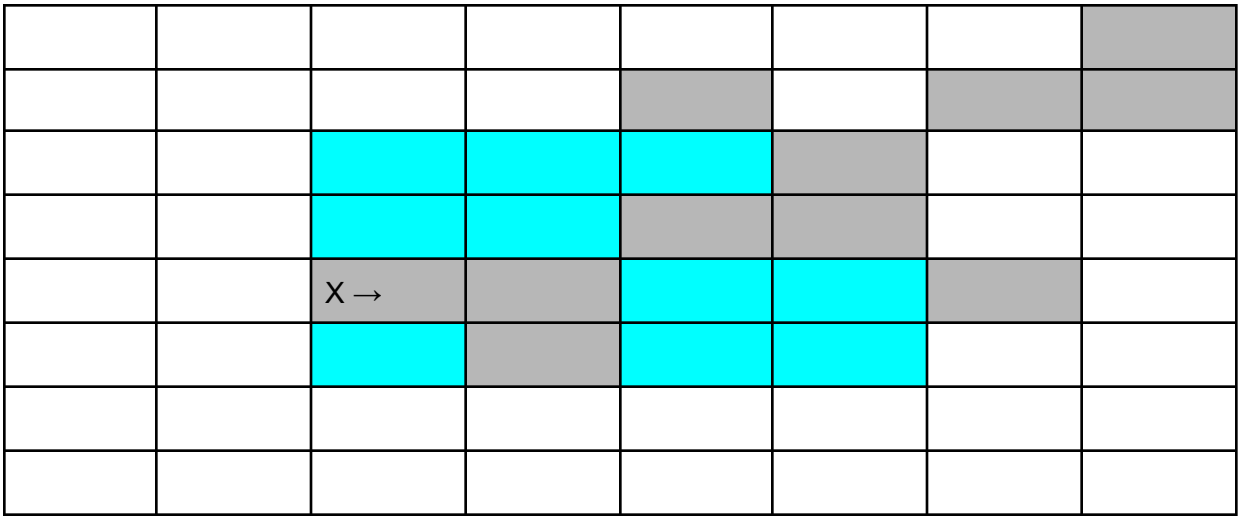
Prototype 1:

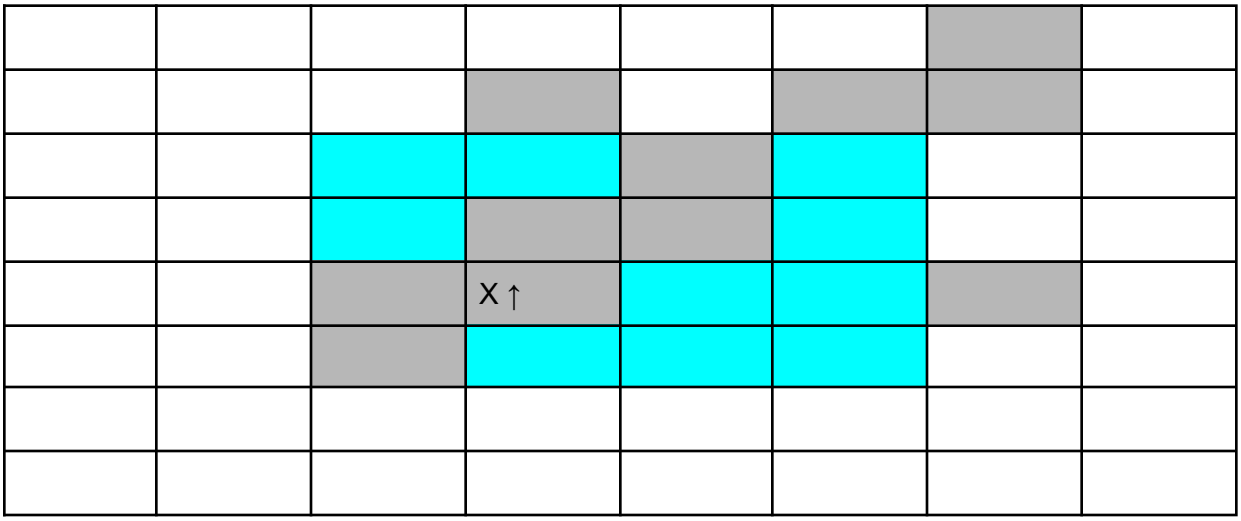
Core mechanics

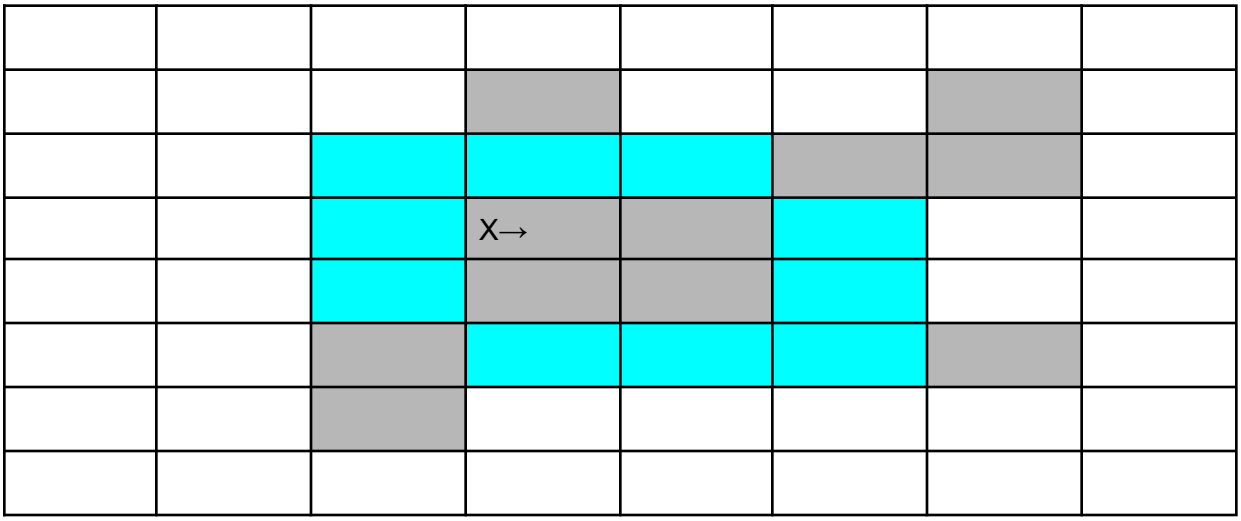
- Player moves left or right: all other rows besides the one the player is on shifts in opposite direction

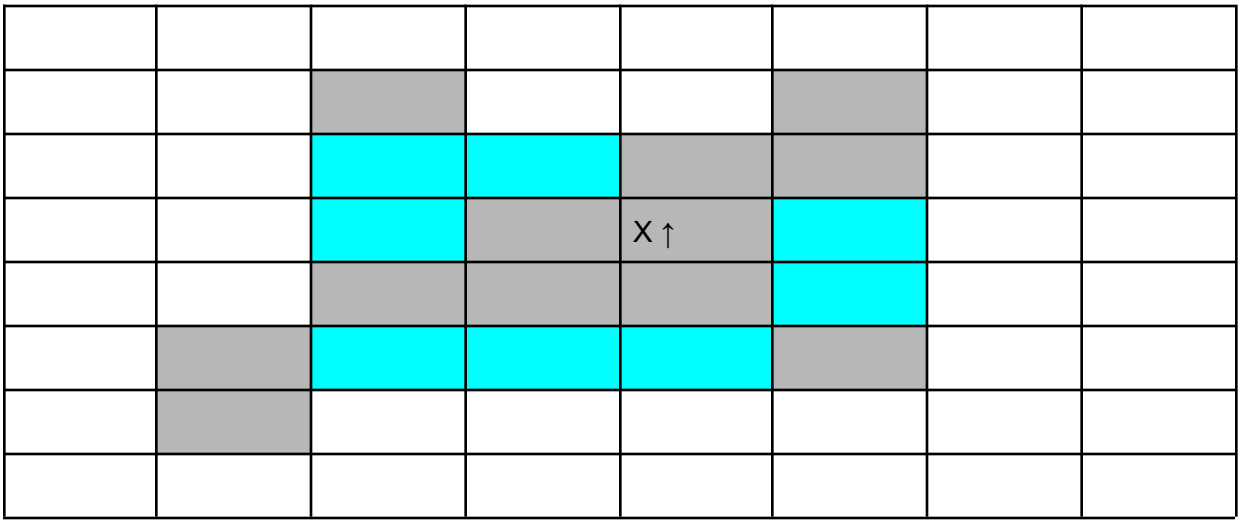
- Player moves up or down: all other columns besides the one the player is on shifts in opposite direction

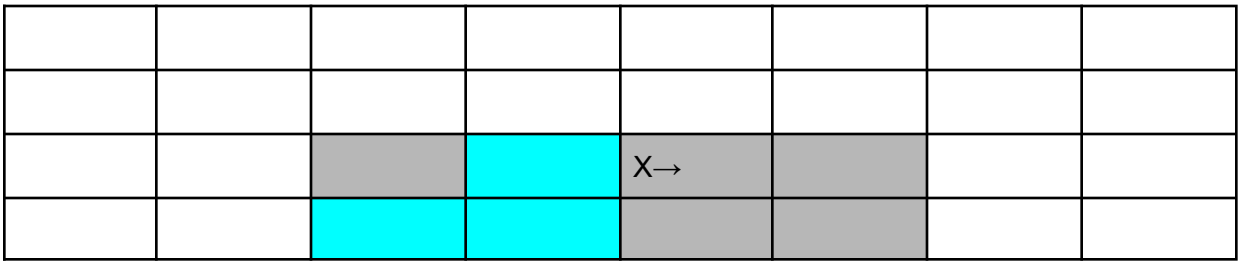
The following is an example puzzle, but I feel like it is too simple / straightforward (only strategy is to keep moving right if possible, and up or down otherwise). Maybe we can add a special colored cloud that when stepped on has some sort of effect (reverses direction of wind, etc.)?

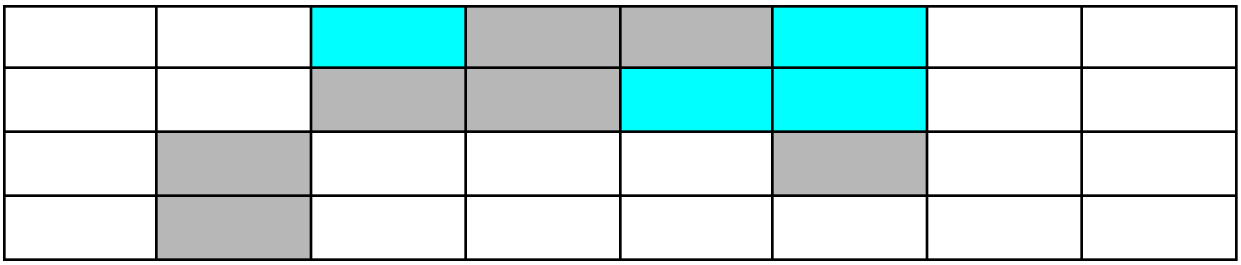


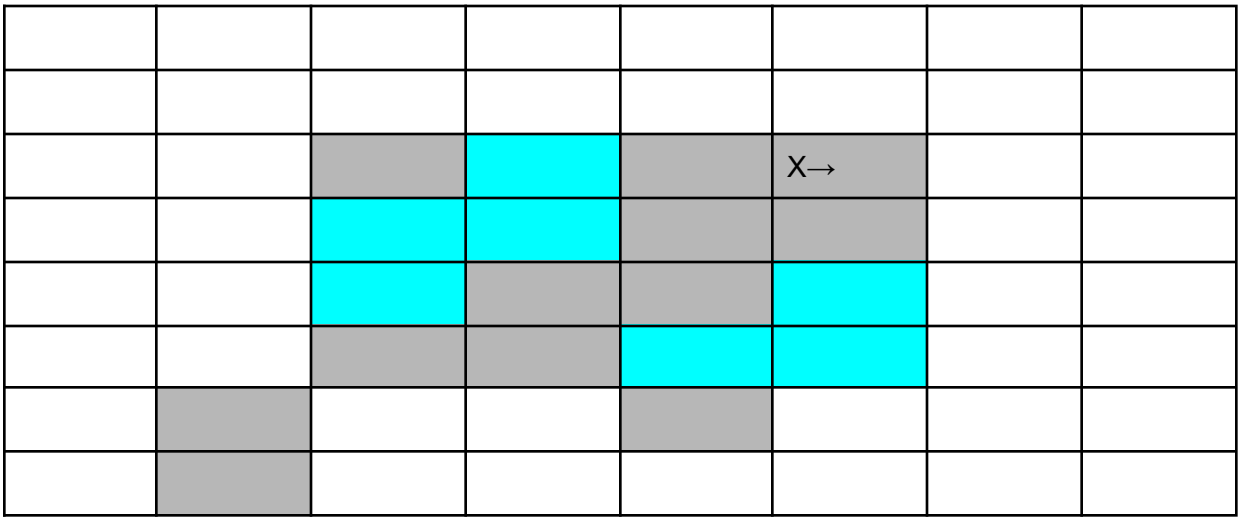



X→


X ↑

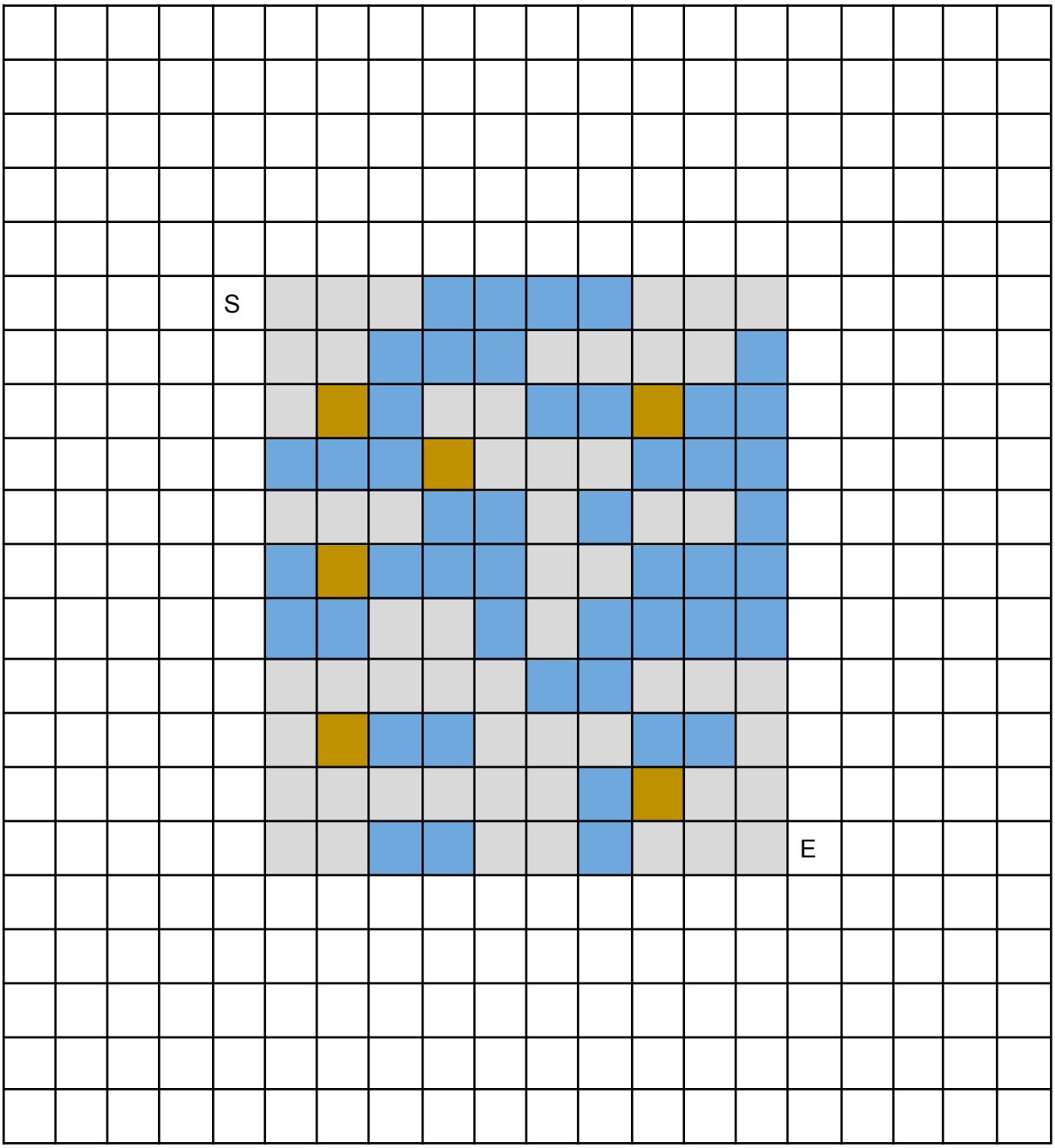

X→

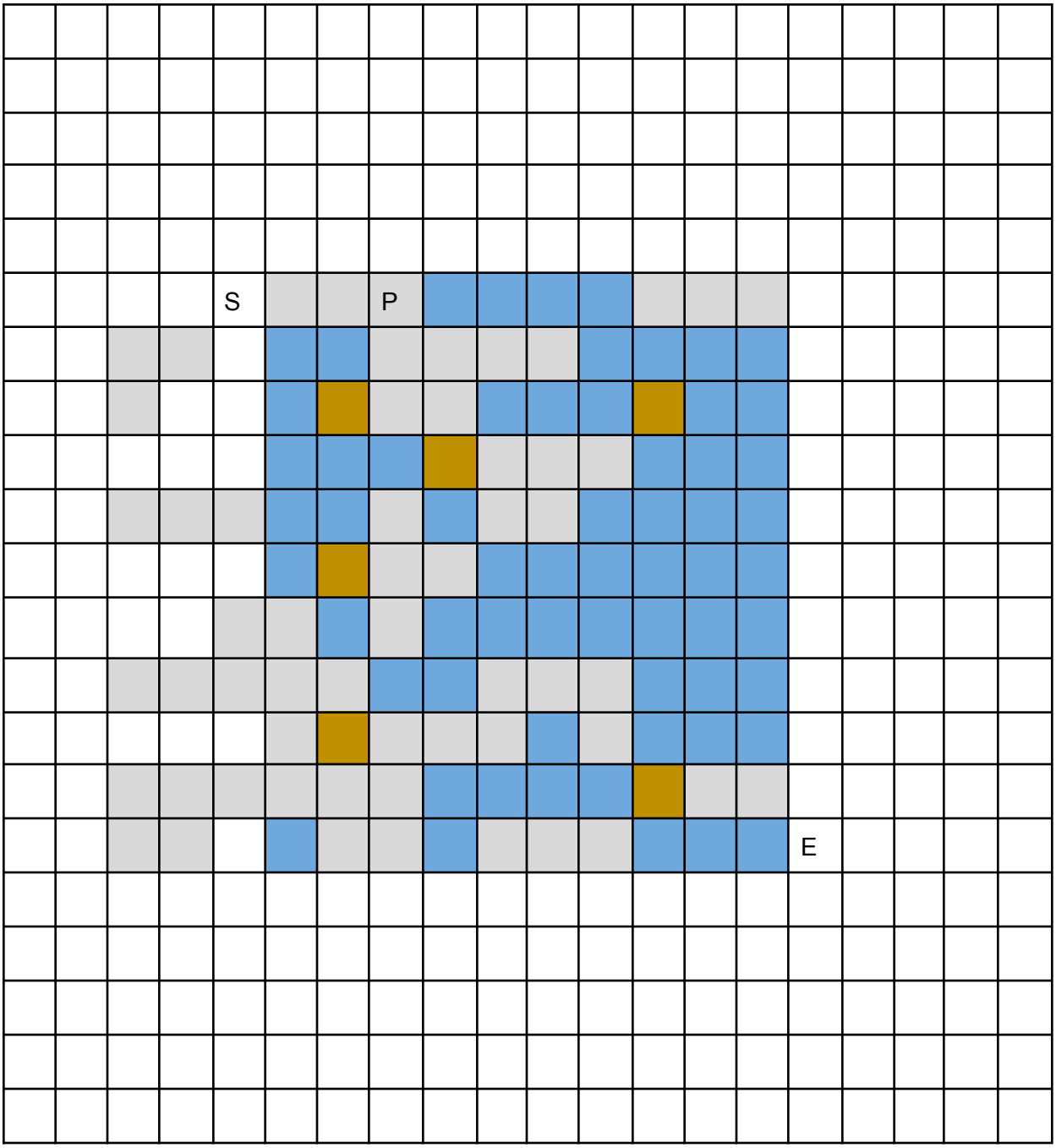



X→

Prototype 2

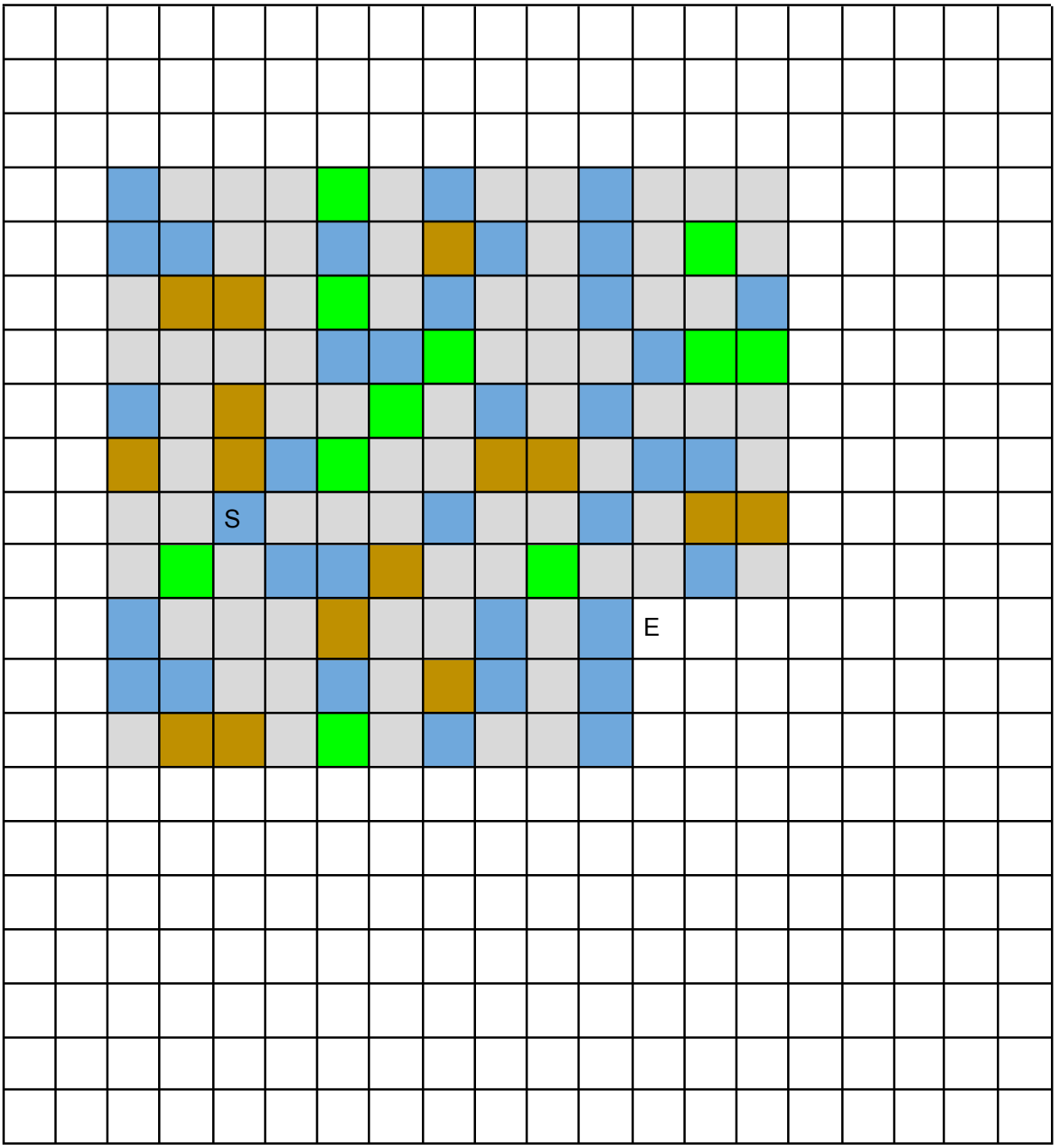
- Larger puzzles, maybe 20x20 with a 10x10 camera?
- Introduced Tile for Mountains, unmovable by shifting winds
- Can be used to merge clouds together, introducing new mechanic




S
P
E

Prototype 3

- Introduce new tree cloud tile
- Essentially a “movable mountain” that blocks players paths
- Most difficult obstacle to navigate around
    - Force players to think strategically and not just randomly guess and check paths



Prototype 4

- Introduce traversable ground tiles
- Fixed size of 12x10
- Can be used to create islands that indicate progress in a puzzle
- Islands introduce new mechanics to the game
    - Strategically build cloud bridges from island to island
    - Use clouds surrounding one island to control clouds on other end of the map
    - Corner-rotation mechanic:
        - Allow for “free” wind shifts in any direction by rotation between a corner ground tile and two adjacent clouds

[We can now design and test levels in the code. No more paper testing!]

```
// Level 1 (Tutorial)
// Designed by William
levels.Add(new Level(new int[,] {
{ 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0 },
{ 0, 0, 0, 0, 0, 0, 1, 2, 1, 1, 0, 0 },
{ 5, 4, 4, 4, 0, 0, 1, 1, 4, 4, 4, 7 },
{ 5, 4, 4, 4, 1, 1, 3, 0, 1, 4, 4, 6 },
{ 0, 0, 0, 4, 0, 1, 4, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0 },
{ 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0 }
}, 3, 0, 3f, colors[6], colors[0]));

// Level 2 (Easy)
// Designed by J.H.
levels.Add(new Level(new int[,]{
{ 0, 0,  0,  4,  1,  0,  0,  3,  1,  4,  4, 6 },
{ 0, 0,  0,  1,  4,  0,  3,  1,  1,  4,  4, 6 },
{ 0, 0,  2,  0,  1,  0,  0,  1,  4,  1,  4, 7 },
{ 5, 4,  4,  0,  4,  4,  0,  0,  0,  1,  1, 0 },
{ 5, 4,  4,  0,  1,  4,  4,  0,  1,  0,  0, 0 },
{ 0, 0,  1,  1,  0,  4,  4,  2,  1,  0,  1, 0 },
{ 0, 0,  1,  1,  0,  1,  0,  1,  0,  1,  0, 0 },
{ 0, 0,  1,  0,  2,  1,  1,  3,  1,  1,  0, 0 }
}, 3, 0, 3f, colors[0], colors[1]));

// Level 3 (Easy-Intermediate)
```

```
// Designed by J.H.
levels.Add(new Level(new int[,] {
{ 0, 0,  0,  1,  1,  2,  0,  1,  0,  3,  0, 0 },
{ 0, 0,  0,  4,  0,  0,  0,  1,  1,  0,  1, 0 },
{ 0, 0,  2,  1,  1,  0,  0,  4,  1,  1,  0, 0 },
{ 5, 4,  4,  0,  0,  1,  2,  0,  0,  0,  1, 0 },
{ 5, 4,  1,  1,  0,  1,  4,  0,  1,  4,  4, 7 },
{ 0, 0,  0,  2,  1,  4,  4,  2,  1,  4,  4, 6 },
{ 0, 0,  4,  3,  1,  1,  0,  1,  0,  0,  0, 0 },
{ 0, 0,  1,  4,  2,  0,  1,  3,  1,  1,  3, 0 }
}, 3, 0, 3f, colors[1], colors[2]));

// Level 4 (Intermediate)
// Designed by J.H.
levels.Add(new Level(new int[,] {
{ 4, 0, 1, 3, 1, 0, 1, 1, 0, 1, 0, 6 },
{ 4, 0, 1, 0, 1, 3, 0, 1, 0, 1, 0, 6 },
{ 4, 0, 1, 2, 1, 0, 1, 1, 0, 1, 0, 6 },
{ 4, 0, 0, 0, 0, 2, 1, 1, 1, 0, 0, 6 },
{ 4, 0, 1, 1, 2, 1, 0, 1, 0, 1, 0, 6 },
{ 5, 4, 0, 1, 0, 1, 2, 3, 1, 0, 0, 6 },
{ 5, 4, 4, 1, 2, 0, 1, 2, 0, 1, 0, 6 },
{ 5, 4, 4, 0, 3, 1, 1, 1, 0, 1, 4, 7 },
{ 4, 0, 0, 3, 1, 1, 0, 1, 0, 4, 4, 6 },
{ 4, 0, 0, 0, 1, 3, 1, 0, 1, 1, 0, 6 },
{ 4, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 6 }
}, 3, 0, 3f, colors[2], colors[3]));

// Level 5 (Intermediate)
// Designed by J.H.
levels.Add(new Level(new int[,] {
{ 0, 4,  0,  1,  1,  2,  0,  1,  4,  0,  3, 0 },
{ 0, 1,  0,  0,  1,  4,  0,  1,  1,  0,  1, 3 },
{ 0, 0,  1,  1,  1,  0,  1,  4,  1,  4,  4, 6 },
{ 5, 4,  1,  0,  0,  4,  2,  0,  0,  4,  4, 7 },
{ 5, 4,  1,  2,  0,  1,  1,  4,  1,  0,  0, 0 },
{ 0, 0,  0,  1,  1,  4,  3,  2,  0,  0,  1, 3 },
{ 0, 1,  3,  4,  1,  1,  1,  4,  0,  3,  0, 0 },
{ 0, 0,  1,  0,  2,  0,  1,  1,  1,  1,  0, 0 }
}, 3, 0, 3f, colors[3], colors[4]));


// Level 6 (Hard)
// Designed by E.L.
```

```
            levels.Add(new Level(new int[,] {
            { 0, 1, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0 },
            { 0, 1, 1, 1, 3, 0, 0, 0, 0, 0, 2, 0 },
            { 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0 },
            { 0, 1, 1, 0, 2, 1, 1, 1, 3, 1, 0, 0 },
            { 0, 0, 0, 2, 1, 0, 1, 0, 1, 0, 1, 0 },
            { 5, 4, 1, 2, 1, 1, 3, 1, 0, 3, 0, 0 },
            { 5, 4, 1, 2, 0, 1, 1, 0, 1, 0, 4, 6 },
            { 5, 4, 0, 3, 1, 1, 0, 1, 1, 1, 4, 7 },
            { 0, 1, 3, 1, 1, 0, 1, 0, 1, 1, 4, 6 },
            { 0, 1, 3, 1, 1, 1, 1, 0, 3, 1, 4, 6 },
            { 0, 1, 2, 0, 0, 1, 1, 1, 1, 1, 0, 0 }
            }, 3, 0, 3f, colors[4], colors[5]));

            // Level 7 (Hard)
            // Designed by J.H.
            levels.Add(new Level(new int[,] {
            { 0,  4,  4,  1,  1,  2,  0,  1,  0,  3,  0, 3 },
            { 0,  0,  1,  1,  0,  4,  0,  1,  2,  0,  1, 3 },
            { 0,  1,  1,  0,  1,  0,  3,  0,  2,  1,  0, 3 },
            { 5,  4,  4,  0,  0,  4,  1,  0,  0,  3,  1, 3 },
            { 5,  4,  4,  0,  4,  4,  4,  0,  0,  4,  4, 7 },
            { 0,  0,  0,  1,  1,  4,  1,  2,  2,  4,  4, 6 },
            { 0,  0,  0,  3,  1,  0,  1,  1,  3,  0,  0, 3 },
            { 0,  0,  1,  0,  2,  0,  1,  3,  1,  1,  0, 3 }
            }, 3, 0, 3f, colors[5], colors[6]));
}
```