i<tag>
INTERACTIVE TECHNOLOGIES AND GAMES CONFERENCE

Playing games with observation, dependency and agency in a new environment for making construals

Meurig Beynon Jonathan Foss

and the CONSTRUIT! project team

CONSTRUIT!

Making construals
as a new digital skill
for creating interactive
open educational resources

construit.org
jseden.dcs.warwick.ac.uk/construit.c3

"The environment for making construals" – aka as "the MCE"

Erasmus+

This project has been funded with support from the European Commission under the Erasmus+ programme (2014-1-UK01-KA200-001818). This presentation reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Construing the title ...

*Playing **games** ...*

Noughts-&-Crosses (N&C)! → OXO-like games

*... with **observation**, **dependency** and **agency***

key concepts in “making construals”

*... in **a new environment for making construals***

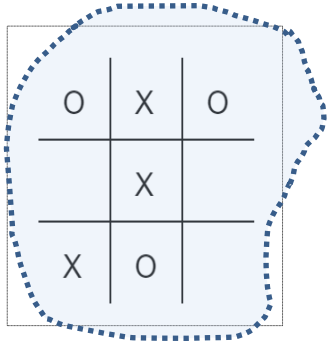
`jseden.dcs.warwick.ac.uk/construit.c3`

Making a construal of N&C

Imagine you are watching two people play N&C

Explain what you think is going on ...

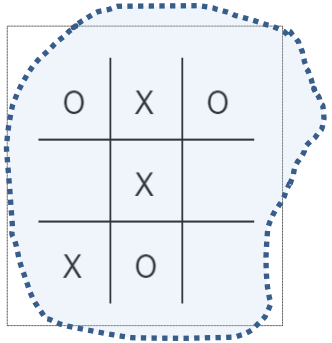
- there's a grid, some squares
- players take turns to place Os and Xs
- if O/X makes a line of Os/Xs, they win
- if neither player makes a line, it's a draw



Making a construal of ... ?

Imagine two people playing *what looks like* N&C

Explain what you think is going on ...

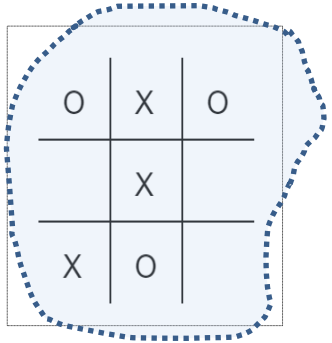


- when an O disappears from the grid
- when X makes two successive moves
- when X wins in the position pictured above
- when the board rotates after each move

Some possible construals

Perhaps when we see the two ‘N&C players’ ...

- they are designing a board for N&C
- a piece falls of the board in play
- a player cheats
- the winning lines are different
- the playing protocol has been changed

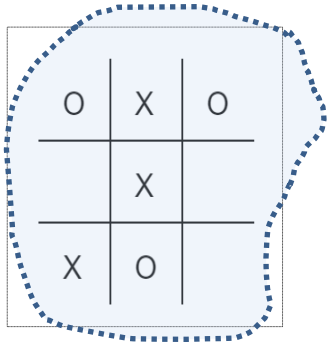


Making a construal … ?

What does it mean to make a construal?

We link what we think is going on with:

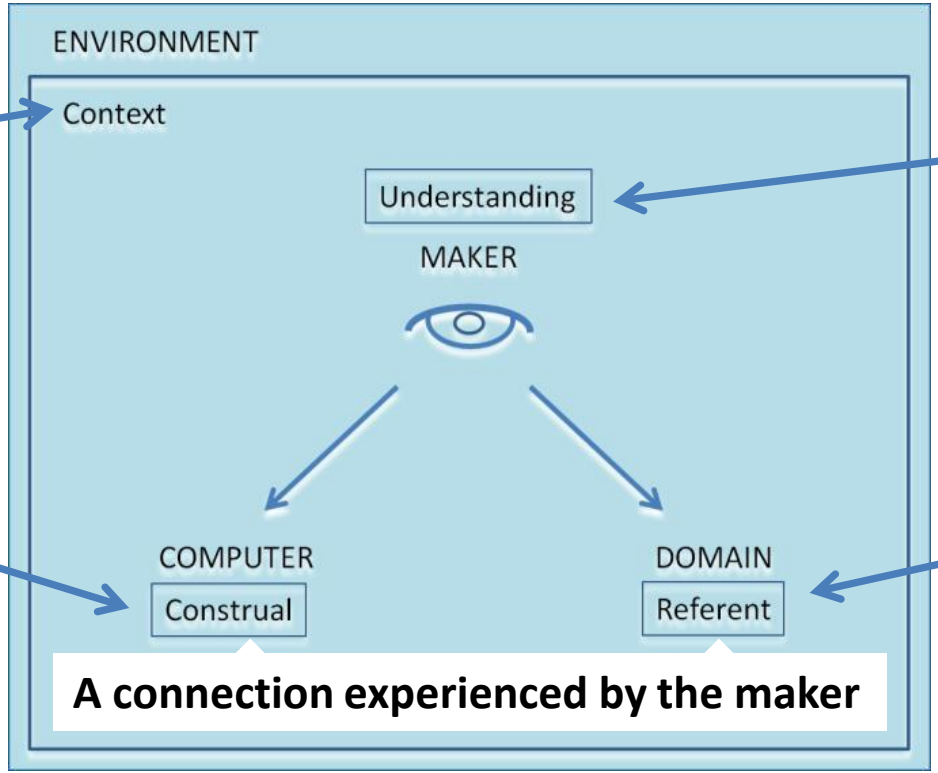
- what/who is *responsible for changes*
- what we/they must be *observing*
- what *connects the changes* we/they observe



These are **agency** / **observables** / **dependencies**

Making a digital construal

From which perspective is the maker making the construal e.g. Agents? Constraints?



What interactions and interpretations is the maker familiar with? Convinced of? puzzled about?

Script of definitions of observables with associated **network** of dependencies

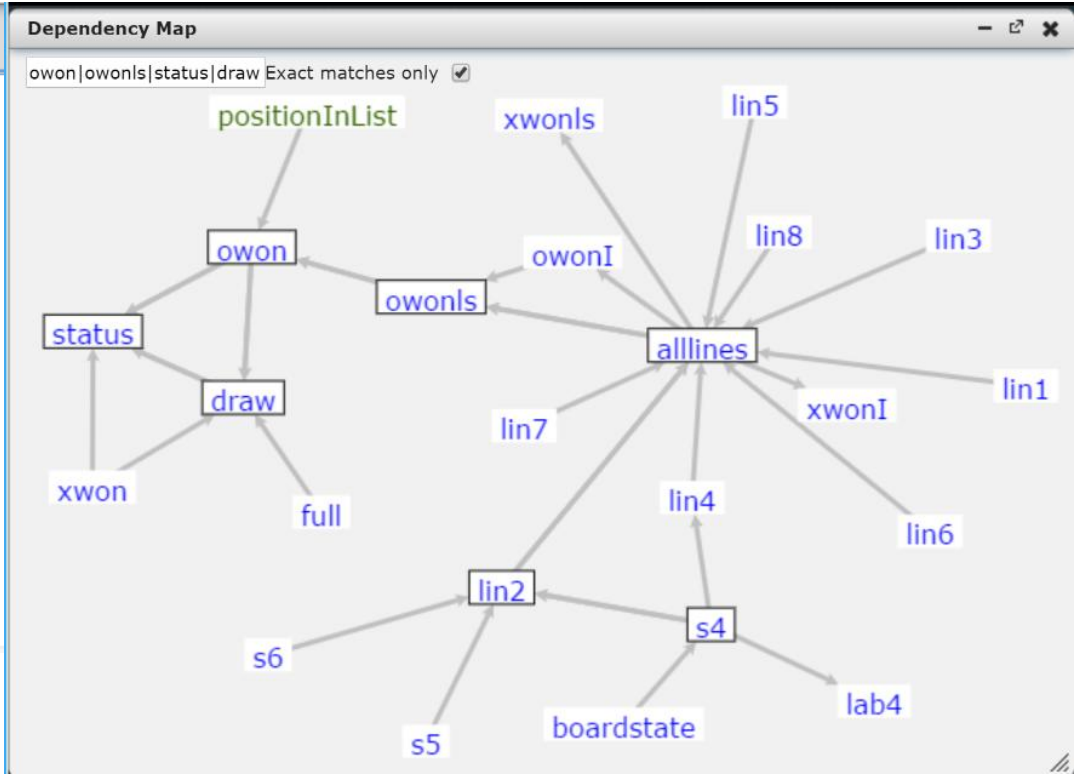
What external subject does the maker have in mind when interacting with the construal?

Dependencies in N&C

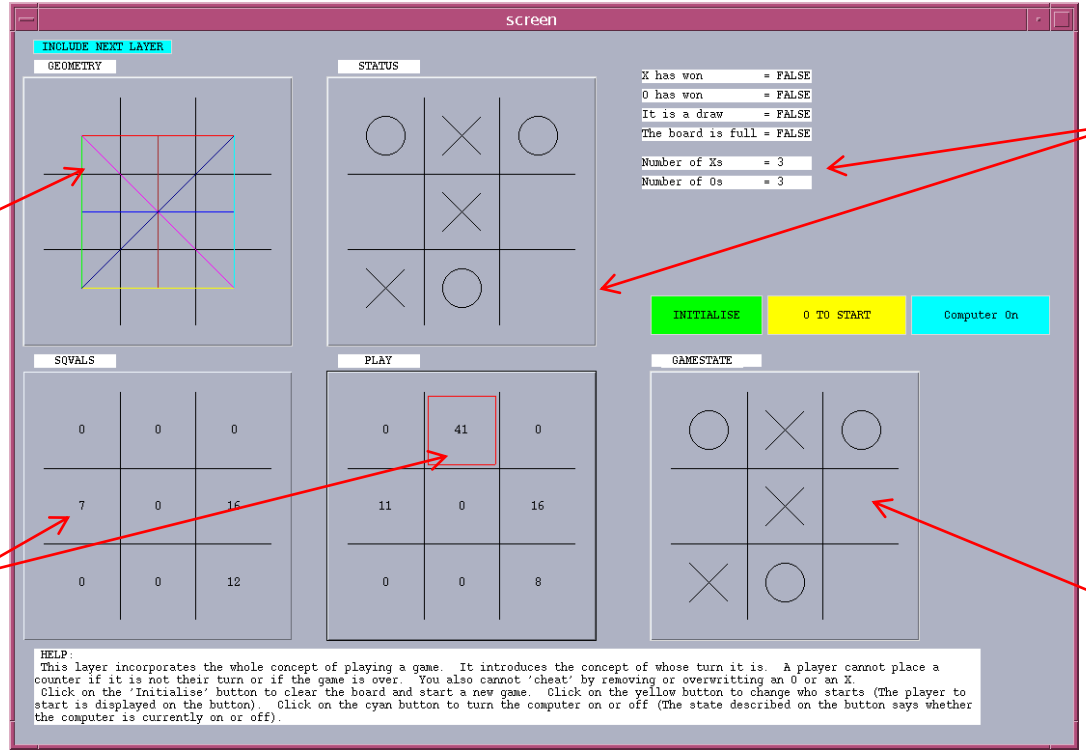
...observableList2

```
s4 is boardstate[4];
lin2 is [s4,s5,s6];
alllines is [lin1,lin2,lin3,lin4,
              lin5,lin6,lin7,lin8];
owonI is (alllines[_i][1] == o) &&
          (alllines[_i][2] == o) &&
              (alllines[_i][3] == o);
owonls is owonI with _i is 1..alllines#;
owon is positionInList(true, owonls) != 0;
draw is (! xwon) && (! owon) && full;
status is (xwon?"X wins ":"") //
          (owon?"O wins ":"") //
          (draw?"Draw ":"") // "";
```

Script of observable definitions and **network** of dependencies



The OXO Laboratory



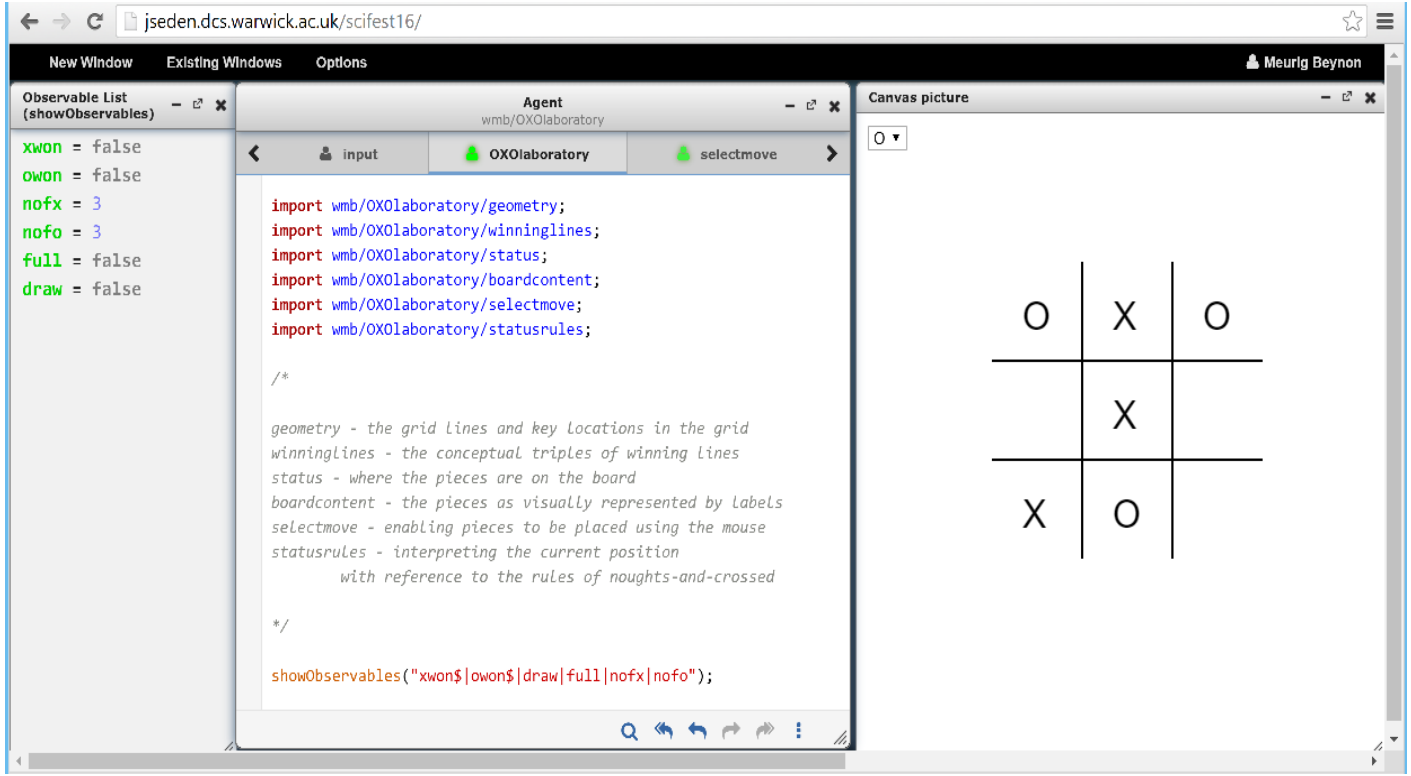
Winning lines

Square evaluation and chosen move

Status of an arbitrary OXO position

State of OXO board in play

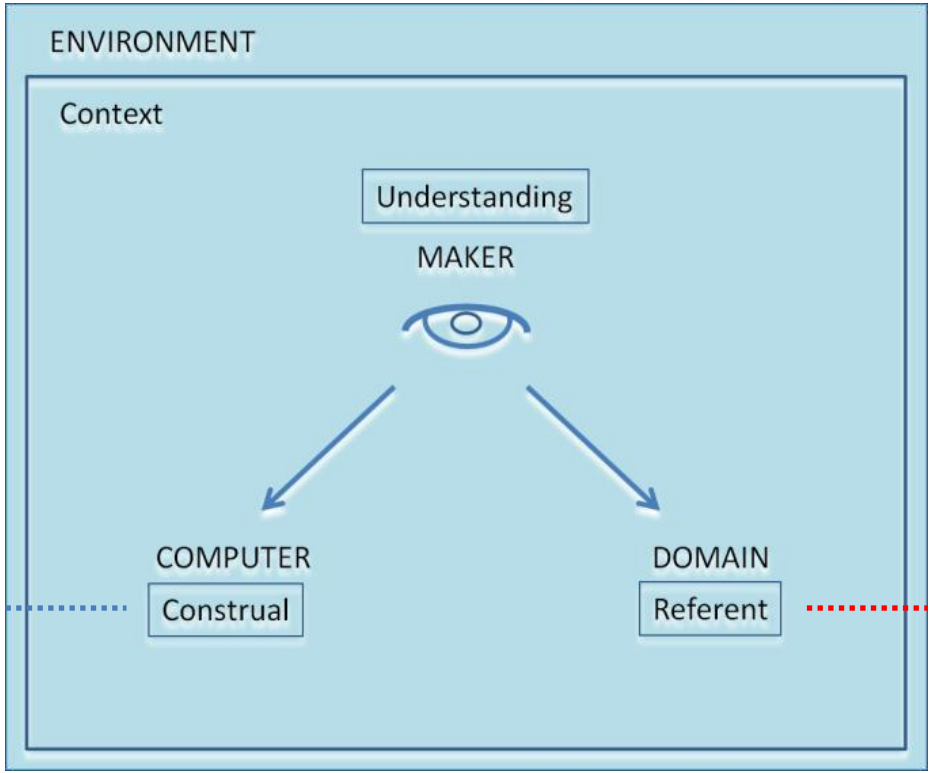
The OXO Lab in the MCE

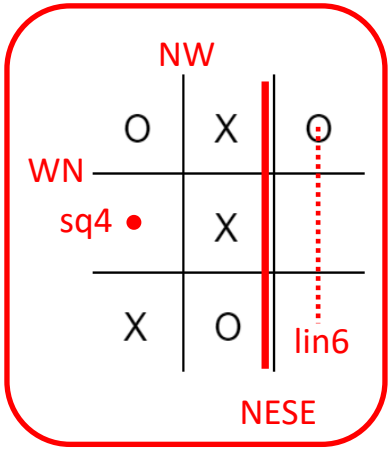


Observing the geometry

Geometry
- lines of the grid (NESE)
- endpoints of lines (NW)
- locations for tokens (sq4)
- size of the grid (size)

Winning lines
- eight winning triples (lin6)



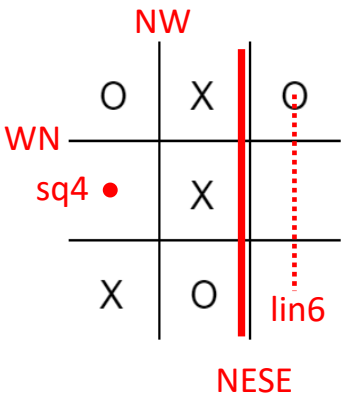


Modelling the OXO grid

Agent
wmb/OXOlaboratory/geometry

statusrules | geometry

```
WN is Point(100+size*0.5, 100+size*4.5);
NW is Point(100+size*2.5, 100+size*6.5);

-----------------------------------------

sq1 is Point(100+size*1.5, 100+size*5.5);
sq2 is Point(100+size*3.5, 100+size*5.5);
sq3 is Point(100+size*5.5, 100+size*5.5);
sq4 is Point(100+size*1.5, 100+size*3.5);

-----------------------------------------

NWSW is Line(NW["x"],NW["y"],SW["x"],SW["y"]);
NESE is Line(NE["x"],NE["y"],SE["x"],SE["y"]);

gridlines is [WNEN, WSES, NWSW, NESE];
picture is gridlines;

size = 40;
```



Agent
wmb/OXOlaboratory/winninglines

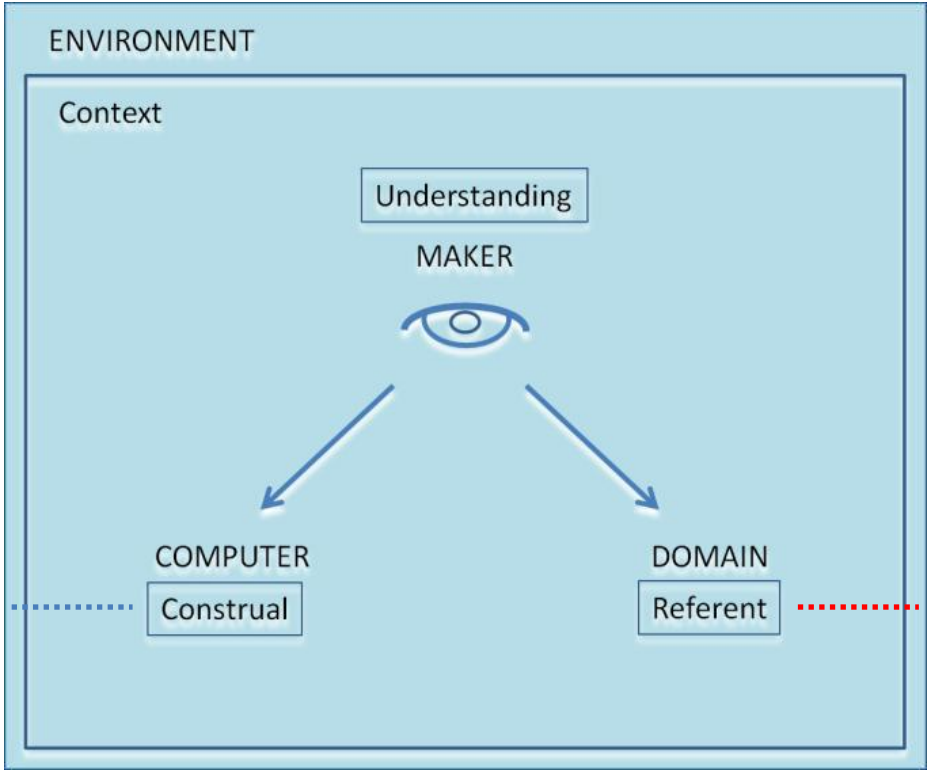
statusrules | geometry | winninglines

```
allsquares is [s1,s2,s3,s4,s5,s6,s7,s8,s9];
nofsquares is allsquares#;

lin1 is [s1,s2,s3];
lin2 is [s4,s5,s6];
lin3 is [s7,s8,s9];
lin4 is [s1,s4,s7];
lin5 is [s2,s5,s8];
lin6 is [s3,s6,s9];
lin7 is [s1,s5,s9];
lin8 is [s3,s5,s7];

alllines is [lin1,lin2,lin3,lin4,lin5,lin6,lin7,lin8];
```

Observing the board state

Status
- current position (boardstate)
- tokens on squares (x / o / u)

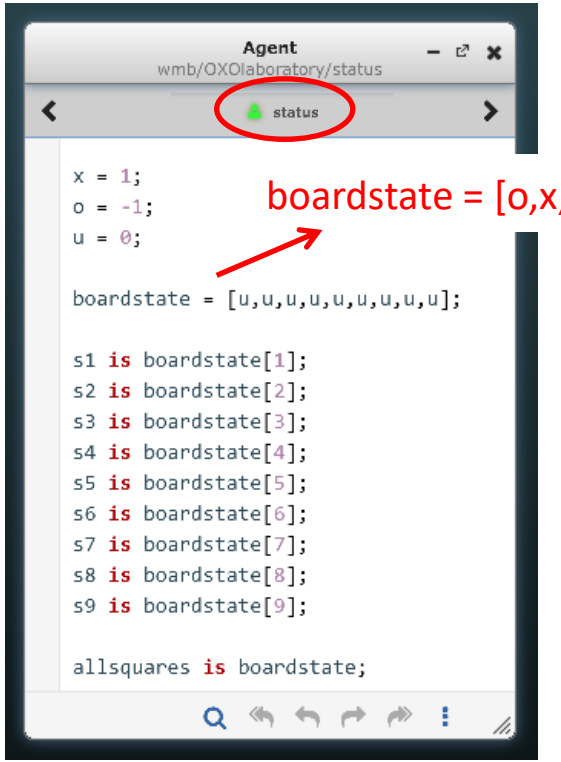
Board content
- visual token (lab5)
- located visual token (piece5)



boardstate :
[o,x,o,u,x,u,x,o,u]
x / o / u: 1 / -1 / 0
lab5: "X"
piece5: "X" at sq5

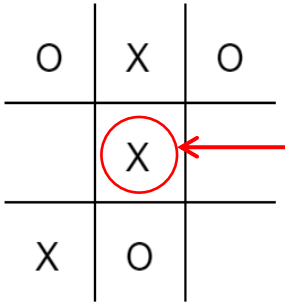
O	X	O
	X	
X	O	

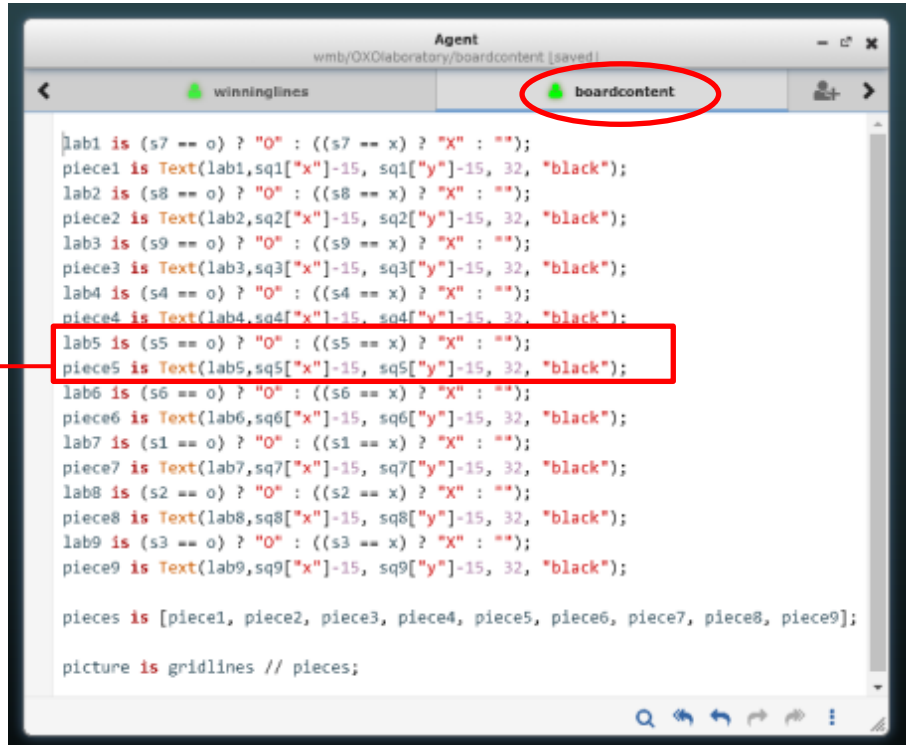
The state of the OXO grid



s5

boardstate = [o,x,o,u,x,u,x,o,u];

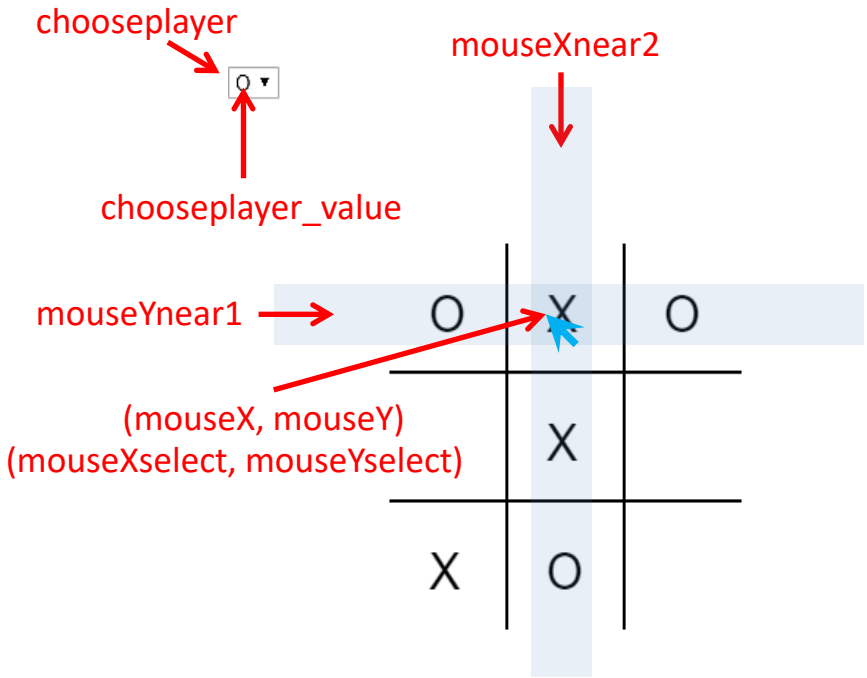




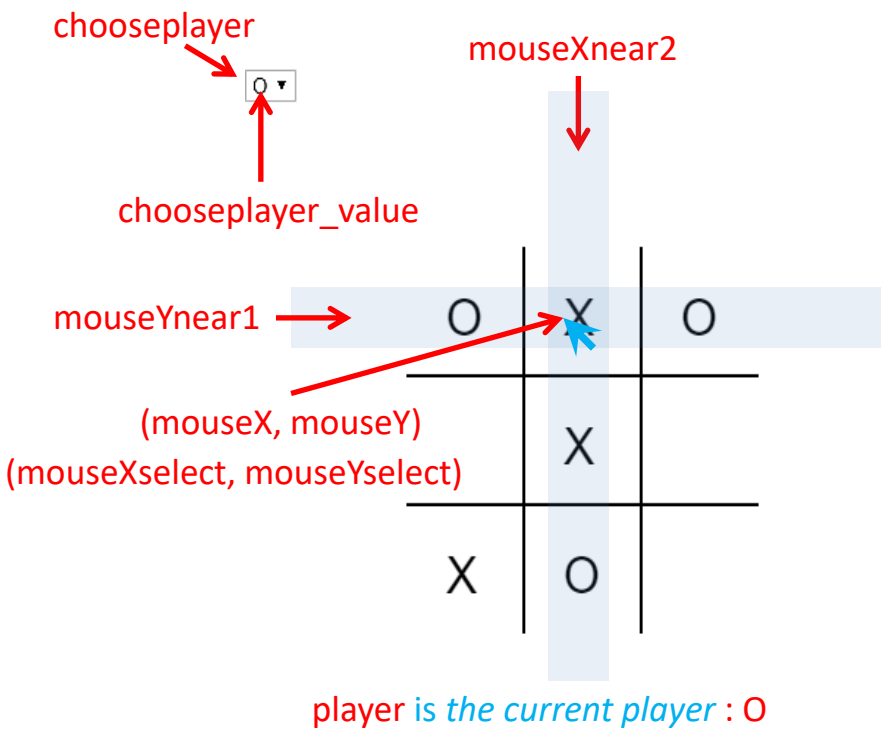
Modelling players' agency

Select move

- whose turn is it?
  (chooseplayer / chooseplayer_value / player)
- where on board to play?
  (mouseXnear2 / mouseYnear1)
- which square?
  (mouseXselect / mouseYselect)



Placing Os and Xs on grid

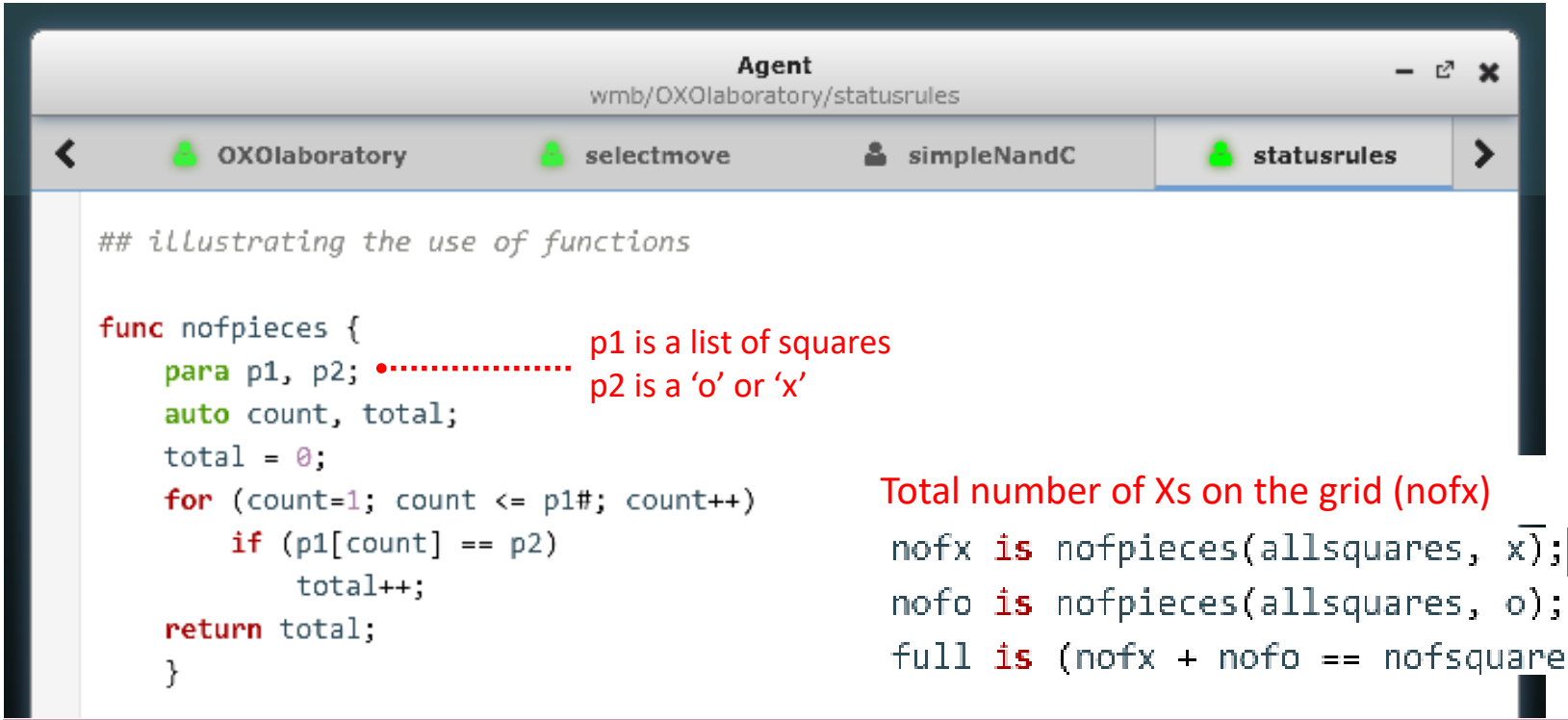


Agent
wmb/OXOlaboratory/selectmove

OXOlaboratory | selectmove | simpleNandC | statusrules

```
near = 50;

mouseXnear1 is ((mouseX-160)*(mouseX-160)<near);
mouseXnear2 is ((mouseX-240)*(mouseX-240)<near);
mouseXnear3 is ((mouseX-320)*(mouseX-320)<near);

mouseYnear1 is ((mouseY-160)*(mouseY-160)<near);
mouseYnear2 is ((mouseY-240)*(mouseY-240)<near);
mouseYnear3 is ((mouseY-320)*(mouseY-320)<near);

mouseXselect is (mouseXnear1 ? 1 : (mouseXnear2 ? 2 : (mouseXnear3 ? 3 : 0)));
mouseYselect is (mouseYnear1 ? 1 : (mouseYnear2 ? 2 : (mouseYnear3 ? 3 : 0)));

proc makemove: mousePressed {
    if ((mousePressed) && (mouseXselect*mouseYselect != 0))
        boardstate[(mouseYselect-1)*3 + mouseXselect] = player;
}

chooseplayer is DropDownList(["O","X"],["O","X"], 10, 10, true);
player is (chooseplayer_value == "O") ? o : ((chooseplayer_value == "X") ? x : "");
picture is gridlines // pieces // [chooseplayer];
```

Maker-defined functions



Using the **with**-construct

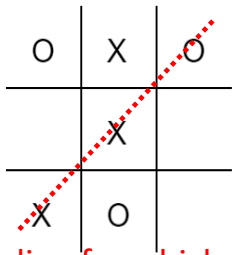
Agent
wmb/OXOlaboratory/statusrules

OXOlaboratory | selectmove | simpleNandC | statusrules

```
## illustrating the with construct

xwonI is (alllines[_i][1] == x) && (alllines[_i][2] == x) && (alllines[_i][3] == x);
xwonls is xwonI with _i is 1..alllines#;

owonI is (alllines[_i][1] == o) && (alllines[_i][2] == o) && (alllines[_i][3] == o);
owonls is owonI with _i is 1..alllines#;

xwon is positionInList(true, xwonls) != 0;
owon is positionInList(true, owonls) != 0;
```

"X has a winning line"

"The line indexed by _i is a winning line for X"



"The line for which _i is 8"

"Has X or O won the game? == "Is there a winning line for X or O?"

Interpreting the rules

Agent
wmb/OXOlaboratory/statusrules

OXOlaboratory | selectmove | simpleNandC | statusrules

```
## illustrating the with construct

xwonI is (alllines[_i][1] == x) && (alllines[_i][2] == x) && (alllines[_i][3] == x);
xwonls is xwonI with _i is 1..alllines#;

owonI is (alllines[_i][1] == o) && (alllines[_i][2] == o) && (alllines[_i][3] == o);
owonls is owonI with _i is 1..alllines#;

xwon is positionInList(true, xwonls) != 0;
owon is positionInList(true, owonls) != 0;
```

"X has a winning line"

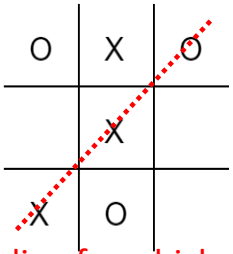
"The _i-th line is a winning line for X"

Agent
wmb/OXOlaboratory/statusrules

OXOlaboratory | selectmove | simpleNandC | statusrules

```
## illustrating the use of functions

func nofpieces {
    para p1, p2;
    auto count, total;
    total = 0;
    for (count=1; count <= p1#; count++)
        if (p1[count] == p2)
            total++;
    return total;
    }
```

p1 is a list of squares
p2 is a 'o' or 'x'



"The line for which _i is 8"

```
nofsquares is allsquares#;
nofx is nofpieces(allsquares, x);
nofo is nofpieces(allsquares, o);
full is (nofx + nofo == nofsquares);
draw is (! xwon) && (! owon) && full;
status is (xwon?"X wins ":"") // (owon?"O wins ":"") // (draw?"Draw ":"") // "";
```

Total number of Xs on the grid

¡ O seeks a winning move

- *Is the game over?*

```
end_of_game is owon || xwon || draw;
```

- *Does line **lin** have just 2 Os on it?*

```
lin_w is lin[1]+lin[2]+lin[3] == -2;
```

- *Does the line with index **ix** have just 2 Os on it?*

```
winlineix is lin_w with lin is alllines[ix];
```

- *Does any line have just 2 Os on it?*

```
winlines is winlineix with ix is 1..8;
```

- *Where is there an empty square on line lin?*

```
gapinlin is 1 if lin[1]==0 else (2 if lin[2]==0 else (3 if lin[3]==0 else 0));
```

- *Where the line with index **ix** has a space ... where each line has a space*

```
playonlinix is gapinlin with lin is alllines[ix];
playonlines is playonlinix with ix is 1..8;
```

¡O makes a winning move

Register the winning lines as sets of indices of cells on the grid, not via their contents

```
alllinesindices is alllines
   with s1 is 1, s2 is 2, s3 is 3, s4 is 4, s5 is 5, s6 is 6, s7 is 7, s8 is 8, s9 is 9;
```

- *The index of the line with index _index if it is a winning line for O*

```
winindex is _index if winlines[_index] else 0;
```

- *The set of indices of line that are winning lines for O – possibly empty*

```
iswinindex is winindex with _index is 1..8;
```

- *An index of a winning line for O, if there is one*

```
wline is max(iswinindex);
```

O is to play *there is a winning line* *the game isn't over*

```
when ((player==o) &&      wline>0        &&   !end_of_game      ) {
    boardstate[     alllinesindices [wline]      [playonlines[wline]]              ] = o;
}                    index of winning line   index of gap on winning line
```

Reflections on the MCE

- Tension between ...
  - bricolage vs. eliminate redundancy, conform to standards, clarify through abstraction
  - cf. functions and procedures 'objective' with's and when's 'agent-oriented'
- Challenges
  - ? Project Manager interface
  - ? intuitive way to represent with's

Acknowledgments

Nick Pope, Elizabeth Hudnott, Joe Butler, Tim Monks: The JS-Eden environment

Simon Gardner: The original OXO laboratory (1999)

Mike Joy: The first OXO construal prototype (1994)

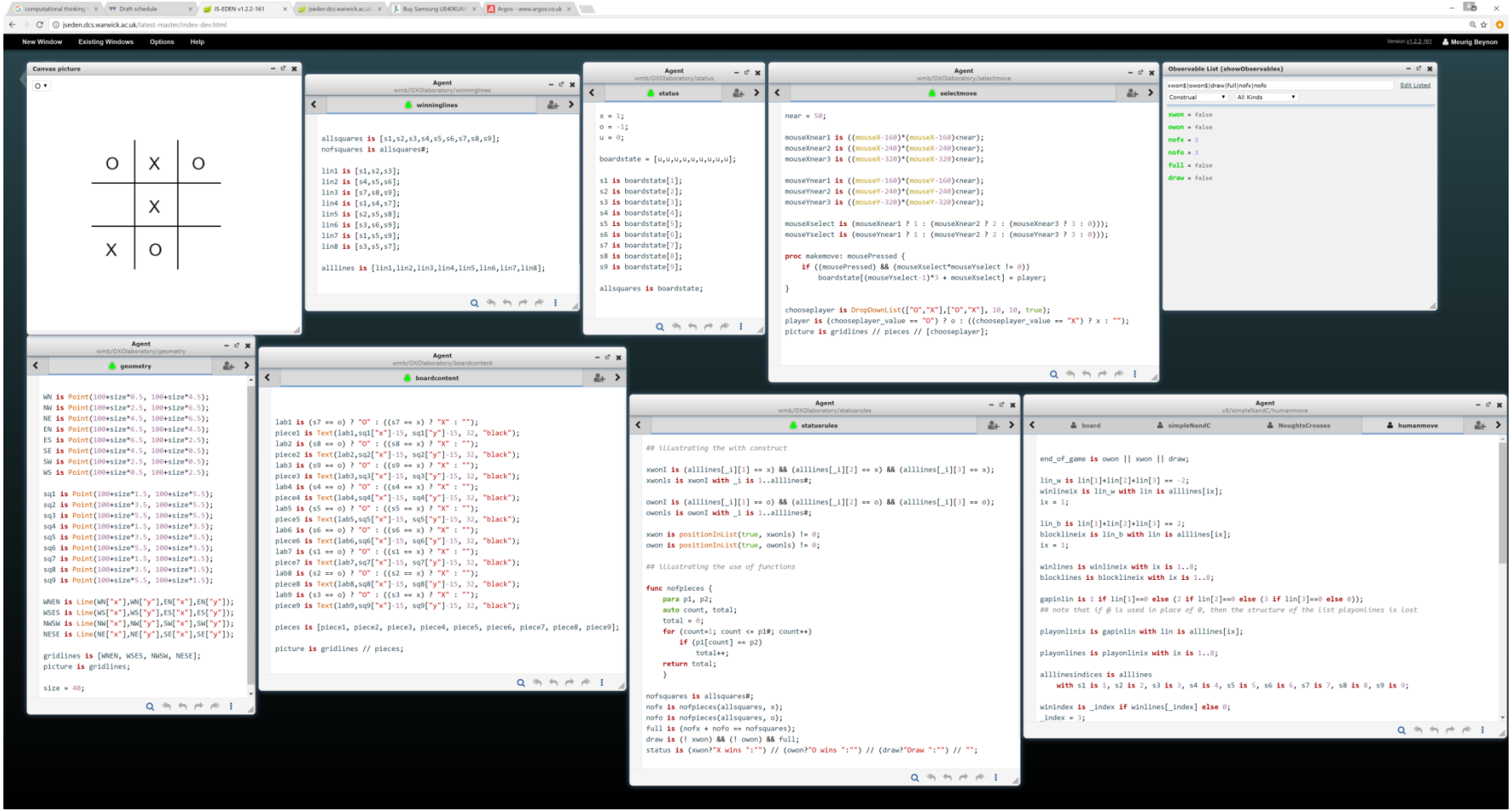
References

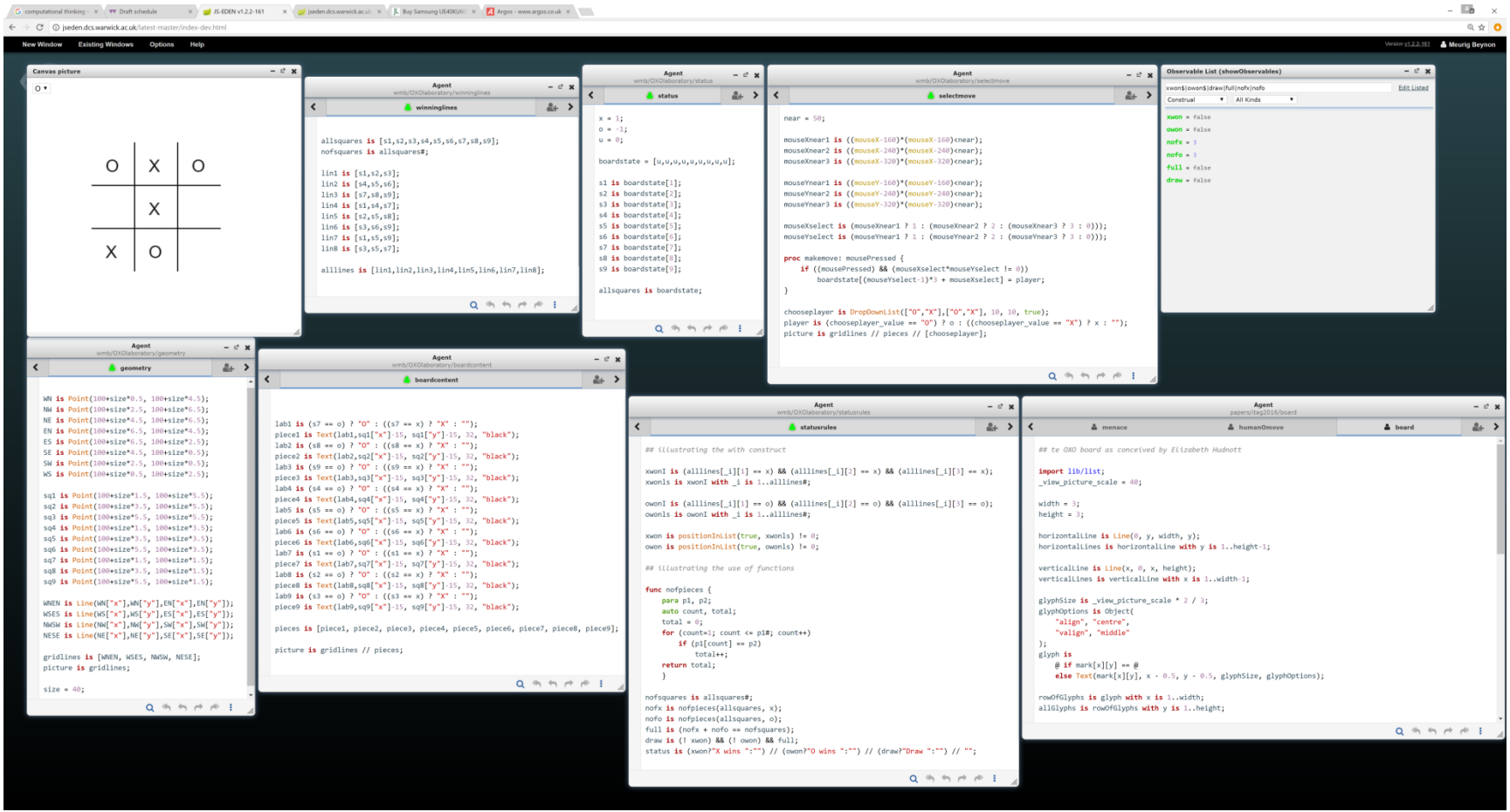
Beynon, M. and Joy, M. “Computer programming for noughts-and-crosses: New frontiers,” in Proceedings of PPIG’94, 1994, pp. 27–37.

Beynon, M. et al, “Making construals as a new digital skill: dissolving the program - and the programmer – interface,” Proceedings of iTAG 2015, pp. 9-16

Thank you

Any questions?


jseden.dcs.warwick.ac.uk/latest-master/index-dev.html
New Window
Existing Windows
Options
Help
Meurig Beynon
Canvas picture
O X O
X
X O
Agent
wmb/OXOlaboratory/winninglines
winninglines
allsquares is [s1,s2,s3,s4,s5,s6,s7,s8,s9];
nofsquares is allsquares#;
lin1 is [s1,s2,s3];
lin2 is [s4,s5,s6];
lin3 is [s7,s8,s9];
lin4 is [s1,s4,s7];
lin5 is [s2,s5,s8];
lin6 is [s3,s6,s9];
lin7 is [s1,s5,s9];
lin8 is [s3,s5,s7];
alllines is [lin1,lin2,lin3,lin4,lin5,lin6,lin7,lin8];
Agent
wmb/OXOlaboratory/status
status
x = 1;
o = -1;
u = 0;
boardstate = [u,u,u,u,u,u,u,u,u];
s1 is boardstate[1];
s2 is boardstate[2];
s3 is boardstate[3];
s4 is boardstate[4];
s5 is boardstate[5];
s6 is boardstate[6];
s7 is boardstate[7];
s8 is boardstate[8];
s9 is boardstate[9];
allsquares is boardstate;
Agent
wmb/OXOlaboratory/selectmove
selectmove
near = 50;
mouseXnear1 is ((mouseX-160)*(mouseX-160)<near);
mouseXnear2 is ((mouseX-240)*(mouseX-240)<near);
mouseXnear3 is ((mouseX-320)*(mouseX-320)<near);
mouseYnear1 is ((mouseY-160)*(mouseY-160)<near);
mouseYnear2 is ((mouseY-240)*(mouseY-240)<near);
mouseYnear3 is ((mouseY-320)*(mouseY-320)<near);
mouseXselect is (mouseXnear1 ? 1 : (mouseXnear2 ? 2 : (mouseXnear3 ? 3 : 0)));
mouseYselect is (mouseYnear1 ? 1 : (mouseYnear2 ? 2 : (mouseYnear3 ? 3 : 0)));
proc makemove: mousePressed {
if ((mousePressed) && (mouseXselect*mouseYselect != 0))
boardstate[(mouseYselect-1)*3 + mouseXselect] = player;
}
chooseplayer is DropDownList(["O","X"],["O","X"], 10, 10, true);
player is (chooseplayer_value == "O") ? o : ((chooseplayer_value == "X") ? x : "");
picture is gridlines // pieces // [chooseplayer];
Observable List (showObservables)
xwon$|owon$|draw|full|nofx|nofo
Edit Listed
Construal
All Kinds
xwon = false
owon = false
nofx = 3
nofo = 3
full = false
draw = false
Agent
wmb/OXOlaboratory/geometry
geometry
WN is Point(100+size*0.5, 100+size*4.5);
NW is Point(100+size*2.5, 100+size*6.5);
NE is Point(100+size*4.5, 100+size*6.5);
EN is Point(100+size*6.5, 100+size*4.5);
ES is Point(100+size*6.5, 100+size*2.5);
SE is Point(100+size*4.5, 100+size*0.5);
SW is Point(100+size*2.5, 100+size*0.5);
WS is Point(100+size*0.5, 100+size*2.5);
sq1 is Point(100+size*1.5, 100+size*5.5);
sq2 is Point(100+size*3.5, 100+size*5.5);
sq3 is Point(100+size*5.5, 100+size*5.5);
sq4 is Point(100+size*1.5, 100+size*3.5);
sq5 is Point(100+size*3.5, 100+size*3.5);
sq6 is Point(100+size*5.5, 100+size*3.5);
sq7 is Point(100+size*1.5, 100+size*1.5);
sq8 is Point(100+size*3.5, 100+size*1.5);
sq9 is Point(100+size*5.5, 100+size*1.5);
WNEN is Line(WN["x"],WN["y"],EN["x"],EN["y"]);
WSES is Line(WS["x"],WS["y"],ES["x"],ES["y"]);
NWSW is Line(NW["x"],NW["y"],SW["x"],SW["y"]);
NESE is Line(NE["x"],NE["y"],SE["x"],SE["y"]);
gridlines is [WNEN, WSES, NWSW, NESE];
picture is gridlines;
size = 40;
Agent
wmb/OXOlaboratory/boardcontent
boardcontent
lab1 is (s7 == o) ? "O" : ((s7 == x) ? "X" : "");
piece1 is Text(lab1,sq1["x"]-15, sq1["y"]-15, 32, "black");
lab2 is (s8 == o) ? "O" : ((s8 == x) ? "X" : "");
piece2 is Text(lab2,sq2["x"]-15, sq2["y"]-15, 32, "black");
lab3 is (s9 == o) ? "O" : ((s9 == x) ? "X" : "");
piece3 is Text(lab3,sq3["x"]-15, sq3["y"]-15, 32, "black");
lab4 is (s4 == o) ? "O" : ((s4 == x) ? "X" : "");
piece4 is Text(lab4,sq4["x"]-15, sq4["y"]-15, 32, "black");
lab5 is (s5 == o) ? "O" : ((s5 == x) ? "X" : "");
piece5 is Text(lab5,sq5["x"]-15, sq5["y"]-15, 32, "black");
lab6 is (s6 == o) ? "O" : ((s6 == x) ? "X" : "");
piece6 is Text(lab6,sq6["x"]-15, sq6["y"]-15, 32, "black");
lab7 is (s1 == o) ? "O" : ((s1 == x) ? "X" : "");
piece7 is Text(lab7,sq7["x"]-15, sq7["y"]-15, 32, "black");
lab8 is (s2 == o) ? "O" : ((s2 == x) ? "X" : "");
piece8 is Text(lab8,sq8["x"]-15, sq8["y"]-15, 32, "black");
lab9 is (s3 == o) ? "O" : ((s3 == x) ? "X" : "");
piece9 is Text(lab9,sq9["x"]-15, sq9["y"]-15, 32, "black");
pieces is [piece1, piece2, piece3, piece4, piece5, piece6, piece7, piece8, piece9];
picture is gridlines // pieces;
Agent
wmb/OXOlaboratory/statusrules
statusrules
illustrating the with construct
xwonI is (alllines[_i][1] == x) && (alllines[_i][2] == x) && (alllines[_i][3] == x);
xwonIs is xwonI with _i is 1..alllines#;
owonI is (alllines[_i][1] == o) && (alllines[_i][2] == o) && (alllines[_i][3] == o);
owonIs is owonI with _i is 1..alllines#;
xwon is positionInList(true, xwonIs) != 0;
owon is positionInList(true, owonIs) != 0;
illustrating the use of functions
func nofpieces {
para p1, p2;
auto count, total;
total = 0;
for (count=1; count <= p1#; count++)
if (p1[count] == p2)
total++;
return total;
}
nofsquares is allsquares#;
nofx is nofpieces(allsquares, x);
nofo is nofpieces(allsquares, o);
full is (nofx + nofo == nofsquares);
draw is (! xwon) && (! owon) && full;
status is (xwon?"X wins ":"") // (owon?"O wins ":"") // (draw?"Draw ":"") // "";
Agent
c6/simpleNandC/humanmove
board
simpleNandC
NoughtsCrosses
humanmove
end_of_game is owon || xwon || draw;
lin_w is lin[1]+lin[2]+lin[3] == -2;
winlineix is lin_w with lin is alllines[ix];
ix = 1;
lin_b is lin[1]+lin[2]+lin[3] == 2;
blocklineix is lin_b with lin is alllines[ix];
ix = 1;
winlines is winlineix with ix is 1..8;
blocklines is blocklineix with ix is 1..8;
gapinlin is 1 if lin[1]==0 else (2 if lin[2]==0 else (3 if lin[3]==0 else 0));
note that if 0 is used in place of 0, then the structure of the list playonlines is lost
playonlinix is gapinlin with lin is alllines[ix];
playonlines is playonlinix with ix is 1..8;
alllinesindices is alllines
with s1 is 1, s2 is 2, s3 is 3, s4 is 4, s5 is 5, s6 is 6, s7 is 7, s8 is 8, s9 is 9;
winindex is _index if winlines[_index] else 0;
_index = 3;


jsseden.dcs.warwick.ac.uk/latest-master/index-dev.html
New Window
Existing Windows
Options
Help
Meurig Beynon
Canvas picture
O X O
X
X O
Agent
wmb/OXOlaboratory/winninglines
winninglines
allsquares is [s1,s2,s3,s4,s5,s6,s7,s8,s9];
nofsquares is allsquares#;
lin1 is [s1,s2,s3];
lin2 is [s4,s5,s6];
lin3 is [s7,s8,s9];
lin4 is [s1,s4,s7];
lin5 is [s2,s5,s8];
lin6 is [s3,s6,s9];
lin7 is [s1,s5,s9];
lin8 is [s3,s5,s7];
alllines is [lin1,lin2,lin3,lin4,lin5,lin6,lin7,lin8];
Agent
wmb/OXOlaboratory/status
status
x = 1;
o = -1;
u = 0;
boardstate = [u,u,u,u,u,u,u,u,u];
s1 is boardstate[1];
s2 is boardstate[2];
s3 is boardstate[3];
s4 is boardstate[4];
s5 is boardstate[5];
s6 is boardstate[6];
s7 is boardstate[7];
s8 is boardstate[8];
s9 is boardstate[9];
allsquares is boardstate;
Agent
wmb/OXOlaboratory/selectmove
selectmove
near = 50;
mouseXnear1 is ((mouseX-160)*(mouseX-160)<near);
mouseXnear2 is ((mouseX-240)*(mouseX-240)<near);
mouseXnear3 is ((mouseX-320)*(mouseX-320)<near);
mouseYnear1 is ((mouseY-160)*(mouseY-160)<near);
mouseYnear2 is ((mouseY-240)*(mouseY-240)<near);
mouseYnear3 is ((mouseY-320)*(mouseY-320)<near);
mouseXselect is (mouseXnear1 ? 1 : (mouseXnear2 ? 2 : (mouseXnear3 ? 3 : 0)));
mouseYselect is (mouseYnear1 ? 1 : (mouseYnear2 ? 2 : (mouseYnear3 ? 3 : 0)));
proc makemove: mousePressed {
if ((mousePressed) && (mouseXselect*mouseYselect != 0))
boardstate[(mouseYselect-1)*3 + mouseXselect] = player;
}
chooseplayer is DropDownList(["O","X"],["O","X"], 10, 10, true);
player is (chooseplayer_value == "O") ? o : ((chooseplayer_value == "X") ? x : "");
picture is gridlines // pieces // [chooseplayer];
Observable List (showObservables)
xwon$|owon$|draw|full|nofx|nofo
Edit Listed
Construal
All Kinds
xwon = false
owon = false
nofx = 3
nofo = 3
full = false
draw = false
Agent
wmb/OXOlaboratory/geometry
geometry
WN is Point(100+size*0.5, 100+size*4.5);
NW is Point(100+size*2.5, 100+size*6.5);
NE is Point(100+size*4.5, 100+size*6.5);
EN is Point(100+size*6.5, 100+size*4.5);
ES is Point(100+size*6.5, 100+size*2.5);
SE is Point(100+size*4.5, 100+size*0.5);
SW is Point(100+size*2.5, 100+size*0.5);
WS is Point(100+size*0.5, 100+size*2.5);
sq1 is Point(100+size*1.5, 100+size*5.5);
sq2 is Point(100+size*3.5, 100+size*5.5);
sq3 is Point(100+size*5.5, 100+size*5.5);
sq4 is Point(100+size*1.5, 100+size*3.5);
sq5 is Point(100+size*3.5, 100+size*3.5);
sq6 is Point(100+size*5.5, 100+size*3.5);
sq7 is Point(100+size*1.5, 100+size*1.5);
sq8 is Point(100+size*3.5, 100+size*1.5);
sq9 is Point(100+size*5.5, 100+size*1.5);
WNEN is Line(WN["x"],WN["y"],EN["x"],EN["y"]);
WSES is Line(WS["x"],WS["y"],ES["x"],ES["y"]);
NWSW is Line(NW["x"],NW["y"],SW["x"],SW["y"]);
NESE is Line(NE["x"],NE["y"],SE["x"],SE["y"]);
gridlines is [WNEN, WSES, NWSW, NESE];
picture is gridlines;
size = 40;
Agent
wmb/OXOlaboratory/boardcontent
boardcontent
lab1 is (s7 == o) ? "O" : ((s7 == x) ? "X" : "");
piece1 is Text(lab1,sq1["x"]-15, sq1["y"]-15, 32, "black");
lab2 is (s8 == o) ? "O" : ((s8 == x) ? "X" : "");
piece2 is Text(lab2,sq2["x"]-15, sq2["y"]-15, 32, "black");
lab3 is (s9 == o) ? "O" : ((s9 == x) ? "X" : "");
piece3 is Text(lab3,sq3["x"]-15, sq3["y"]-15, 32, "black");
lab4 is (s4 == o) ? "O" : ((s4 == x) ? "X" : "");
piece4 is Text(lab4,sq4["x"]-15, sq4["y"]-15, 32, "black");
lab5 is (s5 == o) ? "O" : ((s5 == x) ? "X" : "");
piece5 is Text(lab5,sq5["x"]-15, sq5["y"]-15, 32, "black");
lab6 is (s6 == o) ? "O" : ((s6 == x) ? "X" : "");
piece6 is Text(lab6,sq6["x"]-15, sq6["y"]-15, 32, "black");
lab7 is (s1 == o) ? "O" : ((s1 == x) ? "X" : "");
piece7 is Text(lab7,sq7["x"]-15, sq7["y"]-15, 32, "black");
lab8 is (s2 == o) ? "O" : ((s2 == x) ? "X" : "");
piece8 is Text(lab8,sq8["x"]-15, sq8["y"]-15, 32, "black");
lab9 is (s3 == o) ? "O" : ((s3 == x) ? "X" : "");
piece9 is Text(lab9,sq9["x"]-15, sq9["y"]-15, 32, "black");
pieces is [piece1, piece2, piece3, piece4, piece5, piece6, piece7, piece8, piece9];
picture is gridlines // pieces;
Agent
wmb/OXOlaboratory/statusrules
statusrules
illustrating the with construct
xwonI is (alllines[_i][1] == x) && (alllines[_i][2] == x) && (alllines[_i][3] == x);
xwonls is xwonI with _i is 1..alllines#;
owonI is (alllines[_i][1] == o) && (alllines[_i][2] == o) && (alllines[_i][3] == o);
owonls is owonI with _i is 1..alllines#;
xwon is positionInList(true, xwonls) != 0;
owon is positionInList(true, owonls) != 0;
illustrating the use of functions
func nofpieces {
para p1, p2;
auto count, total;
total = 0;
for (count=1; count <= p1#; count++)
if (p1[count] == p2)
total++;
return total;
}
nofsquares is allsquares#;
nofx is nofpieces(allsquares, x);
nofo is nofpieces(allsquares, o);
full is (nofx + nofo == nofsquares);
draw is (! xwon) && (! owon) && full;
status is (xwon?"X wins ":"") // (owon?"O wins ":"") // (draw?"Draw ":"") // "";
Agent
papers/itag2016/board
menace
humanOmove
board
te OXO board as conceived by Elizabeth Hudnott
import lib/list;
_view_picture_scale = 40;
width = 3;
height = 3;
horizontalLine is Line(0, y, width, y);
horizontalLines is horizontalLine with y is 1..height-1;
verticalLine is Line(x, 0, x, height);
verticalLines is verticalLine with x is 1..width-1;
glyphSize is _view_picture_scale * 2 / 3;
glyphOptions is Object(
"align", "centre",
"valign", "middle"
);
glyph is
@ if mark[x][y] == @
else Text(mark[x][y], x - 0.5, y - 0.5, glyphSize, glyphOptions);
rowOfGlyphs is glyph with x is 1..width;
allGlyphs is rowOfGlyphs with y is 1..height;