapted by adding multiple moving characters or you could use the primary games maker to help pupils code more adaptations.

http://code-it.co.uk/scratch/primarygamesmaker/ScratchPrimaryGamesMaker.pdf

Mistakes
I do my best to avoid mistakes but if you spot one please email me through address on the website stating which page and module it was on.