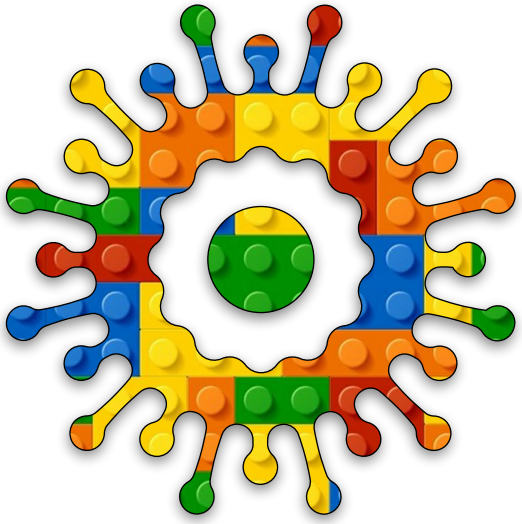
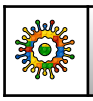
Making Games Module A Unit #7 noughts and crosses



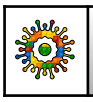
Content

Module A Unit #7 noughts and crosses (tic-tac-toe)



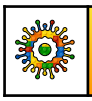
Introduction to noughts and crosses

There are three parts to this unit, which look at the different aspects of a turn-based game. It introduces the `minimax` algorithm to aid an AI version of the game.



Section 1: Self Playing Noughts and Crosses

In this first section, we will just get a working example of noughts and crosses (otherwise known as tic-tac-toe). There is no AI in this first part, but you will see the computer play against itself. We slow it down so that you can see each move.

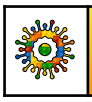


Sketch A7.1 empty board

Our starting sketch. Create an empty board with two players X and 0.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let player1 = 'X'
let player2 = '0'

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}
```



Sketch A7.2 pre-filled board

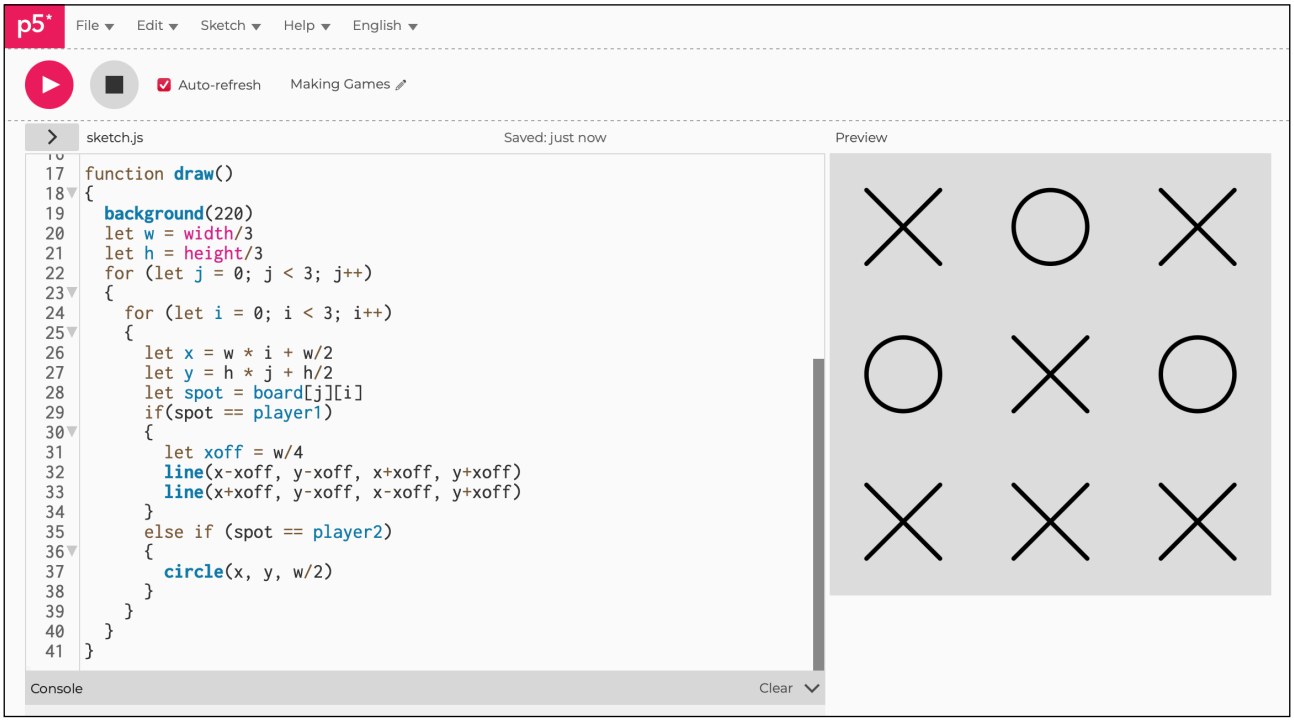
This is our basic board, and it is pre-filled to check that everything lines up and works as such. We draw a cross (X) for `player 1` and a circle (0) for `player 2`.

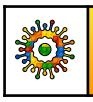
sketch.js

```
let board = [
  ['X', '0', 'X'],
  ['0', 'X', '0'],
  ['X', 'X', 'X']
]

let player1 = 'X'
let player2 = '0'

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
}

function draw()
{
  background(220)
  let w = width/3
  let h = height/3
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
```

```
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == player1)
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == player2)
      {
        circle(x, y, w/2)
      }
    }
  }
}
```

Figure A7.2


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Making Games
sketch.js
Saved: just now
Preview
function draw()
{
background(220)
let w = width/3
let h = height/3
for (let j = 0; j < 3; j++)
{
for (let i = 0; i < 3; i++)
{
let x = w * i + w/2
let y = h * j + h/2
let spot = board[j][i]
if(spot == player1)
{
let xoff = w/4
line(x-xoff, y-xoff, x+xoff, y+xoff)
line(x+xoff, y-xoff, x-xoff, y+xoff)
}
else if (spot == player2)
{
circle(x, y, w/2)
}
}
}
}
Console
Clear



Sketch A7.3 random selection

Let us have a random selection of which player goes first (and clear the array).

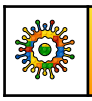
sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', '0']
let currentPlayer

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  currentPlayer = random(players)
}

function draw()
{
  background(220)
  let w = width/3
```

```
  let h = height/3
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == players[0])
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
}
```

Notes

Nothing to see here.



Sketch A7.4 the grid

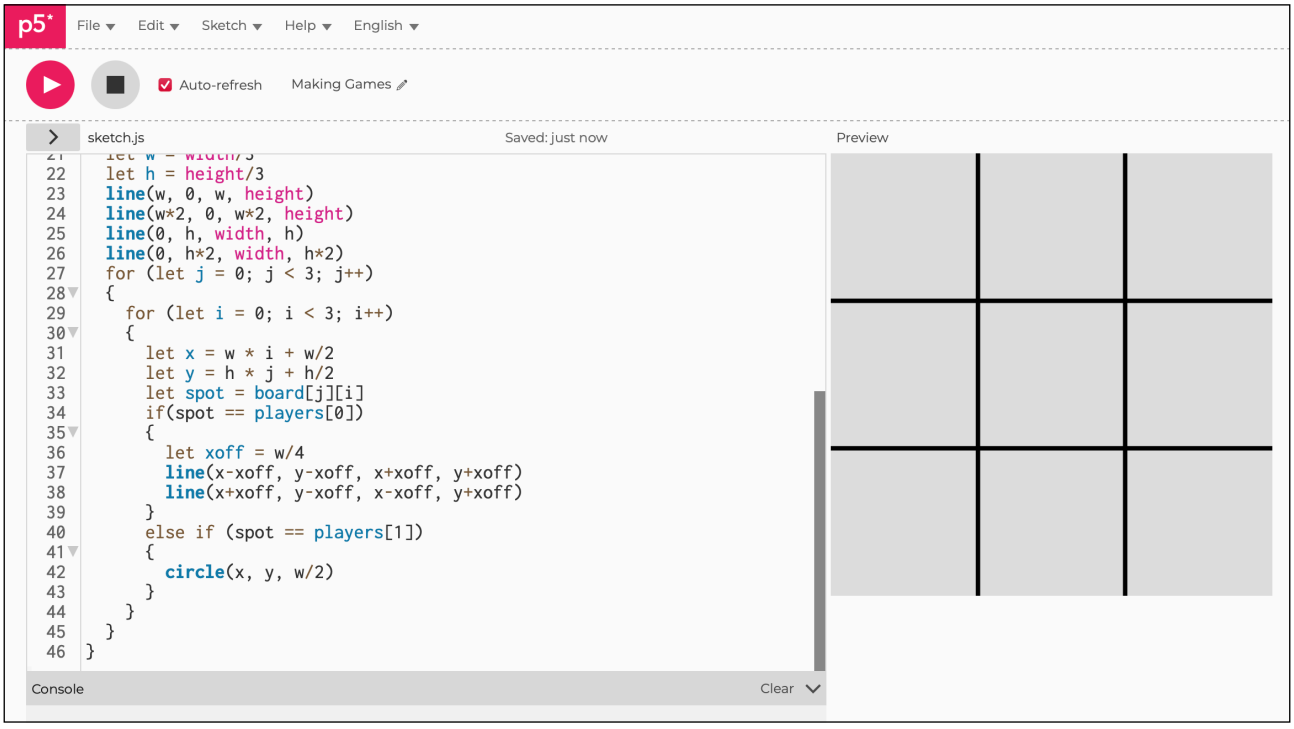
Drawing in the grid.

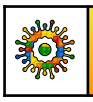
sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', '0']
let currentPlayer

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  currentPlayer = random(players)
}

function draw()
{
  background(220)
  let w = width/3
  let h = height/3
  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
```

```
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == players[0])
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
}
```

Figure A7.4


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Making Games
sketch.js
Saved: just now
Preview
let h = height/3
line(w, 0, w, height)
line(w*2, 0, w*2, height)
line(0, h, width, h)
line(0, h*2, width, h*2)
for (let j = 0; j < 3; j++)
{
for (let i = 0; i < 3; i++)
{
let x = w * i + w/2
let y = h * j + h/2
let spot = board[j][i]
if(spot == players[0])
{
let xoff = w/4
line(x-xoff, y-xoff, x+xoff, y+xoff)
line(x+xoff, y-xoff, x-xoff, y+xoff)
}
else if (spot == players[1])
{
circle(x, y, w/2)
}
}
}
Console
Clear



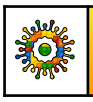
Sketch A7.5 available spots

The computer is going to play itself randomly until someone wins. First, create an `available[]` array; at this point, every point (`i`, `j`) is available.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', '0']
let currentPlayer
let available = []

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  currentPlayer = random(players)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      available.push([i, j])
    }
  }
}
```

```
function draw()
{
  background(220)
  let w = width/3
  let h = height/3
  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == players[0])
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
}
```



Sketch A7.6 next turn

Create a `nextTurn()` function. Click on the canvas repeatedly to generate a new turn.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', '0']
let currentPlayer
let available = []

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  currentPlayer = random(players)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      available.push([i, j])
    }
  }
}
```

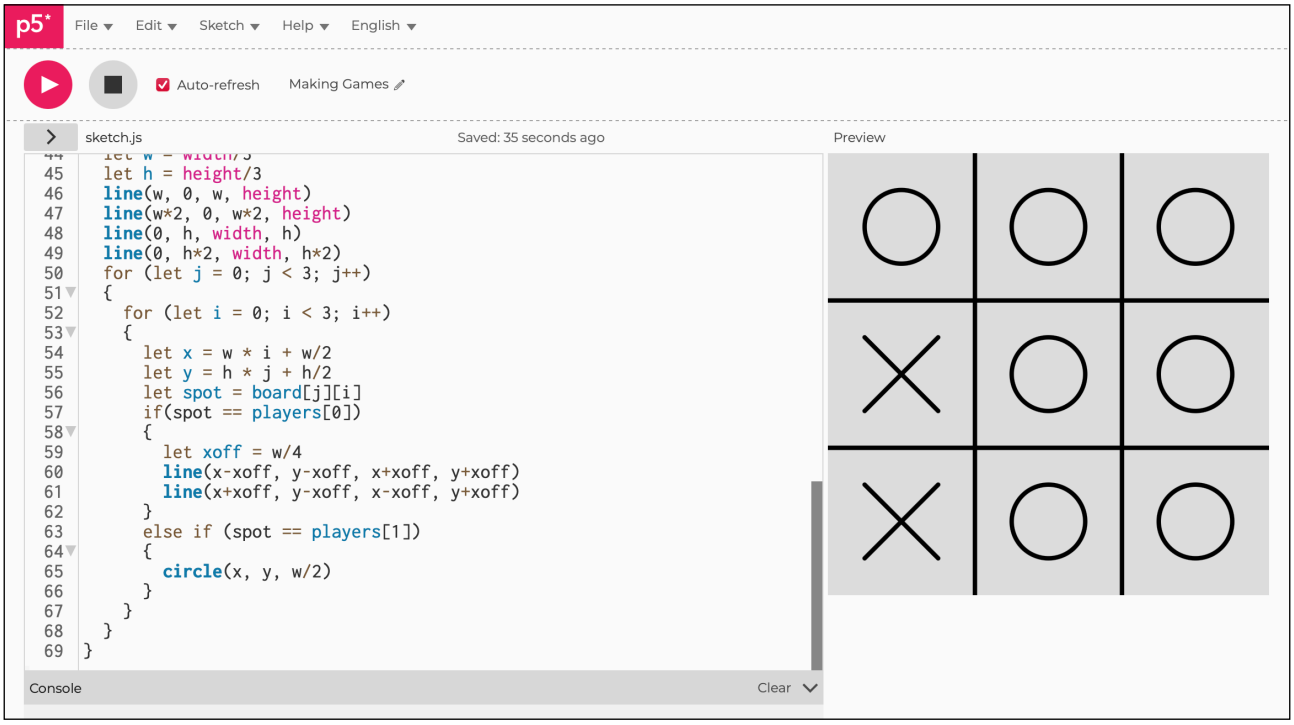
```
function nextTurn()
{
  let index = floor(random(available.length))
  let spot = available.splice(index, 1)[0]
  let i = spot[0]
  let j = spot[1]
  board[i][j] = currentPlayer
  currentPlayer = random(players)
}

function mousePressed()
{
  nextTurn()
}

function draw()
{
  background(220)
  let w = width/3
  let h = height/3
  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
```

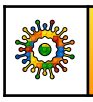
```
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == players[0])
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
}
```

📝 Notes

As you can see it makes no sense, they are not taking turns.

Figure A7.6


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Making Games
sketch.js
Saved: 35 seconds ago
Preview
let h = height/3
line(w, 0, w, height)
line(w*2, 0, w*2, height)
line(0, h, width, h)
line(0, h*2, width, h*2)
for (let j = 0; j < 3; j++)
{
for (let i = 0; i < 3; i++)
{
let x = w * i + w/2
let y = h * j + h/2
let spot = board[j][i]
if(spot == players[0])
{
let xoff = w/4
line(x-xoff, y-xoff, x+xoff, y+xoff)
line(x+xoff, y-xoff, x-xoff, y+xoff)
}
else if (spot == players[1])
{
circle(x, y, w/2)
}
}
}
Console
Clear



Sketch A7.7 in order

Now to place them in order, player 1 then player 2, etc., as you click on the canvas.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', '0']
let currentPlayer
let available = []

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  currentPlayer = floor(random(players.length))
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      available.push([i, j])
    }
  }
}
```

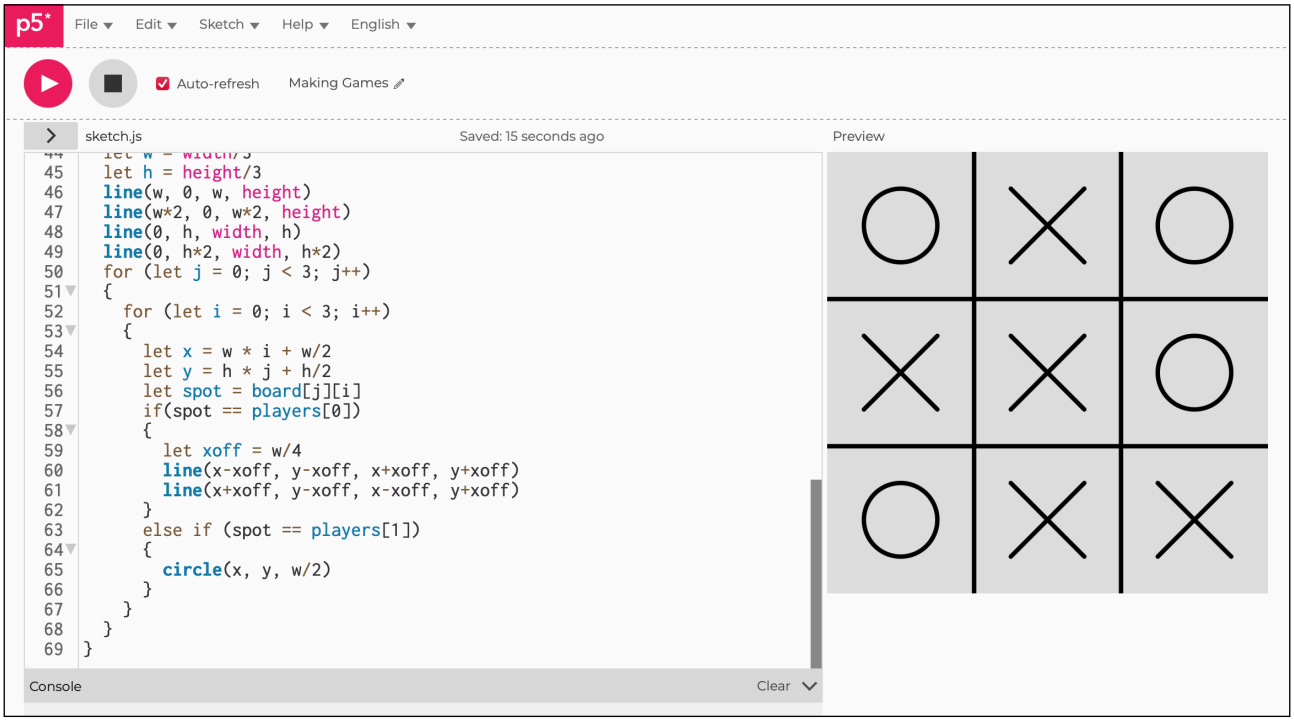
```
function nextTurn()
{
  let index = floor(random(available.length))
  let spot = available.splice(index, 1)[0]
  let i = spot[0]
  let j = spot[1]
  board[i][j] = players[currentPlayer]
  currentPlayer = (currentPlayer + 1) % players.length
}

function mousePressed()
{
  nextTurn()
}

function draw()
{
  background(220)
  let w = width/3
  let h = height/3
  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
```

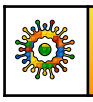
```
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == players[0])
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
}
```

🗒️ Notes

Now they take their turn.

Figure A7.7


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Making Games
sketch.js
Saved: 15 seconds ago
Preview
let h = height/3
line(w, 0, w, height)
line(w*2, 0, w*2, height)
line(0, h, width, h)
line(0, h*2, width, h*2)
for (let j = 0; j < 3; j++)
{
for (let i = 0; i < 3; i++)
{
let x = w * i + w/2
let y = h * j + h/2
let spot = board[j][i]
if(spot == players[0])
{
let xoff = w/4
line(x-xoff, y-xoff, x+xoff, y+xoff)
line(x+xoff, y-xoff, x-xoff, y+xoff)
}
else if (spot == players[1])
{
circle(x, y, w/2)
}
}
}
}
Console
Clear



Sketch A7.8 automated next turn

We don't need the `mousePressed()` to call `nextTurn()`, we can do it in `draw()`.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', '0']
let currentPlayer
let available = []

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  currentPlayer = floor(random(players.length))
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      available.push([i, j])
    }
  }
}
```

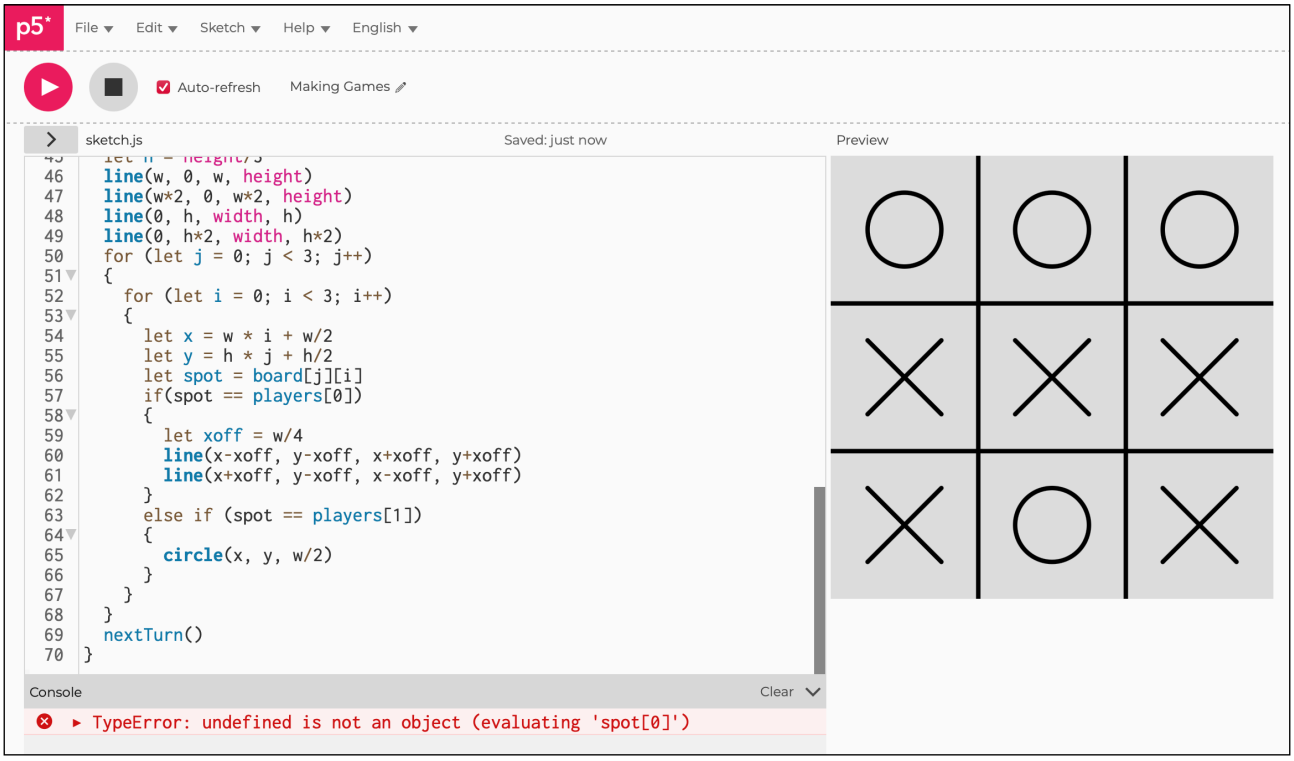
```
function nextTurn()
{
  let index = floor(random(available.length))
  let spot = available.splice(index, 1)[0]
  let i = spot[0]
  let j = spot[1]
  board[i][j] = players[currentPlayer]
  currentPlayer = (currentPlayer + 1) % players.length
}

// function mousePressed()
// {
//   nextTurn()
// }

function draw()
{
  background(220)
  let w = width/3
  let h = height/3
  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
```

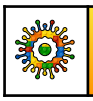
```
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == players[0])
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
  nextTurn()
}
```

📝 Notes

It will now fill up automatically but you will get an error message! Don't panic.

Figure A5.8


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Making Games
sketch.js
Saved: just now
Preview
line(w, 0, w, height)
line(w*2, 0, w*2, height)
line(0, h, width, h)
line(0, h*2, width, h*2)
for (let j = 0; j < 3; j++)
{
for (let i = 0; i < 3; i++)
{
let x = w * i + w/2
let y = h * j + h/2
let spot = board[j][i]
if(spot == players[0])
{
let xoff = w/4
line(x-xoff, y-xoff, x+xoff, y+xoff)
line(x+xoff, y-xoff, x-xoff, y+xoff)
}
else if (spot == players[1])
{
circle(x, y, w/2)
}
}
}
nextTurn()
}
Console
Clear
TypeError: undefined is not an object (evaluating 'spot[0]')



Sketch A7.9 check for a winner

Need to find a way to check for a winner, first a tie if none are available.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', '0']
let currentPlayer
let available = []

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  currentPlayer = floor(random(players.length))
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      available.push([i, j])
    }
  }
}
```

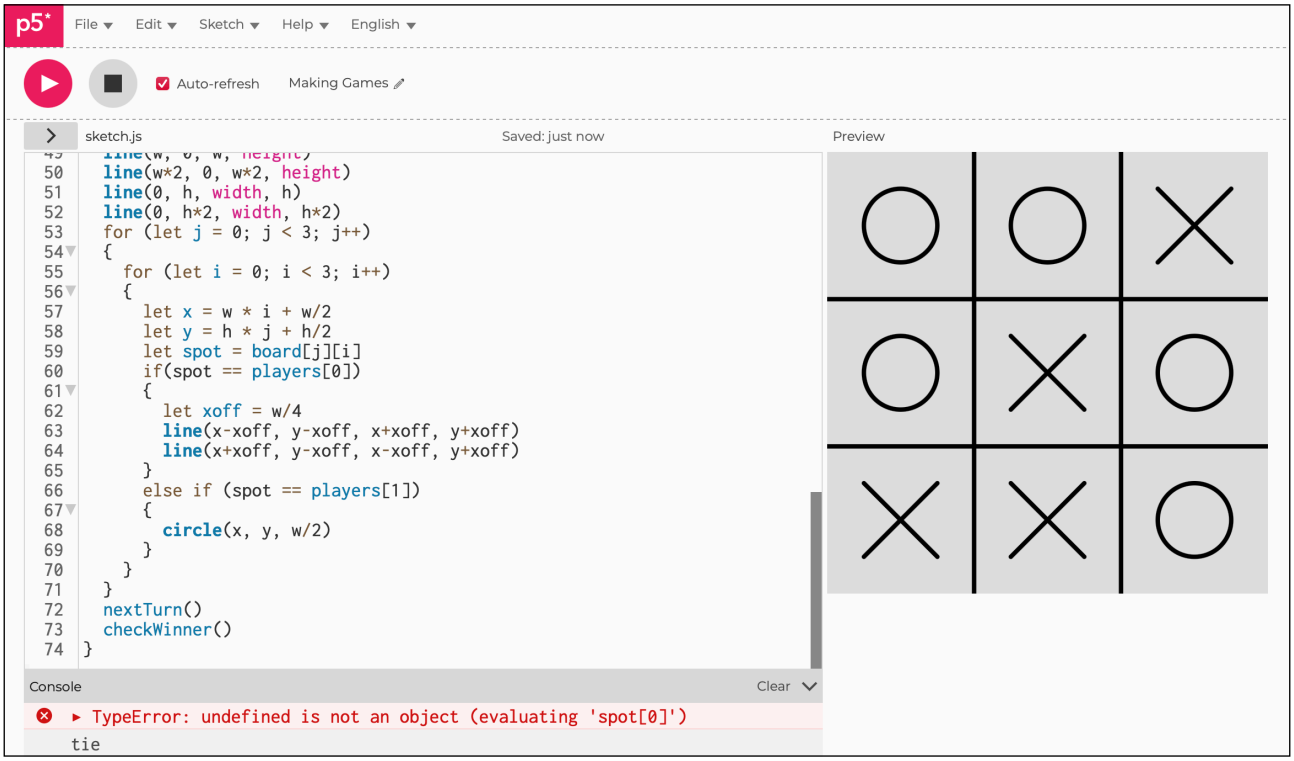
```
function checkWinner()
{
  if (available.length == 0)
  {
    console.log('tie')
  }
}


function nextTurn()
{
  let index = floor(random(available.length))
  let spot = available.splice(index, 1)[0]
  let i = spot[0]
  let j = spot[1]
  board[i][j] = players[currentPlayer]
  currentPlayer = (currentPlayer + 1) % players.length
}

function draw()
{
  background(220)
  let w = width/3
  let h = height/3
  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
```

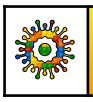
```
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == players[0])
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
  nextTurn()
  checkWinner()
}
```

🗒️ Notes

You should get `tie` every time at this point just after the error message.

Figure A7.9


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Making Games
sketch.js
Saved: just now
Preview
line(w*2, 0, w*2, height)
line(0, h, width, h)
line(0, h*2, width, h*2)
for (let j = 0; j < 3; j++)
{
for (let i = 0; i < 3; i++)
{
let x = w * i + w/2
let y = h * j + h/2
let spot = board[j][i]
if(spot == players[0])
{
let xoff = w/4
line(x-xoff, y-xoff, x+xoff, y+xoff)
line(x+xoff, y-xoff, x-xoff, y+xoff)
}
else if (spot == players[1])
{
circle(x, y, w/2)
}
}
}
nextTurn()
checkWinner()
}
Console
Clear
▸ TypeError: undefined is not an object (evaluating 'spot[0]')
tie



Sketch A7.10 check in all directions

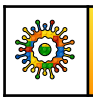
Now we need to check all the horizontals, verticals, and diagonals to see if there is a winner. We will write a function to do the checking called sameAs(), which will have three arguments and returns true if all three are true.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', '0']
let currentPlayer
let available = []

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  currentPlayer = floor(random(players.length))
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      available.push([i, j])
    }
  }
```

```
}

function sameAs(a, b, c)
{
  return (a == b && b == c && a != '')
}

function checkWinner()
{
  if (available.length == 0)
  {
    console.log('tie')
  }
}

function nextTurn()
{
  let index = floor(random(available.length))
  let spot = available.splice(index, 1)[0]
  let i = spot[0]
  let j = spot[1]
  board[i][j] = players[currentPlayer]
  currentPlayer = (currentPlayer + 1) % players.length
}

function draw()
{
  background(220)
  let w = width/3
  let h = height/3
```

```
  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == players[0])
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
  nextTurn()
  checkWinner()
}
```



Sketch A7.11 using sameAs function

Now we make use of that function in the `checkWinner()` function.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', 'O']
let currentPlayer
let available = []

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  currentPlayer = floor(random(players.length))
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      available.push([i, j])
    }
  }
}
```

```
function sameAs(a, b, c)
{
  return (a == b && b == c && a != '')
}
```

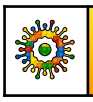
```
function checkWinner()
{
  let winner = null
 // horizontal
  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[i][0], board[i][1], board[i][2]))
    {
      winner = board[i][0]
    }
  }

 // vertical
  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[0][i], board[1][i], board[2][i]))
    {
      winner = board[0][i]
    }
  }

 // diagonals
  if (sameAs(board[0][0], board[1][1], board[2][2]))
  {
    winner = board[0][0]
```

```
  }
  if (sameAs(board[2][0], board[1][1], board[0][2]))
  {
    winner = board[2][0]
  }

  if (winner == null && available.length == 0)
  {
    return 'tie'
  }
  else
  {
    return winner
  }
}

function nextTurn()
{
  let index = floor(random(available.length))
  let spot = available.splice(index, 1)[0]
  let i = spot[0]
  let j = spot[1]
  board[i][j] = players[currentPlayer]
  currentPlayer = (currentPlayer + 1) % players.length
}

function draw()
{
  background(220)
  let w = width/3
```

```
  let h = height/3
  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[j][i]
      if(spot == players[0])
      {
        let xoff = w/4
        line(x-xoff, y-xoff, x+xoff, y+xoff)
        line(x+xoff, y-xoff, x-xoff, y+xoff)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
  nextTurn()
  checkWinner()
}
```

Notes

You will still get an error message, that will be addressed very soon.



Sketch A7.12 slow it down

Return the result and give it a `frameRate()` of `1` frame per second.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let players = ['X', 'O']
let currentPlayer
let available = []

function setup()
{
  createCanvas(400, 400)
  frameRate(1)
  noFill()
  strokeWeight(4)
  currentPlayer = floor(random(players.length))
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      available.push([i, j])
    }
  }
}
```

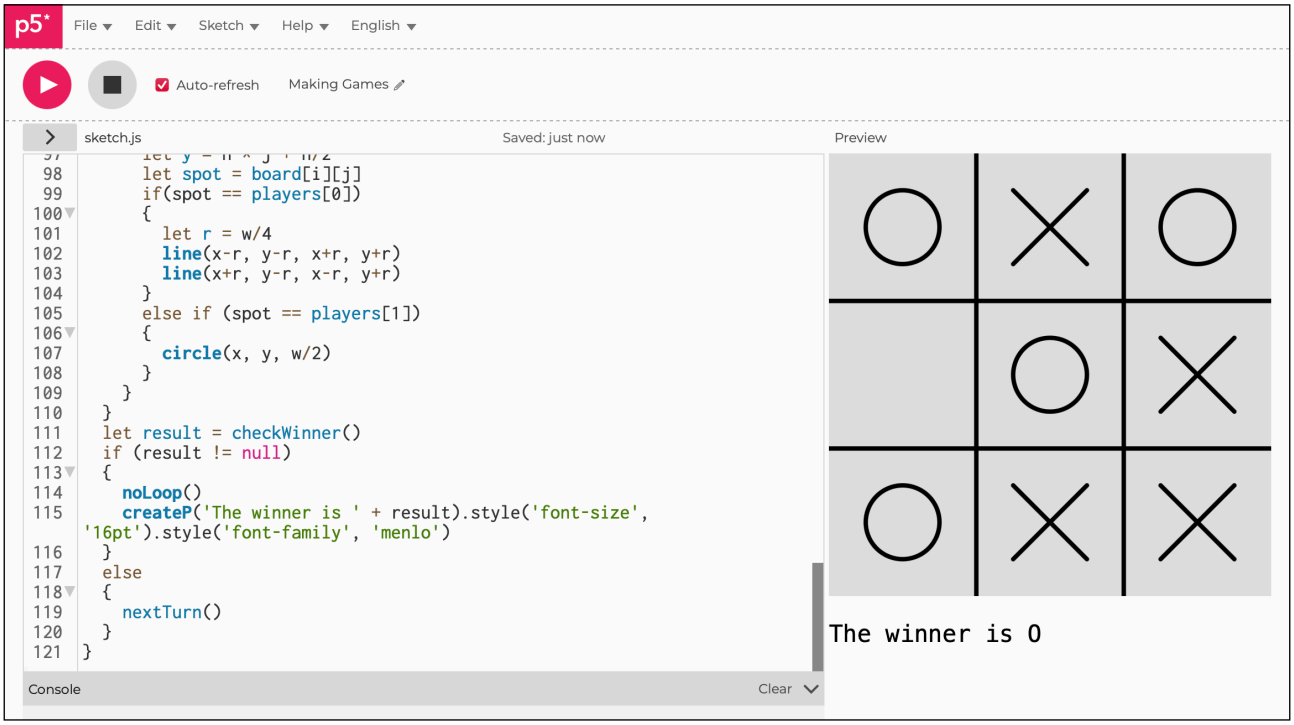
```
function sameAs(a, b, c)
{
  return (a == b && b == c && a != '')
}

function checkWinner()
{
  let winner = null
 // horizontal
  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[i][0], board[i][1], board[i][2]))
    {
      winner = board[i][0]
    }
  }

 // vertical
  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[0][i], board[1][i], board[2][i]))
    {
      winner = board[0][i]
    }
  }

 // diagonals
  if (sameAs(board[0][0], board[1][1], board[2][2]))
  {
```

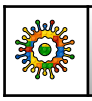
```
    winner = board[0][0]
  }
  if (sameAs(board[2][0], board[1][1], board[0][2]))
  {
    winner = board[2][0]
  }

  if (winner == null && available.length == 0)
  {
    return 'tie'
  }
  else
  {
    return winner
  }
}

function nextTurn()
{
  let index = floor(random(available.length))
  let spot = available.splice(index, 1)[0]
  let i = spot[0]
  let j = spot[1]
  board[i][j] = players[currentPlayer]
  currentPlayer = (currentPlayer + 1) % players.length
}

function draw()
{
  background(220)
```

```
  let w = width/3
  let h = height/3
  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[i][j]
      if(spot == players[0])
      {
        let r = w/4
        line(x-r, y-r, x+r, y+r)
        line(x+r, y-r, x-r, y+r)
      }
      else if (spot == players[1])
      {
        circle(x, y, w/2)
      }
    }
  }
  let result = checkWinner()
  if (result != null)
  {
    noLoop()
    createP('The winner is ' + result).style('font-size',
'16pt').style('font-family', 'menlo')
```

```
  }
  else
  {
    nextTurn()
  }
}
```

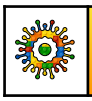
Figure A7.12


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Making Games
sketch.js
Saved: just now
Preview
let spot = board[i][j]
if(spot == players[0])
{
let r = w/4
line(x-r, y-r, x+r, y+r)
line(x+r, y-r, x-r, y+r)
}
else if (spot == players[1])
{
circle(x, y, w/2)
}
}
}
let result = checkWinner()
if (result != null)
{
noLoop()
createP('The winner is ' + result).style('font-size', '16pt').style('font-family', 'menlo')
}
else
{
nextTurn()
}
}
Console
Clear
The winner is O



Section 2: Human versus Computer

This requires a lot of refactoring to change it from the computer playing against itself to you playing it. Just work through the code and check what to leave in, change, or remove. This will form the basic code to start making a more sophisticated AI version of the game.



Sketch A7.13 playable version

This is now adapted to make it playable by you against the computer. You go first and are the circle (0); the computer makes an instant decision. There are a lot of changes which should be reasonably self-explanatory; just take the time to understand them and enjoy the game.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let w
let h
let ai = 'X'
let human = 'O'
let currentPlayer = human

function setup()
{
  createCanvas(400, 400)
  // frameRate(1)
  noFill()
  strokeWeight(4)
  w = width/3
  h = height/3
  // currentPlayer = floor(random(players.length))
  // for (let j = 0; j < 3; j++)
  // {
```

```
  //   for (let i = 0; i < 3; i++)
  //   {
  //     available.push([i, j])
  //   }
  // }
}

function sameAs(a, b, c)
{
  return (a == b && b == c && a != '')
}

function checkWinner()
{
  let winner = null
 // horizontal
  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[i][0], board[i][1], board[i][2]))
    {
      winner = board[i][0]
    }
  }

 // vertical
  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[0][i], board[1][i], board[2][i]))
    {
      winner = board[0][i]
```

```
    }
  }

// diagonals
  if (sameAs(board[0][0], board[1][1], board[2][2]))
  {
    winner = board[0][0]
  }
  if (sameAs(board[2][0], board[1][1], board[0][2]))
  {
    winner = board[2][0]
  }

  let openSpots = 0
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        openSpots++
      }
    }
  }

  if (winner == null && openSpots == 0)
  {
    return 'tie'
  }
  else
```

```
  {
    return winner
  }
}

function mousePressed()
{
  if (currentPlayer == human)
  {
    let i = floor(mouseX / w)
    let j = floor(mouseY / h)
    if (board[i][j] == '')
    {
      board[i][j] = human
      currentPlayer = ai
      nextTurn()
    }
  }
}

function nextTurn()
{
  let available = []
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        available.push({ i, j })
```

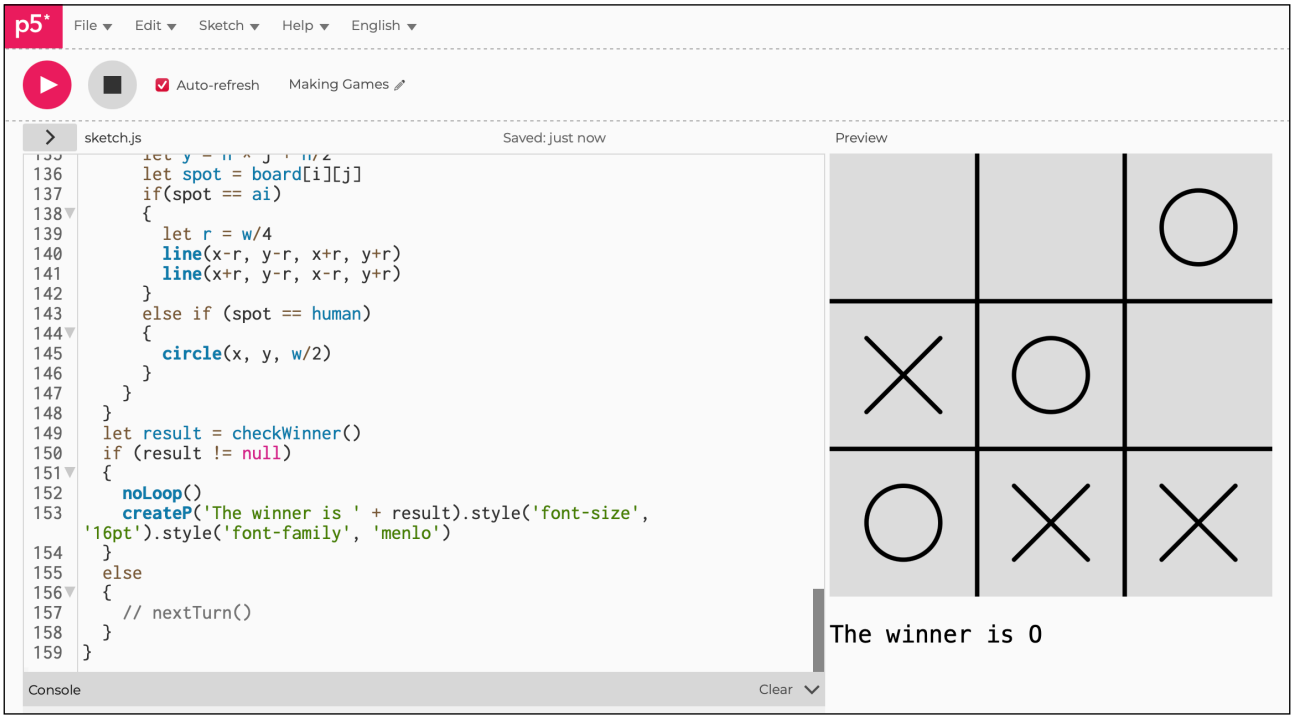
```
      }
    }
  }
  let move = random(available)
  board[move.i][move.j] = ai
  currentPlayer = human
}

function draw()
{
  background(220)

  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[i][j]
      if(spot == ai)
      {
        let r = w/4
        line(x-r, y-r, x+r, y+r)
        line(x+r, y-r, x-r, y+r)
      }
      else if (spot == human)
```

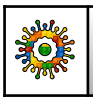
```
      {
        circle(x, y, w/2)
      }
    }
  }
  let result = checkWinner()
  if (result != null)
  {
    noLoop()
    createP('The winner is ' + result).style('font-size',
'16pt').style('font-family', 'menlo')
  }
  else
  {
    // nextTurn()
  }
}
```

🗒️ Notes

You should have a playable version, but the AI part isn't very clever; it just randomly guesses. You will get an error message when you use the full board!

Figure A7.13


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Making Games
sketch.js
Saved: just now
Preview
let spot = board[i][j]
if(spot == ai)
{
let r = w/4
line(x-r, y-r, x+r, y+r)
line(x+r, y-r, x-r, y+r)
}
else if (spot == human)
{
circle(x, y, w/2)
}
}
}
let result = checkWinner()
if (result != null)
{
noLoop()
createP('The winner is ' + result).style('font-size', '16pt').style('font-family', 'menlo')
}
else
{
// nextTurn()
}
}
Console
Clear
The winner is O

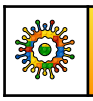


Section 3: The AI Bit

This is where the fun and the challenge start. So far, it has been unintelligent and not really a challenge to you; here, we start to give it a bit more sense of what is going on. It will be able to look more steps ahead and work out the best next move based on the possibility of winning, hence maximising its score. This is the essence of the minimax algorithm.

There is nothing deeply complicated about this, and the principle could be applied to many other games or solutions. Obviously, the more depth there is, then the longer and harder the computation becomes as the number of possibilities increases dramatically.

It is worth searching online for detailed explanations and examples that might prove rather interesting.



Sketch A7.14 clean it up

Starting with the above sketch, with redundant code, take it out.

sketch.js

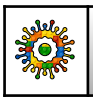
```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let w
let h
let ai = 'X'
let human = 'O'
let currentPlayer = human

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  w = width/3
  h = height/3
}

function sameAs(a, b, c)
{
  return (a == b && b == c && a != '')
}
```

```
function checkWinner()
{
  let winner = null
  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[i][0], board[i][1], board[i][2]))
    {
      winner = board[i][0]
    }
  }

  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[0][i], board[1][i], board[2][i]))
    {
      winner = board[0][i]
    }
  }

  if (sameAs(board[0][0], board[1][1], board[2][2]))
  {
    winner = board[0][0]
  }
  if (sameAs(board[2][0], board[1][1], board[0][2]))
  {
    winner = board[2][0]
  }

  let openSpots = 0
```

```
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        openSpots++
      }
    }
  }

  if (winner == null && openSpots == 0)
  {
    return 'tie'
  }
  else
  {
    return winner
  }
}

function mousePressed()
{
  if (currentPlayer == human)
  {
    let i = floor(mouseX / w)
    let j = floor(mouseY / h)
    if (board[i][j] == '')
    {
      board[i][j] = human
```

```
      currentPlayer = ai
      nextTurn()
    }
  }
}

function nextTurn()
{
  let available = []
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        available.push({ i, j })
      }
    }
  }
  let move = random(available)
  board[move.i][move.j] = ai
  currentPlayer = human
}

function draw()
{
  background(220)

  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
```

```
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[i][j]
      if(spot == ai)
      {
        let r = w/4
        line(x-r, y-r, x+r, y+r)
        line(x+r, y-r, x-r, y+r)
      }
      else if (spot == human)
      {
        circle(x, y, w/2)
      }
    }
  }
  let result = checkWinner()
  if (result != null)
  {
    noLoop()
    createP('The winner is ' + result).style('font-size',
'16pt').style('font-family', 'menlo')
  }
}
```



Introduction to minimax

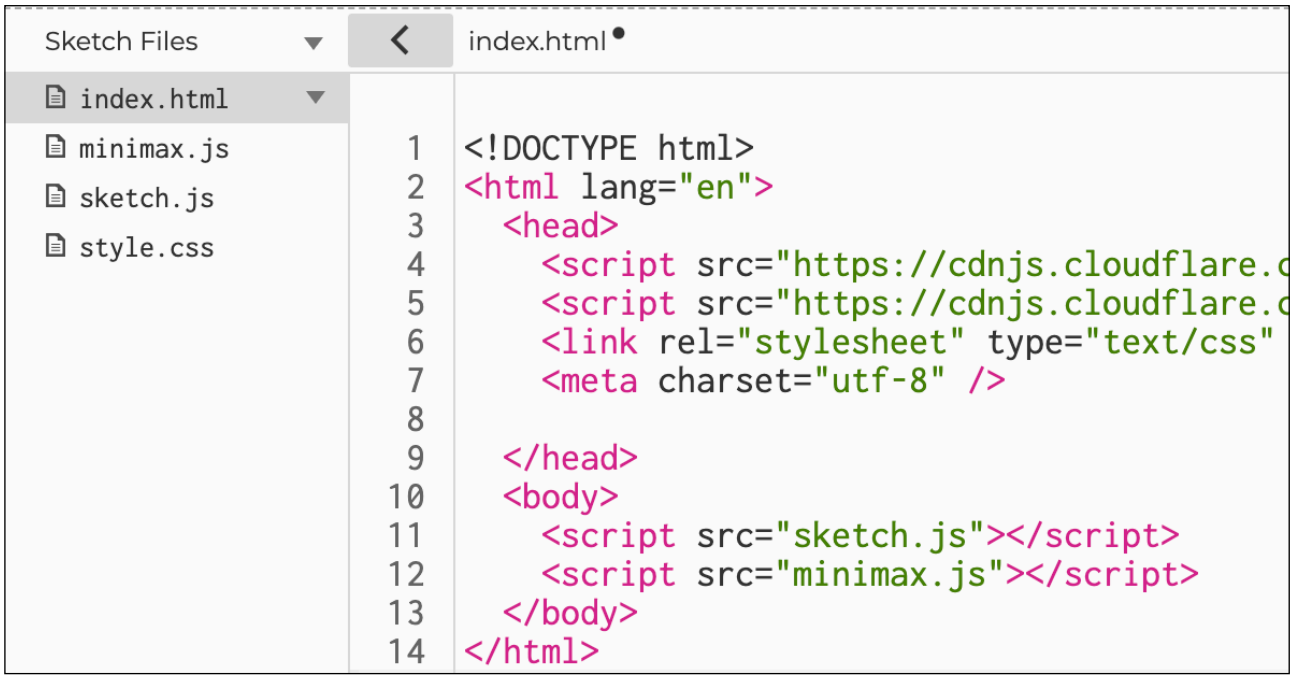
Minimax explores in a tree-like search, looking at all the possibilities. Noughts and Crosses is a good way to start with minimax as it is relatively simple. We could give it scores so that if X wins, it gets a score of +1, if O wins, a score of −1, and if it is a tie, then a score of 0.

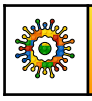
X: +1
O: -1
Tie: 0

So the minimax algorithm is going to look through all the possibilities and select the move which is most likely to give the optimal result, assuming that the other player also chooses their optimal move.

However, before we get started properly, we need to create another file and call it minimax.js. We also need to move the nextTurn() function block of code into that file. Also, reference it in the index.html file.

Figure 1: Adding the minimax file



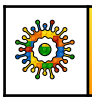


Sketch A7.15 index.html

Adding the `minimax.js` file to the index.html file.

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.11.10/p5.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.11.10/addons/p5.sound.min.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8" />

  </head>
  <body>
    <script src="sketch.js"></script>
    <script src="minimax.js"></script>
  </body>
</html>
```



Sketch A7.16 next turn function

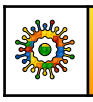
We put the `nextTurn()` function into the `minimax.js` file.

minimax.js

```
function nextTurn()
{
  let available = []
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        available.push({ i, j })
      }
    }
  }
  let move = random(available)
  board[move.i][move.j] = ai
  currentPlayer = human
}
```

Notes

Just check it all works by clicking on the grid as before.

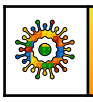


Sketch A7.17 best move rename

We will rename `nextTurn()` and call it `bestMove()`. Remove the `available` array.

minimax.js

```
function bestMove()
{
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board)
      }
    }
  }
  let move = random(available)
  board[move.i][move.j] = ai
  currentPlayer = human
}
```

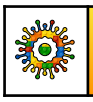


Sketch A7.18 to infinity and beyond

We need to keep track of the score as it works through the possible options. -Infinity makes the starting initial value definitely very low, very low indeed; you can't get any smaller!

minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board)
        if (score > bestScore)
        {
          bestScore = score
        }
      }
    }
  }
  let move = random(available)
  board[move.i][move.j] = ai
  currentPlayer = human
}
```

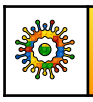


Sketch A7.19 best move, best score

So we want the bestMove to be based on the bestScore. We remove random move.

minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  let bestMove
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board)
        if (score > bestScore)
        {
          bestScore = score
          bestMove = { i, j }
        }
      }
    }
  }
  board[move.i][move.j] = ai
  currentPlayer = human
}
```

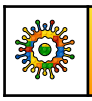


Sketch A7.20 clear the board

We undo the board.

minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  let bestMove
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board)
        board[i][j] = ''
        if (score > bestScore)
        {
          bestScore = score
          bestMove = { i, j }
        }
      }
    }
  }
  board[bestMove.i][bestMove.j] = ai
  currentPlayer = human
}
```



Sketch A7.21 a tidy up

❗ sketch.js

Replace `nextTurn()` with `bestMove()` in `sketch.js`.

sketch.js

```
let board = [
  ['', '', ''],
  ['', '', ''],
  ['', '', '']
]

let w
let h
let ai = 'X'
let human = 'O'
let currentPlayer = human

function setup()
{
  createCanvas(400, 400)
  noFill()
  strokeWeight(4)
  w = width/3
  h = height/3
  bestMove()
}

function sameAs(a, b, c)
{
```

```
  return (a == b && b == c && a != '')
}

function checkWinner()
{
  let winner = null
  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[i][0], board[i][1], board[i][2]))
    {
      winner = board[i][0]
    }
  }

  for (let i = 0; i < 3; i++)
  {
    if (sameAs(board[0][i], board[1][i], board[2][i]))
    {
      winner = board[0][i]
    }
  }

  if (sameAs(board[0][0], board[1][1], board[2][2]))
  {
    winner = board[0][0]
  }
  if (sameAs(board[2][0], board[1][1], board[0][2]))
  {
    winner = board[2][0]
  }
```

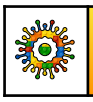
```
  let openSpots = 0
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        openSpots++
      }
    }
  }

  if (winner == null && openSpots == 0)
  {
    return 'tie'
  }
  else
  {
    return winner
  }
}

function mousePressed()
{
  if (currentPlayer == human)
  {
    let i = floor(mouseX / w)
    let j = floor(mouseY / h)
    if (board[i][j] == '')
```

```
    {
      board[i][j] = human
      currentPlayer = ai
      bestMove()
    }
  }
}

function draw()
{
  background(220)

  line(w, 0, w, height)
  line(w*2, 0, w*2, height)
  line(0, h, width, h)
  line(0, h*2, width, h*2)
  for (let j = 0; j < 3; j++)
  {
    for (let i = 0; i < 3; i++)
    {
      let x = w * i + w/2
      let y = h * j + h/2
      let spot = board[i][j]
      if(spot == ai)
      {
        let r = w/4
        line(x-r, y-r, x+r, y+r)
        line(x+r, y-r, x-r, y+r)
      }
      else if (spot == human)
```

```
      {
        circle(x, y, w/2)
      }
    }
  }
  let result = checkWinner()
  if (result != null)
  {
    noLoop()
    createP('The winner is ' + result).style('font-size',
'16pt').style('font-family', 'menlo')
  }
}
```

🗒️ Notes

You will get error messages!



Sketch A7.22 depth

We give the minimax() function a depth argument (set to 0) and create the function, returning 1 just to check everything still works.

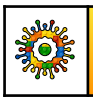
minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  let bestMove
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board, 0)
        board[i][j] = ''
        if (score > bestScore)
        {
          bestScore = score
          bestMove = { i, j }
        }
      }
    }
  }
  board[bestMove.i][bestMove.j] = ai
  currentPlayer = human
}
```

```
function minimax(board, depth)
{
  return 1
}
```

Notes

Still not that intelligent but it should be working at least.



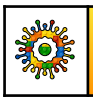
Sketch A7.23 minimise the human

Because it is a turn-based game, we need to add another argument when it is the human's turn: `isMaximising`, and set it to `false`.

minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  let bestMove
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board, 0, false)
        board[i][j] = ''
        if (score > bestScore)
        {
          bestScore = score
          bestMove = { i, j }
        }
      }
    }
  }
  board[bestMove.i][bestMove.j] = ai
  currentPlayer = human
}
```

```
function minimax(board, depth, isMaximising)
{
  if (isMaximising)
  {


  }
  return 1
}
```



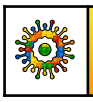
Sketch A7.24 a look up table

We need to check if there is a winner and track the score with a lookup table based on our scoring. This is a sort of reward system. Allowing the programme to calculate the best next move based on the possible outcomes (scores).

minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  let bestMove
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board, 0, false)
        board[i][j] = ''
        if (score > bestScore)
        {
          bestScore = score
          bestMove = { i, j }
        }
      }
    }
  }
  board[bestMove.i][bestMove.j] = ai
  currentPlayer = human
```

```
}

let scores = {X: 1, O: -1, tie: 0}

function minimax(board, depth, isMaximising)
{
  let result = checkWinner()
  if(result !== null)
  {
    let score = scores[result]
    return score
  }
  if (isMaximising)
  {

  }
  return 1
}
```



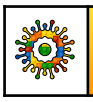
Sketch A7.25 maximising the AI

Need to work out the best score to maximise the best move.

minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  let bestMove
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board, 0, false)
        board[i][j] = ''
        if (score > bestScore)
        {
          bestScore = score
          bestMove = { i, j }
        }
      }
    }
  }
  board[bestMove.i][bestMove.j] = ai
  currentPlayer = human
}
```

```
let scores = {X: 1, O: -1, tie: 0}

function minimax(board, depth, isMaximising)
{
  let result = checkWinner()
  if(result !== null)
  {
    let score = scores[result]
    return score
  }
  if (isMaximising)
  {
    let bestScore = -Infinity
    for (let i = 0; i < 3; i++)
    {
      for (let j = 0; j < 3; j++)
      {
        if (board[i][j] == '')
        {
          board[i][j] = ai
          let score = minimax(board, depth + 1, false)
          board[i][j] = ''
        }
      }
    }
  }
  return 1
}
```



Sketch A7.26 the best score

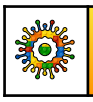
Now, to return the best score.

minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  let bestMove
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board, 0, false)
        board[i][j] = ''
        if (score > bestScore)
        {
          bestScore = score
          bestMove = { i, j }
        }
      }
    }
  }
  board[bestMove.i][bestMove.j] = ai
  currentPlayer = human
}
```

```
let scores = {X: 1, O: -1, tie: 0}

function minimax(board, depth, isMaximising)
{
  let result = checkWinner()
  if(result !== null)
  {
    let score = scores[result]
    return score
  }
  if (isMaximising)
  {
    let bestScore = -Infinity
    for (let i = 0; i < 3; i++)
    {
      for (let j = 0; j < 3; j++)
      {
        if (board[i][j] == '')
        {
          board[i][j] = ai
          let score = minimax(board, depth + 1, false)
          board[i][j] = ''
          if (score > bestScore)
          {
            bestScore = score
          }
        }
      }
    }
    return bestScore
```

```
  }
  return 1
}
```



Sketch A7.27 puny human!

Now add in the minimising player (human), which is almost a mirror of the ai maximise.

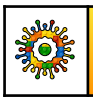
❗ Remove `return 1`.

minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  let bestMove
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board, 0, false)
        board[i][j] = ''
        if (score > bestScore)
        {
          bestScore = score
          bestMove = { i, j }
        }
      }
    }
  }
  board[bestMove.i][bestMove.j] = ai
  currentPlayer = human
```

```
}

let scores = {X: 1, 0: -1, tie: 0}

function minimax(board, depth, isMaximising)
{
  let result = checkWinner()
  if(result !== null)
  {
    let score = scores[result]
    return score
  }
  if (isMaximising)
  {
    let bestScore = -Infinity
    for (let i = 0; i < 3; i++)
    {
      for (let j = 0; j < 3; j++)
      {
        if (board[i][j] == '')
        {
          board[i][j] = ai
          let score = minimax(board, depth + 1, false)
          board[i][j] = ''
          if (score > bestScore)
          {
            bestScore = score
          }
        }
      }
```

```
    }
    return bestScore
  }
  else
  {
    let bestScore = Infinity
    for (let i = 0; i < 3; i++)
    {
      for (let j = 0; j < 3; j++)
      {
        if (board[i][j] == '')
        {
          board[i][j] = human
          let score = minimax(board, depth + 1, true)
          board[i][j] = ''
          if (score > bestScore)
          {
            bestScore = score
          }
        }
      }
    }
    return bestScore
  }
}
```



Sketch A7.28 min and max

We can use the `min()` and `max()` functions, and also refactor the return `scores[result]`.

minimax.js

```
function bestMove()
{
  let bestScore = -Infinity
  let bestMove
  for (let i = 0; i < 3; i++)
  {
    for (let j = 0; j < 3; j++)
    {
      if (board[i][j] == '')
      {
        board[i][j] = ai
        let score = minimax(board, 0, false)
        board[i][j] = ''
        if (score > bestScore)
        {
          bestScore = score
          bestMove = { i, j }
        }
      }
    }
  }
  board[bestMove.i][bestMove.j] = ai
  currentPlayer = human
}
```

```
let scores = {X: 1, O: -1, tie: 0}

function minimax(board, depth, isMaximising)
{
  let result = checkWinner()
  if(result !== null)
  {
    return scores[result]
  }
  if (isMaximising)
  {
    let bestScore = -Infinity
    for (let i = 0; i < 3; i++)
    {
      for (let j = 0; j < 3; j++)
      {
        if (board[i][j] == '')
        {
          board[i][j] = ai
          let score = minimax(board, depth + 1, false)
          board[i][j] = ''
          bestScore = max(score, bestScore)
        }
      }
    }
    return bestScore
  }
  else
  {
```

```
    let bestScore = Infinity
    for (let i = 0; i < 3; i++)
    {
      for (let j = 0; j < 3; j++)
      {
        if (board[i][j] == '')
        {
          board[i][j] = human
          let score = minimax(board, depth + 1, true)
          board[i][j] = ''
          bestScore = min(score, bestScore)
        }
      }
    }
    return bestScore
  }
}
```

🗒️ Notes

It is now playing to win rather trying not to lose, which was a defensive stance. If you are playing to your best then it will always be a tie. If you make the grid larger then there are more options.

🌻 Challenges

1. Have a bigger board e.g. 5x5
2. Create a 3D version