DetectWins

When a player places three of his or her marks in a horizontal, vertical, or diagonal line, the player wins;

EnforceTurns

To play, one player marks a square in a 3 by 3 grid with X, then the other player marks a square with O, then it is X’s turn again, and so on;

SquareTaken

Once a square is marked, it cannot be marked again;

DefaultOMoves

When other tactics are not applicable, player O should prefer the center square, then the corners, and mark an edge square only when there is no other choice;

“requirements world”

“implementation world”

```
function* detectWinByX() {
 const eventFn = matchAny("X", [
  cell1,
  cell2,
  cell3
 ]);
 yield { wait: eventFn };
 yield { wait: eventFn };
 yield { wait: eventFn };
 yield { request: "XWins" };
}
```

```
function* enforcePlayerTurns() {
 while (true) {
  yield { wait: "X", block: "O" };
  yield { wait: "O", block: "X" };
 }
}
```

```
function* squareTaken(idx) {
 const eventFn = event =>
  (event.type === "X" ||
   event.type === "O") &&
  event.payload === idx;
 yield {
  wait: eventFn
 };
 yield {
  block: eventFn
 };
}
```

```
function* defaultMoves() {
 while (true) {
  yield {
   request: [
    { type: "O", payload: 0 },
    { type: "O", payload: 1 },
    { type: "O", payload: 2 },
    { type: "O", payload: 3 },
    { type: "O", payload: 4 },
    { type: "O", payload: 5 },
    { type: "O", payload: 6 },
    { type: "O", payload: 7 },
    { type: "O", payload: 8 }
   ]
  };
 }
```

Independent units

Append only

Upfront negativity

Multi-modality

No Hierarchy

DetectWins

When a player places three of his or her marks in a horizontal, vertical, or diagonal line, the player wins;

EnforceTurns

To play, one player marks a square in a 3 by 3 grid with X, then the other player marks a square with O, then it is X's turn again, and so on;

SquareTaken

Once a square is marked, it cannot be marked again;

DefaultOMoves

When other tactics are not applicable, player O should prefer the center square, then the corners, and mark an edge square only when there is no other choice;

"requirements world"

"implementation world"

```
function* squareTaken(idx) {
 const eventFn = event =>
  (event.type === "X" ||
   event.type === "O") &&
  event.payload === idx;
 yield {
  wait: eventFn
 };
 yield {
  block: eventFn
 };
}
```

```
function* detectWinByX() {
 const eventFn = matchAny("X", [
  cell1,
  cell2,
  cell3
 ]);
 yield { wait: eventFn };
 yield { wait: eventFn };
 yield { wait: eventFn };
 yield { request: "XWins" };
}
```

```
function* enforcePlayerTurns() {
 while (true) {
  yield { wait: "X", block: "O" };
  yield { wait: "O", block: "X" };
 }
}
```

```
function* defaultMoves() {
 while (true) {
  yield {
   request: [
    { type: "O", payload: 0 },
    { type: "O", payload: 1 },
    { type: "O", payload: 2 },
    { type: "O", payload: 3 },
    { type: "O", payload: 4 },
    { type: "O", payload: 5 },
    { type: "O", payload: 6 },
    { type: "O", payload: 7 },
    { type: "O", payload: 8 }
   ]
  };
 }
```

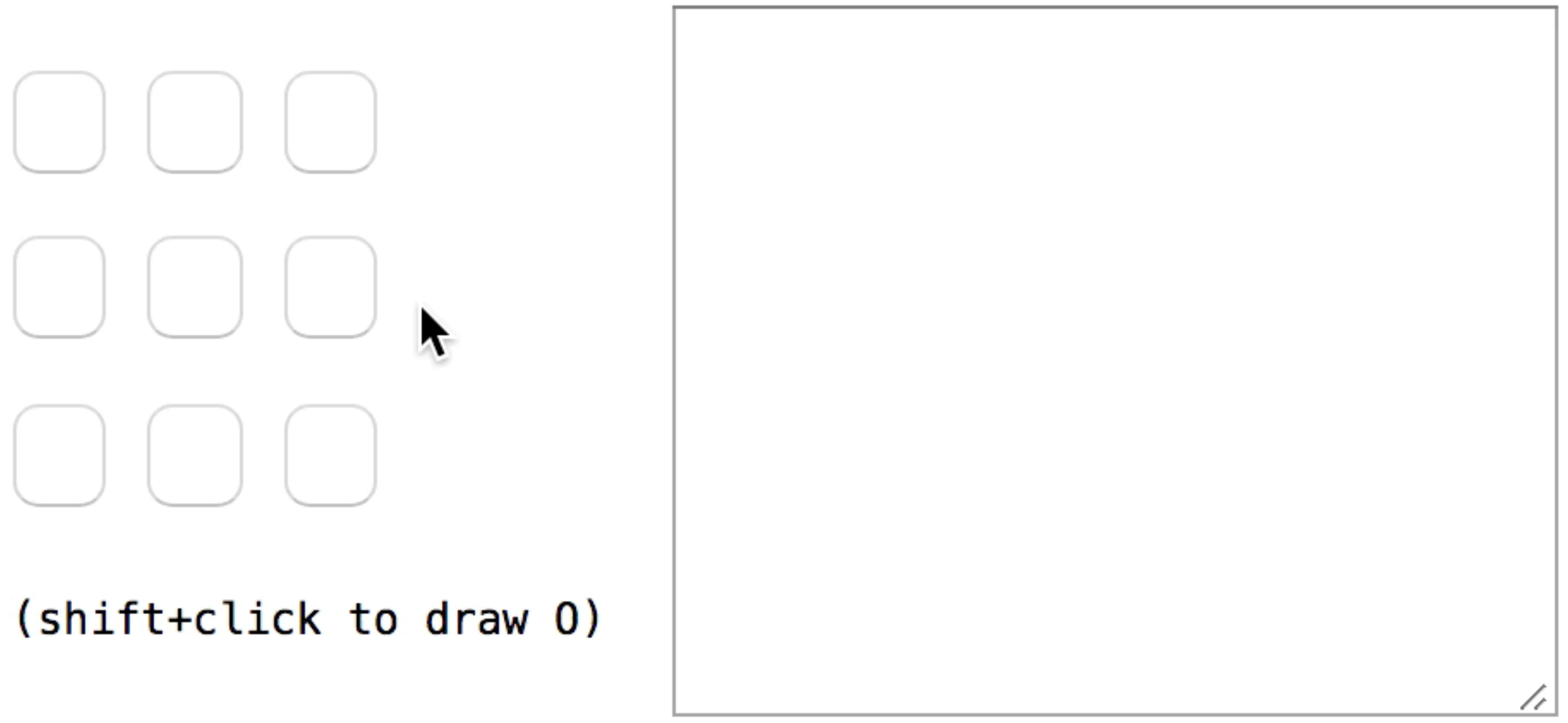
Independent units

No Hierarchy

Append only

Upfront negativity

Multi-modality



(shift+click to draw O)

```
function* enforcePlayerTurns() {
 while (true) {
  yield { wait: 'X', block: 'O' };
  yield { wait: 'O', block: 'X' };
 }
}
```