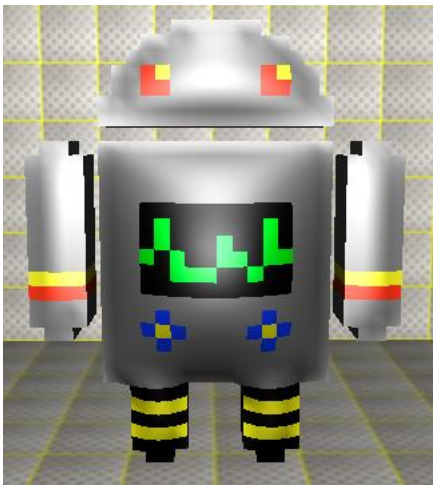
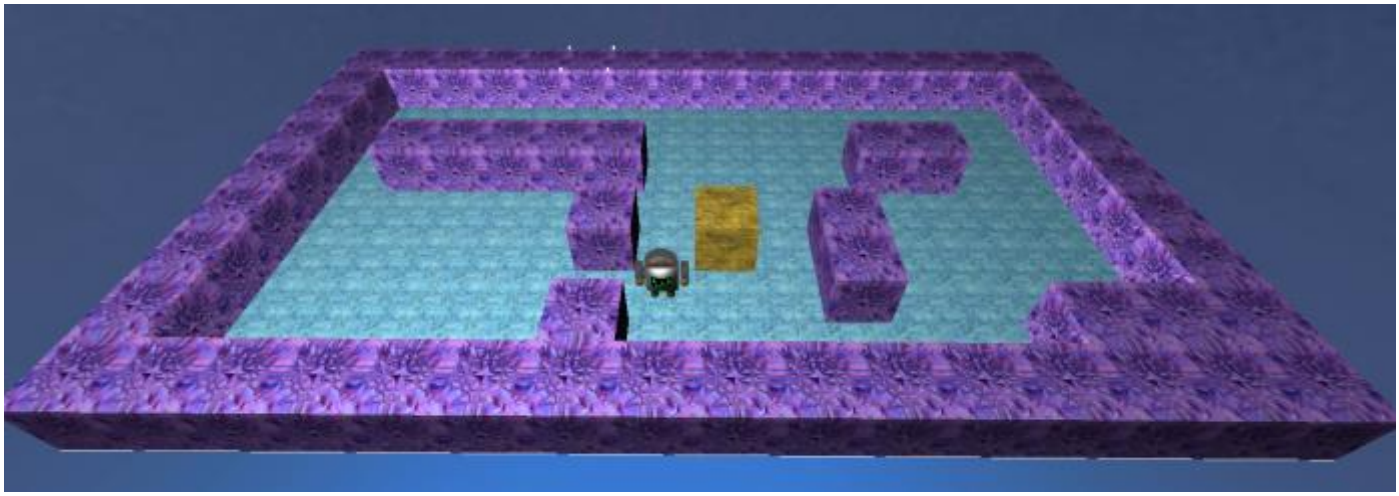


Sokoban Tutorial
for
AgentCubes Online



Created by Catharine Brand, Scalable Game Design Team, University of Colorado, Boulder

Definition:

Sokoban is a puzzle game. The player uses arrow keys to move the Sokoban agent around the world. The goal is to have the Sokoban agent push a crate onto a destination using the fewest possible number of steps. A Sokoban world may have multiple crates and destinations. Walls and narrow passages make it difficult to move crates onto destinations because the Sokoban agent cannot pull a crate. A crate that has been pushed into a corner is stuck there and cannot be moved so the player must restart the game.

Section 1: Setting up a Sokoban world

This section covers concepts introduced in other tutorials so the directions are brief. For more information on agent creation, world creation, movement in 4 directions and not moving on top of other agents, look at the AgentCubes Online Frogger and Journey tutorials.

Step 1: Make a basic Sokoban world.

1. Create a floor tile, a wall cube, a cube named “Crate” and a tile named “Destination”.
2. Create a Sokoban agent named “Soko” who will push the crate onto the destination when the player types the arrow keys.
   Draw your own Soko agent or choose one of the inflatable icon agents.
3. Design your world so that it has a floor and some walls, at least one destination and at least one crate and one Soko agent.

Step 2: Make your Soko agent move when the player types an arrow key.

This rule says, “If the up-arrow key is typed, move up”.
Add rules so that your Soko agent moves in all 4 directions: up, down, left and right.

Step 3: Make your Soko agent avoid moving onto walls and crates but allow it to move onto a destination.

What condition could you add to your movement rules that would prevent Soko from moving onto a wall or a crate?

Think about what agents Soko is permitted to move onto: a floor tile or a destination.

What would happen if you added this condition to your move up rule?

Could Soko move onto walls? No.
Could Soko move onto floors? Yes.
Could Soko move onto destinations? Yes.
Could Soko move onto crates? Yes.

Using the NOT condition will not work because Soko must avoid 2 agents: the walls and the crates.

How can you tell Soko what to move onto?

These conditions tell Soko what to move onto:
But can both conditions go in the same rule?

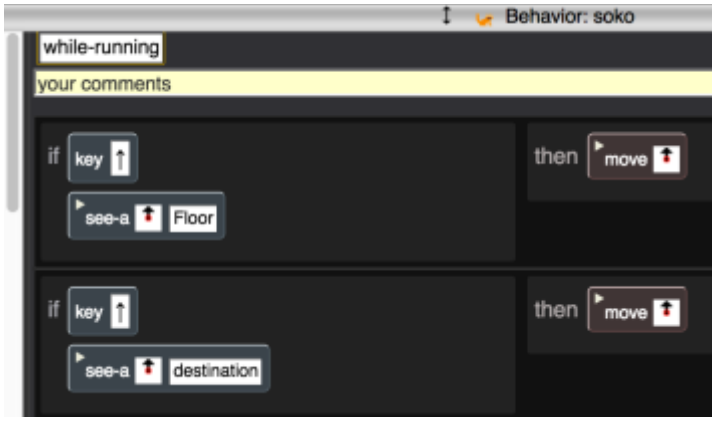
Try it!

What happens?

Why can't Soko move up with both conditions in the same move up rule?

Because the Soko agent can only "see" whatever is visible above it. Either the floor is visible or the destination is visible but Soko cannot "see" both agents above it at the same time so the condition of the move up rule would never be true and Soko would never move up.

How can you allow Soko to move up onto a floor or up onto a destination?
Make two separate move up rules:



Section 2: Making Sokoban Push a Crate to a Destination

What should happen when a crate agent is above the Soko agent and the player types an up-arrow? The player is trying to make Soko push the crate up.

To answer this question, consider two different cases:

1. There is a wall or a crate above the crate that Soko is pushing.
2. There is a floor tile or a destination above the crate that Soko is pushing.

If there is a wall or crate above the crate that Soko is pushing, the crate Soko is pushing should **not** move.

If there is a floor tile or a destination above the crate that Soko is pushing, Soko should be able to push the crate up one square.

Can Soko tell what is above the crate? Why not?

No. Soko can only see agents next to it. Soko cannot see agents that are two squares away.

Here's the solution: When Soko tells the crate agent it's being pushed up, the crate agent must check what is above it: a wall, another crate, a floor tile or a destination. If the crate can move up because there is a floor tile or a destination, it must tell Soko to move up as well and then move itself up.

This is an example of the **Push Computational Thinking Pattern**.

How does Soko tell the crate that it is being pushed and how does the crate tell Soko to move up?

AgentCubes Online has a message action that allows one agent to communicate with another agent.
When an agent receives a message, it looks for a method with that same name and checks the rules inside that method to see if one of them is true.

The action in the picture should be read this way: send a message to the agent above me to do its method named "push_up".

Here is the programming step by step for the **Push Computational Thinking Pattern**:

Step 1. Create a group of move rules for the Soko agent that will handle movement when Soko is next to a crate and wants to push it.

The quick way to make Soko's rules to push a crate is to duplicate Soko's rules to move onto a floor tile.
Your new move up rule should have these two conditions in it:

What action belongs in this rule?
Soko cannot move up because it does not know whether the crate can move up.
Soko must **send the crate a message** using the message action so the crate knows it has been pushed up and can check to see if it can move up.
Any name can be typed into the message box but use push_up so that your code looks like the example.

Here's the whole rule from Soko's **while-running** method:

Step 2. Program the crate agent's push_up method.

First, click on the +Method button at the bottom of the AgentCubes window to make a new method for the crate agent.
Click on the box next to the work "on" at the top of the method and name the method **push_up**.

When can the crate move up?
It can move up if there is a floor tile or a destination above it.

Can the crate check for a floor tile and a destination in the same rule?

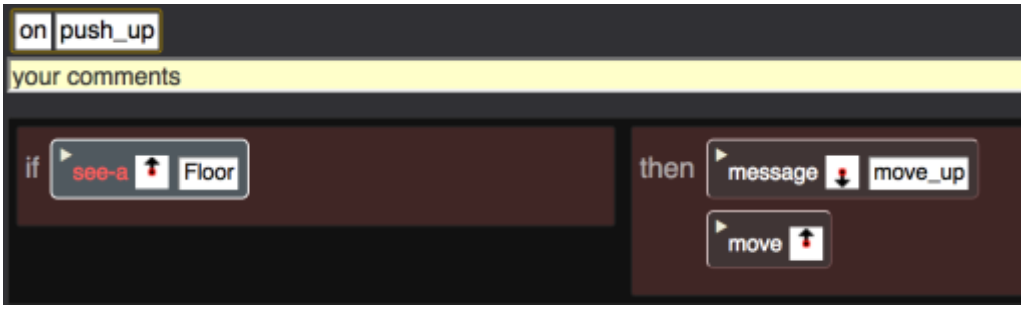
No. So the push-up method needs two rules.

What condition could go in the first rule?
Make the crate check for a floor tile above it.

What action(s) could go in the first rule?
The crate agent needs to move up.

The crate agent must use the message action to tell the Soko agent below it to move up so that it looks like Soko pushed the crate.

Here is the first rule for the crate's push_up method:



The push-up method needs a second rule that lets it move up if there is a destination above it.
Program that rule.

Try running your Sokoban game and making Soko push a crate up.
What happens?

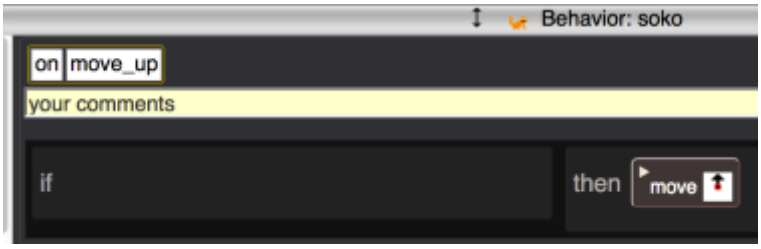
AgentCubes will display an error message telling you that Soko does not know how to move_up.

What is missing?
Soko does not have a move_up method!

Step 3. Program Soko's move_up method.

1. Open Soko's rules.
2. Use the +Method button to make a new method.
3. Click in the box next to "on" and name it "move_up".
4. Add a move action to the rule and make the arrow point up.

Here is Soko's new move method:



Challenge question:
Does it matter which action comes first in the crate's push_up method?
Try reversing the actions in the floor rule.
What do you think will happen? Why?

The crate must tell Soko to move up so that action must be first.
Otherwise, the crate sends a message to the floor agent below it.
The floor agent does not have a move_up method so that is why AgentCubes displays an error message.

It's OK if the crate sends the move_up message first because Soko will not move up until the crate agent has completed all its actions including its own 

The Push Computational Thinking Pattern

- Agent1 sends a message to Agent2 saying it's been pushed.
- Agent 2 checks whether it can move in the direction of the push.
- If Agent2 can move, it sends a message to Agent1 telling it to move too and then moves itself. Agent1 moves when it receives the move message from Agent2.
- If Agent2 **cannot** move, nothing happens.

Section 3: Programming the Step Counter to count correctly

Part of the challenge of the Sokoban game is to use Soko to move the crates to the destinations using the fewest possible number of steps.

We will break keeping track of the number of steps into two parts. First, we will make a counter that knows how to add one to itself. Second, we will program a GameMaster agent that will make the counter add one for each step and that will also check whether Soko has won the game.

Step 1: Create the Number Agent.

1. Click on the +Agent button and name your agent Number. Choose any shape for this agent.
2. Select your number agent by clicking on it with the big arrow tool.
3. Click on the +Shape button next to the +Agent button.
4. Name your new shape "zero" and click OK.
5. Select your new shape and then click on the dot in the white square to the right of your shape.
6. Choose Tile and then find the Numbers category under it and find an image of a zero. Click OK.
7. Continue to add new shapes until you have all the numbers from 0 to 9.
8. **Use the pencil tool to put a <u>zero</u> on the edge of your Sokoban world where you can see it.** Make sure there are at least 2 spaces to the left of the zero so the counter can count to 999 steps.
9. **Save your world!**

Step 2. Program the Number Agent.

The Number agent needs to add one to itself so that it can act as a counter that keeps track of the number of steps taken by the Soko agent. You cannot test the counter until the GameMaster agent is programmed.

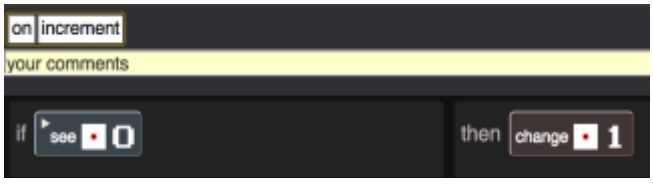
Let's think about what should happen: every time the number agent gets an "increment" (add one) message, it should add one to its current value by changing the shape of the number.
Zero becomes one, one becomes two, and so on.

What happens when nine is incremented?
There are two possibilities.

1. If nine is the only visible number, adding one means replacing the 9 by 10.
   The nine becomes a zero and a new number is created to the left of the zero.
2. If there is a number to the left of the nine, the nine becomes a zero and the number to the left is incremented, so 19 changes to 20.

Programming steps:

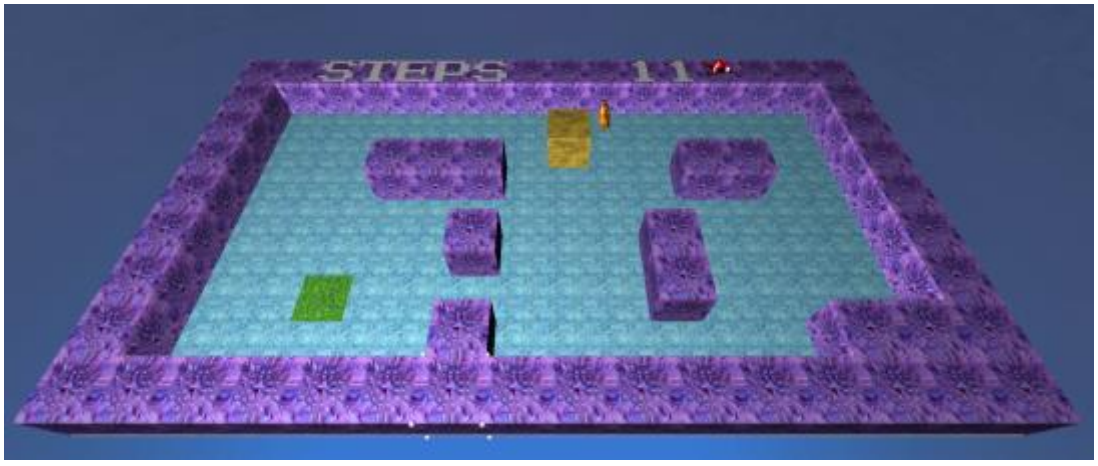
1. Make a new method for the Number agent by clicking on the +Method button and name the method "increment".
2. Make a rule like this one:
3. Make a similar rule for each number shape.



4. Make two rules for 9, following the directions above.
5. The first rule for 9 should say something like “if I see myself looking like a 9 and there is a Number agent to my left, change me to a zero and send a message to the number on my left to increment itself”.
6. The second rule for 9 should say something like “if I see myself looking like a 9 and there is a square without a number to my left, then change me to a zero and make a new “1” to my left”.

Here are the two rules for nine in the Number’s increment method:

Note that in this Sokoban game, there is a wall around the edge of the world and the numbers sit on top of the wall so that they are visible to the player.



Section 4: Programming the GameMaster

The GameMaster agent has three jobs:

1. Updating the step-counter whenever the Soko agent moves a step.
2. **Polling** the crates to see whether any of them are still on the floor.
3. Checking whether the player has won.

Step1. Create the GameMaster agent.

1. Add a new agent named "GameMaster" to your Sokoban game.
2. Give the GameMaster whatever appearance you like. In the tutorial pictures, the lobster is the GameMaster.
3. **Place the GameMaster just to the *right* of your Number zero agent on each level of your Sokoban game. <u>The programming depends on the Number agent being next to the GameMaster.</u>**
4. **Save your world!**

Step 2. Program the GameMaster to make the number agent add one each time the Soko agent moves 1 step.

This goal can be accomplished this way:

1. Soko tells the GameMaster to update the step counter every time Soko moves.
2. The GameMaster tells the Number agent to increment (add one to) itself.

Here's how to program updating the step counter each time Soko moves:
How do Soko and the GameMaster communicate with each other and with other agents?
Using messages.

The message action sends a message to a neighboring agent.
The arrow points to the location of the receiving agent.

How can Soko send a message to the GameMaster if Soko is across the world from the GameMaster?

Soko can use this action:

Broadcast sends the message "update-steps" to every GameMaster anywhere in the world.
In this case, there is only one GameMaster, so the step counter will be incremented once each time AgentCubes does this action.

If we want 1 added to the step-counter each time that Soko moves one square, which rules need this broadcast action added?

All Soko's move rules need this broadcast action added to them. Remember that Soko has move actions in its move_up method and the other similar methods. **Do not add this action to Soko's push rules because Soko may or may not move, depending on what is next to the crate.**

Here's how to program making the step counter increment itself:
When the GameMaster receives the "update_steps" message, what should happen?
The GameMaster must find the method named "update_steps" and check each rule in it to see if one of them has a true condition. Then the GameMaster does the action(s) for that rule.

Make a new method for the GameMaster and name it "update_steps".
The GameMaster needs to tell the step counter to add 1.
This means that the number in the ones' place in the step counter must add 1.
We have already set up the Number agent so that if there is a 9 in the ones' place, it will become a 0 and either put a new Number agent with a 1 shape in the tens' place or tell the Number agent in the tens' place to increment itself.

Should the GameMaster use a message or a broadcast action?
There may be more than one Number agent in the step counter so it would be a bad idea to increment every Number agent in the world.

The best choice is to use a message action to send a message to the Number agent in the ones' place.
We are counting on the GameMaster to be on the right of the Number agent in the step counter's ones' place!

In these pictures, the GameMaster lobster is just to the right of the ones' place in the step counter.

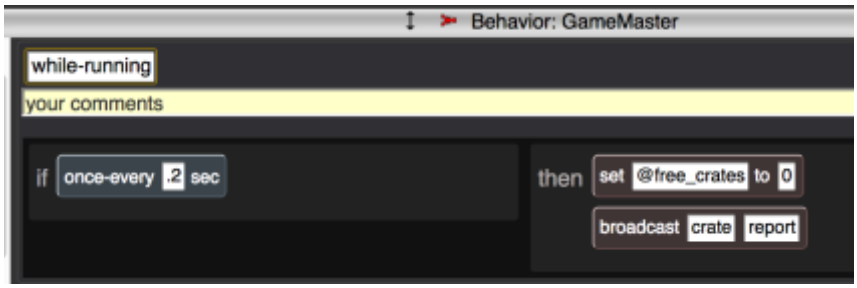

Here is the update_steps method:

Step 3: The GameMaster polls the crates to see whether any crates are not on destinations.

1. Create a simulation property to keep track of crates that are not on destinations.
2. Go to the gear menu  on the far left of the AgentCubes Online menu bar and select the "Show Simulation Properties" option.
3. Click on the plus in the simulation properties window to add a new simulation property and name the new property Free_crates.
4. **Click Save on the simulation property window to save your new property!**
5. Set up the rule in the GameMaster's while_running method with a once_every .2 sec condition.
6. In the action side of the GameMaster's while_running rule, set the **@Free_crates** simulation property to 0.

7. Should the GameMaster use a message or a broadcast action to tell all the crates to report? This time, the GameMaster needs to reach all the crates in the world whether they are on neighboring squares or across the world so broadcast is the correct choice and the GameMaster broadcasts a report message to all crates.

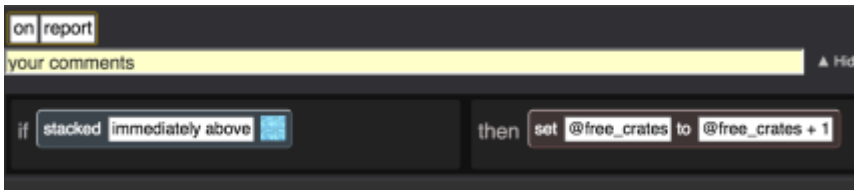
   Here is the GameMaster's while_running rule set up for polling:



8. What does the crate agent do when it receives a report message? It must check whether it is still on the floor and if it is, increment the **@Free_crates** simulation property.

   Remember that the '@' must be used in front of the simulation property name so that AgentCubes does not interpret Free_crates as an agent attribute.

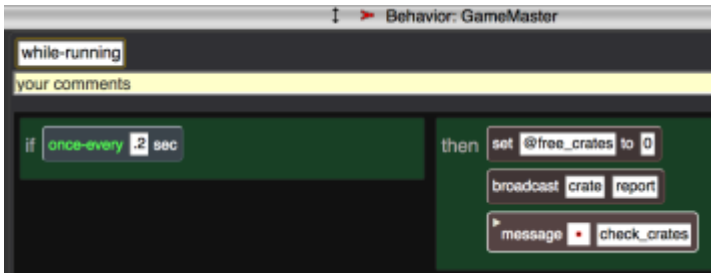
   Here is the crate's report method:



Step 4. The GameMaster must check whether the player has won the level.

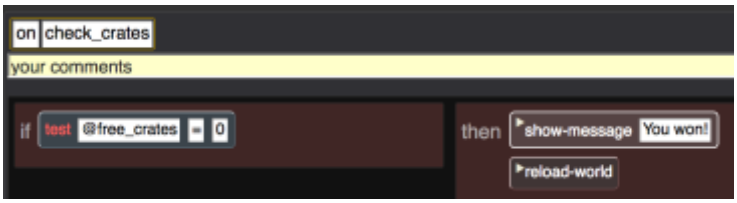
After asking the crates to report, the GameMaster sends a message to itself to see whether the simulation property **@Free_crates** equals zero because all the crate agents are sitting on destinations.

Here is the final version of the GameMaster's while_running method:



Make a check_crates method for the GameMaster.
What condition lets an agent check a value? Use the test condition to check whether @Free_crates equals 0. Display a message to let the player know s/he won and reload the world.

Here is the GameMaster's check_crates method:



The Polling Computational Thinking Pattern

- The polling agent (the GameMaster) sets a global simulation property to zero.
- The polling agent sends a message to all agents it needs to poll (the crates) to update the simulation property.
- The polling agent can use the new information in the simulation property to do something (end the game on the current level).

Step 5: Test your program!

1. Does your step counter count all the steps Soko takes, including when Soko is pushing a crate or moving onto a destination?
   Move Soko slowly and count the steps yourself.
   Make sure that all Soko's move rules, including the move actions in Soko's methods, have a broadcast action that tells the GameMaster to update_steps.
   Do not put a broadcast action in Soko's push rules because Soko may or may not move, depending on what is next to the crate.

2. Does the player win when all the boxes are on destinations?
   Check that there is an "@" before every reference to Free_crates.
   Make sure that the Free_crates simulation property is still listed in the simulation properties window. If not, recreate Free_Crates by clicking on the '+' in the simulation property window and save it.

Challenges:

1. Make your Soko agent face the direction that it is moving or pushing.
2. Create some worlds that make it harder for the player to get the crates onto the destinations. Use more walls and narrow passages. Add more crates. Look at some of the Sokoban puzzles online for ideas.
3. Let the player control two agents that can push crates. How would you count steps in this game?
4. Make several different crate agents and give each crate a matching destination. The player wins by pushing each crate to its own destination. Crates and destinations could have matching numbers or colors.
5. Add some other elements to the game such as sinkholes, teleport pads, hungry monsters, one-way passages or moving blocks.
6. Program Soko to pull crates as well as push them. Would this make the game easier or harder to win?