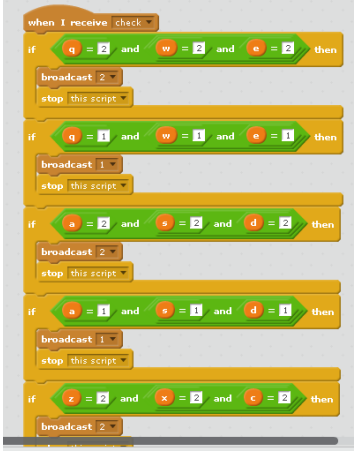
Criterion C: Development

Structure of the product:

The structure of the product was made in Scratch using arrays and lists and having various costumes available for sprite I order to represent the status of a place. I found the rules for the game of Nine Men's Morris from Wikipedia[1] so I could write algorithms to determine the way the game must function. When programming the game, I looked through existing projects on Scratch which had similar programming structures such as Noughts and Crosses. I used these to aid my use of lists and arrays. I originally made the game Noughts and Crosses in order to get an idea as to how Nine Men's Morris will operate and I used a project by "starswap"[2] to help me understand. Originally I looked at how they detected a three-in-a-row however because my project encountered so many issues, I was unable to do so. If I were to programme this, it would be simple as the coding would be exactly the same, however have to be repeated for both players.



Why it's appropriate:

Scratch was appropriate to use for programming Nine Men's Morris as it is primarily event driven and sprite based. I was able to create sprites and backdrops quickly without needing to write code for their appearance. This was useful as there were many costumes for each sprite and this would take longer to programme if I were to use Java and therefore I would not have much time to then focus on gameplay.

Scratch I also found was simple to store variables, array and lists in which I frequently used. Within the backdrop I had to programme an array which set an unoccupied place to '0' when it was empty and using Scratch made it easy to clone code rather than have to continuously type and I find this useful to help concentrate on just the programming side of the project rather than having to spend a lot of time over writing the exact correct syntax.

The fact objects can interact with each other by exchanging messages was also helpful to me as the game needed to detect whether a move was valid due to which sprites are already in place, whose turn it is and how many sprites there are remaining.

Algorithmic thinking in development of product:

In order to go about programming the product I first had to think logically and write ideas down on paper. Before jumping straight into programming I had to consider what data was involved and how it needed to be approached. I then had to decide what data interacted with each other and how I

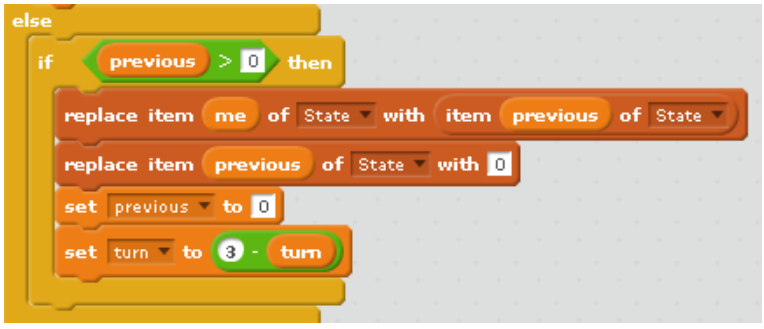
would manipulate it to carry out its necessary actions. The data involved was the status of a place, such as its costume, and the status of the variable canMove.

Initially I thought having two separate sprites to represent either player would work seeing as I thought I would make the sprites appear on both sides of the game and the player could then drag their piece to the board. I soon realised I was unable to suitably code 'dragging a piece' in Scratch and therefore had to change the data to just one sprite which had different costumes and switching them depending on the status of the sprite. This was simpler to code algorithmically as I could then use 'if statements' effectively.

Data such as the variable canMove were more difficult to manipulate as there were many factors which had to be true I order for canMove to be true. Firstly I just programmed canMove to be true when the player had three in a row however it was challenging to find a way for the sprites to communicate between each other so instead I created another sprite labelled '3 in a row' where the player presses it when they have three sprites in a row. After this sprite has been clicked, I have programmed it to check if the place the player wants to move to is empty or not and this was much simpler to programme.

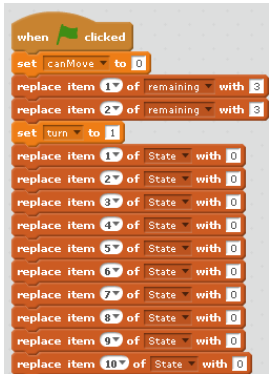
Techniques used:

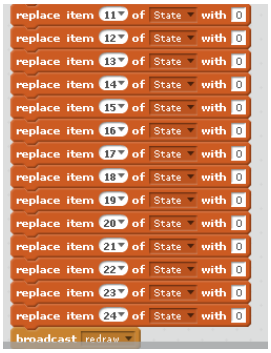
I used an array within the stage to store that the status of a place at the beginning of a game would be unoccupied. Then within each sprite I placed a piece of code stating the current status of the place, the desired place where the player wants to move to and then setting the place it's moved from to '0'. This involved the two variables 'me' and 'previous' where 'me' translates to the current place and 'previous' translates to the previous place. When the chosen sprite is clicked on it replaces 'me' with 'previous' and sets 'previous' to '0'. As previously stated I was unable to code for sprites to communicate between each other in order for them to recognise when there was a three in a row, therefore I used an additional sprite so once clicked, then the checks of whether a move is valid begin to run.

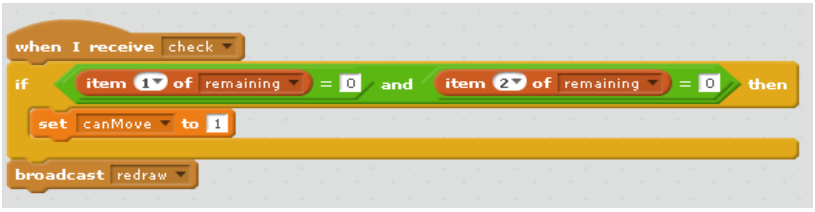


'Remaining' is a variable showing how many pieces there are yet to be laid for each player. The game begins with nine pieces each and the players take it in turns to lay their pieces until 'remaining' is equal to zero for both players. I achieved this through taking one away from the value already in remaining of the turn equivalent to the player. Once the value of both 'remainings' is equivalent to zero the board checks whether item one and item two of remaining are equivalent to zero, and if so there are no more pieces to be laid, they can now only be moved.

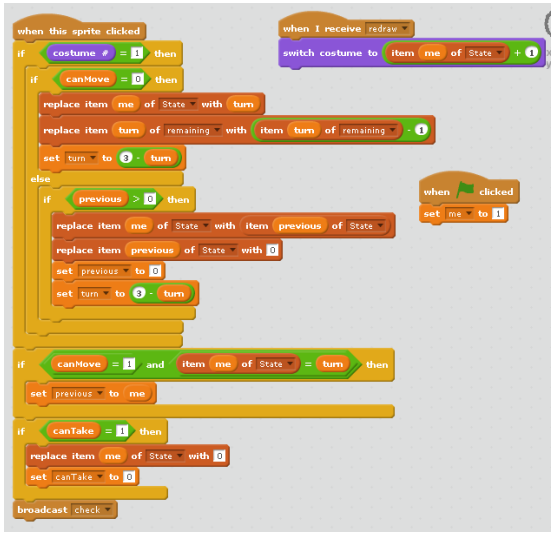
The board sets the players turn and replaces the state of a sprite with the number player which is on it as well as stating how many pieces each player has remaining to lay.







The Sprites assign the different coloured sprites to each player so it can change colour depending on whose turn it is or whether. The sprites also detect whether it is a valid place to be clicked by using if statements.



Why they're appropriate:

The reason I used an array is because I wanted to store a collection of values all in one set and it was then easier to call upon those values and use them in the code. Using variables to indicate a previous, current and next place ensures there is communication between sprites therefore recognising valid or invalid moves.