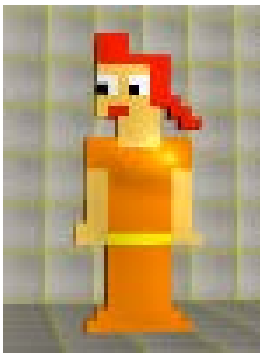


Creating "Journey"
With AgentCubes Online

You are a traveler on a journey to find a treasure. You travel on the ground amid walls, chased by one or more chasers. The chasers at first move randomly on the ground, and later, begin to chase by following your scent. When you collect the treasure, you win. If a chaser catches you, you lose.

Created by: Cathy Brand, University of Colorado
Edited by Jeffrey Bush, University of Colorado, School of Education

This curriculum has been designed as part of the Scalable Games Design project.
It was created using portions of prior work completed by Susan Miller.

This material is based upon work supported by the National Science Foundation under Grant No. DRL-1312129 and CNS-1138526. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Vocabulary/Definitions

Algorithm.....**a set of instructions designed to perform a specific task.**

Attribute.....**a value assigned to an agent (such as scent)**

Bird's Eye View..... **looking down on a World as if you were flying over it**

Brackets .....**method of setting information apart using "[" and "]"**

Broadcast ..... **controller agents broadcast (or send out) messages**

Chaser ..... **the agent that chases the traveler**

Collision .....**an event wherein two agents run into each other.**

Diffusion.....**the process in which an attribute (in this case, scent) changes its value, being larger near its source and smaller farther way from its source**

Increment.....**to increase by one**

Hill Climbing.....**a local search technique, or algorithm, that attempts to find the best solution by testing each possible solution in turn until no better solution can be found. Here the algorithm searches for the strongest scent in the grid squares surrounding the current square.**

Local Variable.....**a variable (attribute) belonging to a specific agent**

Method .....**a named set of rules evaluated by an agent in response to a message**

Parentheses.....**method of setting information apart using "(" and ")"**

Polling.....**the process of asking agents to update a simulation property and then taking some action based on the value of the simulation property**

Propagated.....**spreading a value (scent) through a grid of agents**

Randomly.....**to occur in non-systematic ways**

Rule Order.....**the order in which rules are placed for each agent**

Simulation Property A named value that all agents can see and update

Traveler.....**the main character who is searching for treasure**

Student Handout 1:

Part I - Basic Game

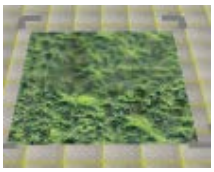
Initial Story: You are a traveler on a journey to find a treasure. You travel on the ground amid walls along with one or more chasers. The chasers move randomly on the ground. When you collect the treasure, you win. If a chaser catches you, you lose.

Create these Agents:
Choose the Inflatable Icon – People category and pick 2 images for your agents:

Traveler

Chaser

Choose a Tile to be the ground. Pick a color that contrasts with your traveler and chaser.



Choose a Cube for making walls. Pick a color that contrasts with the ground.

Choose an image for your Treasure agent from the Inflatable icon – miscellaneous category.

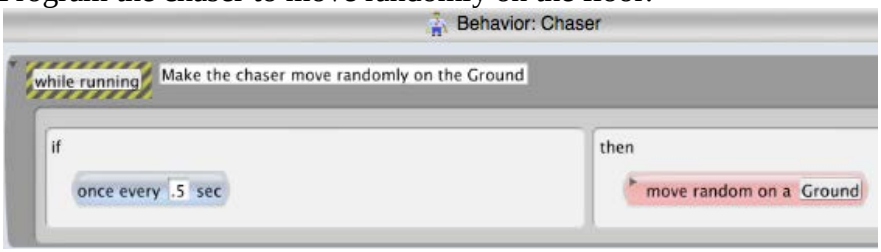
3-D Journey (Continued)

Create your Level 1 World:

Create the following BEHAVIORS for your agents:

Step 1: Chaser:

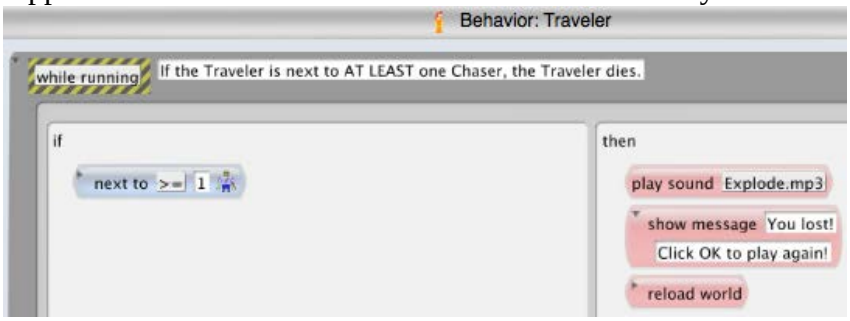
Program the chaser to move randomly on the floor.



Step 2: Traveler:

Set up your agent to move with the arrows (cursor control).

Create game ending conditions (collision). Make sure that the game ends if the Traveler is caught by 1 or more Chasers. Create a similar rule if your traveler approaches the treasure. What happens if the win rule is below the move rules? Does your Traveler always win? Why not?



Be SURE to reload the World when the game ends.

Step 3: Walls

Add walls to your worksheet. Then, prevent your Traveler from walking through the walls. Work with the person next to you to figure out how to prevent the Traveler from walking into a wall. Here is one way to think about it. Challenge yourselves to find a different way!

Student Handout 2
Part 2 – Making the Chaser Chase the Traveler

So far, your Chaser just moves randomly…he doesn't actually chase the traveler, does he? That's about to change!

We will make the Chaser pursue the Traveler agent using a search algorithm called "hill climbing." An algorithm is a set of rules followed by an agent to achieve a goal. Imagine the traveler agent emits a scent. Using the hill climbing search algorithm, the Chaser finds the direction in which the scent is strongest and moves that direction, following the Traveler.

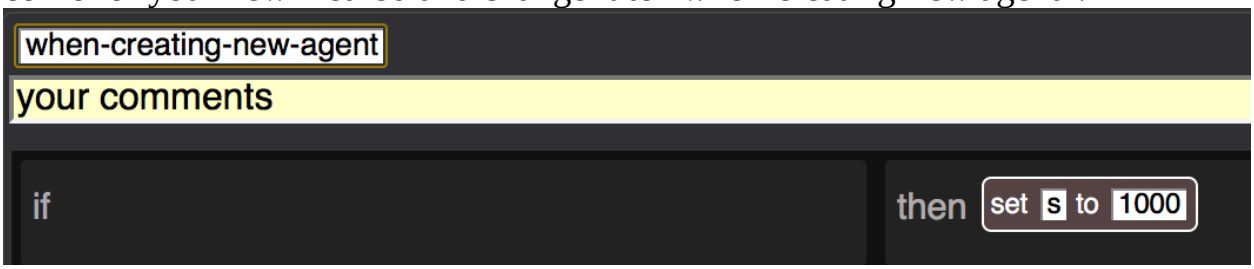
The scent will spread out or be **propagated** by the ground agents using a computational thinking pattern called "diffusion" which copies the physical process of diffusion by which molecules move from areas of highest concentration to areas of lowest concentration. In this game, the values spread out from the Traveler to all the ground agents in the World. The values are highest in the ground agents close to the Traveler and smallest in the ground agents far away from the Traveler.

We will introduce the concept of an "**agent attribute**," which is a piece of information that is stored within each occurrence of an agent. Computer scientists call this attribute a **local variable** because each agent has its own copy of it and each copy has its own value.

Step 1: Set the Traveler's Scent attribute S.
First, let's make sure our traveler gives off a scent. To do this, we need to set the value of an attribute named "S" which stands for Scent. The scent attribute is associated with the Traveler and is set when the Traveler is created by being drawn on the world. This guarantees that the scent attribute always has a meaningful value when the game starts running.

3-D Journey (Continued)

To this, we must click on the +Method button at the bottom of the AgentCubes Online window and make a new method. Click on the word "on" in the upper left corner of your new method and change it to "when-creating-new agent".



Erase your Traveler agent and redraw it, then <u>SAVE</u> the world, so that the Traveler is saved with its scent attribute S set to 1000.

Step 2: Add a rule to the ground agent.

Now, since the scent is diffusing, or spreading out from the Traveler, we need to find the value of the scent in each ground agent. Imagine that the smells are coming in from the North, South, East and West of each ground agent. The value of the smell in any ground agent, then, is the average of the smells in the four surrounding agents. How will you program that?

The ground agent will have the behavior below; the single action is to calculate and store the
average of the values of the four surrounding agents' S attributes. Remember, you used the arbitrary name of the agent attribute "s" (for scent).

The "set" action sets each ground agent's attribute "s" to the **<u>average</u>** of the values of the attributes in the agents above, below, and on each side:

s = 0.25*(s[up]+s[down]+s[right]+s[left])

3-D Journey (Continued)

Match both the parentheses "(" and the brackets "[" as shown in the equation.

A METHOD is a set of rules to follow in a specific situation. You can create a METHOD by clicking the +Method button at the bottom of the AgentCubes window.

Why do we multiply by 0.25?

When you find the average of a set of numbers, you add them up and divide by the number of numbers.

In this case, dividing by 4 is the same as multiplying by 0.25

Step 3:

For the Chaser to know which way to walk, it has to determine where the scent is the strongest. If this were real life, it would smell up, smell down, smell left and smell right. Wherever the smell was strongest, it would walk in that direction. We need to program the Chaser to do this.

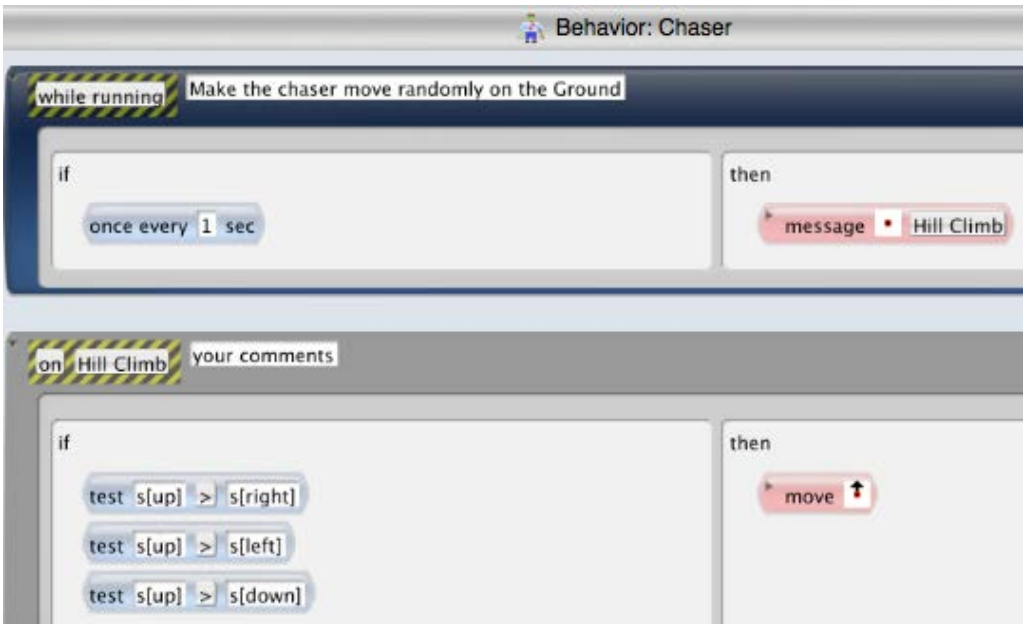
We will create a METHOD that enables the Chaser to do a hill climbing search so that it moves in whichever direction the scent is strongest.

The rule in the while running method says:

ONCE EVERY 1 second, send me a message to do my Hill Climb method.

The first rule in the Hill Climb method says:
IF the smell **above you** is greater than or equal to any of the other smells in different directions (down, left or right), THEN move up.

3-D Journey (Continued)



Now, add the <u>three other rules</u> so that the Chaser knows what to do if the smell down (s[down]) is greater. What if the smell to the left or the right is greater?

What is the scent is exactly the same in all four directions? What would the chaser do? Add this 5th rule at the **bottom** of the **hill climbing method box**:

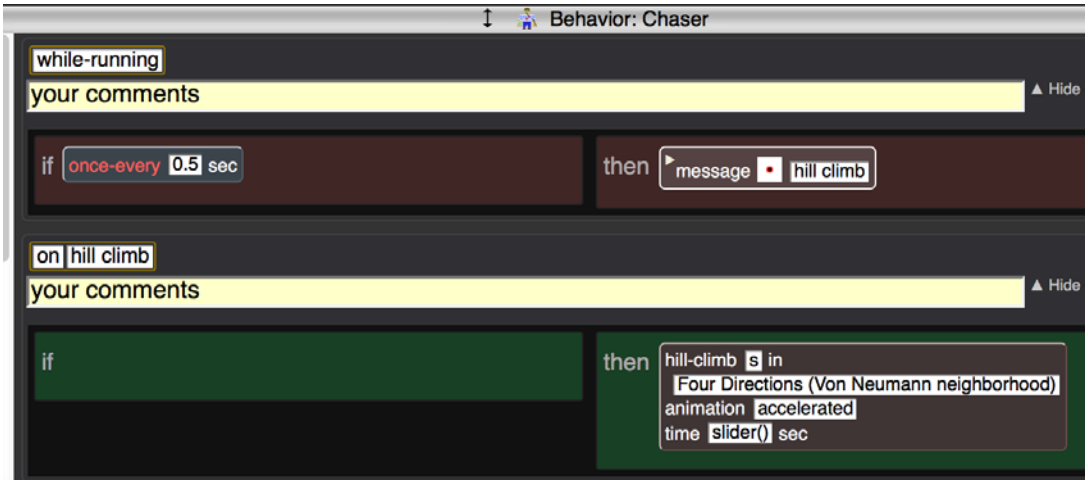
With this rule added, the Chaser will always make a random move if the S values in all 4 directions are the same. For example, the S values may all be equal to 0 if the Traveler's S value has not diffused all the way across the world yet.

Note: this rule must be the last rule in the Chaser's hill climbing method!

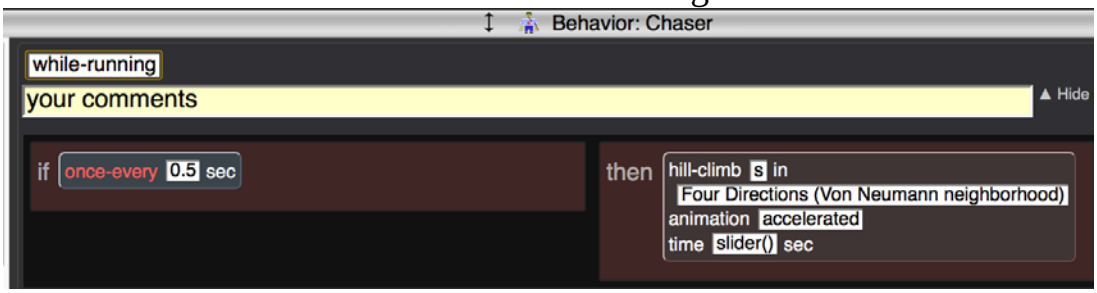
Shortcut for Hill Climbing

AgentCubes Online has an even easier way to handle the process of Hill Climbing. The use of the Hill Climbing action (see rule below) simplifies your code and eliminates errors.

There is a single action, hill climb, which replaces all the rules in the Hill Climb method.



It is possible to eliminate the Hill Climb method by simply putting the hill climb action in the rule in the Chaser's while-running method.



Test out the options in the hill climb action.

- What happens if the Chasers search in 8 directions?
- Can the Traveler escape?
- Would it help to use fewer Chasers?

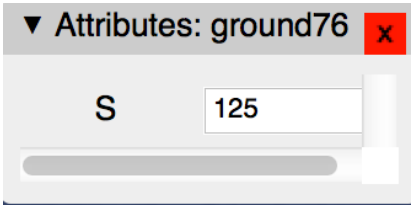
After you have run your game several times, choose whether your Chasers will search in 4 or 8 directions and decide on the number of Chasers in your game.

Student Handout:

Troubleshooting Guide for Journey Part II

Diffusion and Hill Climbing

To determine what is happening in your game, it is sometimes helpful to look at the agent attributes. Reset your game, then single step the game by clicking on the black triangle  next to the red stop button. Do not reset at this point. Since your game has run briefly, the ground agents close to the Traveler now have a scent. Check the s value of the ground agents by double clicking on them with the big arrow tool  . You can also click on the gear button  on the top right edge of the AgentCubes Online window and select "Show Agent Attributes.



This window will appear after double clicking or using the gear button menu:

Click on different agents in the world with the big arrow tool to see their S values.

Try checking the attributes of the four ground agents around the Chaser (up, down, left and right) and then single stepping the game using the black triangle next to the stop button. Does your Chaser move in the direction you expected him to go?

If your game isn't working, it's time to do some troubleshooting. Check the following:

- After programming the when-creating-new-agent method for the Traveler, make sure that you erased the Traveler, redrew that agent and **Saved the world** so that the Traveler has an S value = 1000.
- Use of parentheses "(" and brackets "[" in the Ground agent rule must be correct. Look at the picture of the Ground agent's equation 2 pages ago and compare it to the equation in your ground agent.

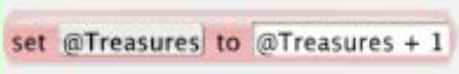
Student Handout Part 3:

Adding Challenge to the Game – Polling and Broadcast

We will add another challenge to our Journey Game. Now the Traveler must "collect" – that is move on top of – multiple treasures in order to win. The game does not end until all of the treasures are collected.

To accomplish this, we introduce the concept of SIMULATION PROPERTIES, which are named values that can be used and checked by all agents in a project. Simulation properties are also known as global variables since all agents can check their values.

A Simulation Property name is always preceded by an "@" when its value is needed in an action



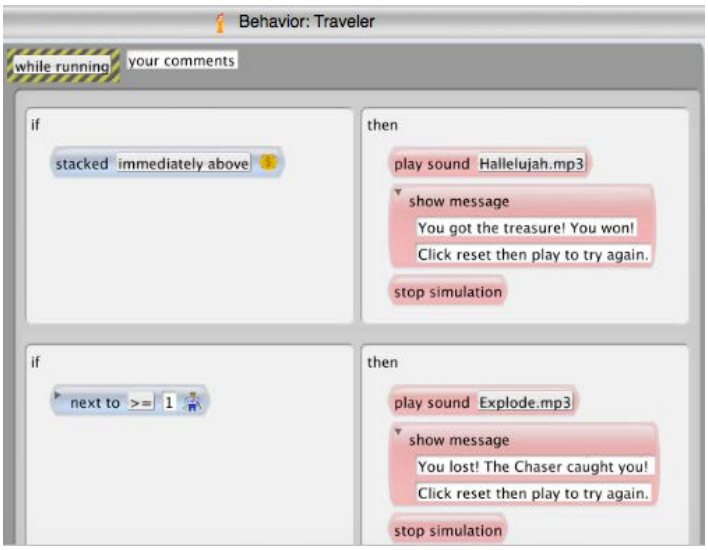
or a condition

.

The @ differentiates simulation properties from agent attributes.

We will create a new agent, the "Controller" to manage the process of polling the treasure agents to determine when they are all "collected"; that is, when there are none left on the worksheet. To begin, we must change the behavior of the traveler agent so that it no longer declares the game is over when it moves on top of the treasure.

Step 1: Remove the win rule from the Traveler that makes the game end when she moves above the treasure.



Highlight the rule by clicking on the bar between the condition and action. Then press the delete button on your keyboard.

3-D Journey (Continued)

Step 2: Create a Controller agent
Create a Controller agent, using a colorful tile or any predefined shape or drawing your own. Use the pencil tool to place **one** Controller on your World.

Step 3: Counting up the treasures to see whether the player won and the game has ended.

Imagine this conversation…That process is similar to the way polling will work in your program.

The teacher has given an assignment to the class and wants to know if everyone is finished. She says to the class, "Put your hand up if you are still working." Hands go up. She counts them – there are five students still working. "Okay, put your hands down and keep working."

A few minutes later, she does it again. She says to the class, "Put your hand up if you are still working." Hands go up. She counts them – there are two students still working. "Okay, put your hands down and keep working."

A few minutes later, she does it again. She says to the class, "Put your hand up if you are still working." This time, no hands go up. "Everyone is done, put your books away."

Once per second, the Controller will say, "Treasure agent count starts at zero" (like the classroom, no hands are up when the teacher asks who is still working).

When the treasure agents 'hear' the Controller ask (broadcast) the question, the treasure agents respond back (raise their hands).

The controller checks the treasure agent count. If this count is more than zero, nothing happens and the game continues. If the answer is zero (meaning that there are no remaining treasures on the board), the game ends.

How does Polling work?

In its "While Running" method, the Controller agent first sets the simulation property "@Treasures" to zero.

Then it broadcasts a signal "Count" to all treasure agents. Each treasure agent responds by adding one to the @Treasures simulation property.

Finally, the Controller calls upon the "Check Win" method. The player wins if no treasure agents are left in the world, which is determined by the @Treasures simulation property being zero.

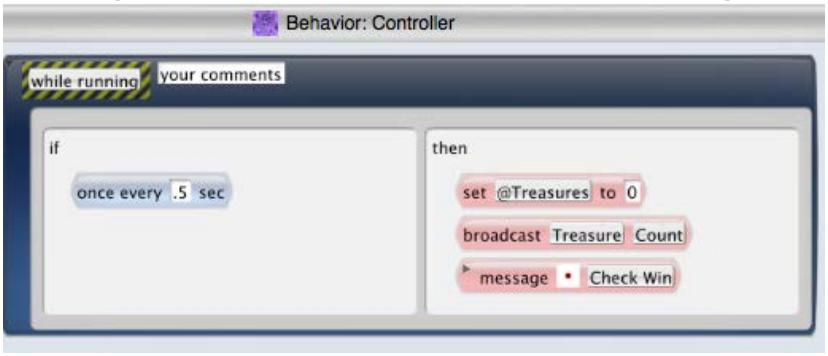
3-D Journey (Continued)

Definition: Computer scientists call the process of making a decision by sending a message to multiple recipients and checking responses **polling**.

Create the rule in the while running method of the Controller:
Pick a time interval for the **once every** condition
Add these three actions to the same rule:

1. Set the value of the simulation property Treasures to zero. (this is like the teacher saying "hands down"). **Note you must write @Treasures in the set action!**
2. Use broadcast to ask all the treasure agents to evaluate the rules in their Count methods if they are still on the World.
3. Send me (the Controller) a message to evaluate the rules in my Check Win method to see if the Traveler has collected all the treasures and won the game.

The single rule in the Controller's **while running** method should look like this:



Click on the +Method button below the Controller's rules. This method box will appear in the Controller:

Click on the word "Untitled" in the upper left corner and choose the same name that you entered the message action from the rule in the Controller's **while running** method.

3-D Journey (Continued)

Make the rule for the Check Win method:

1. Drag in the **test** action and use it to check whether the value of the simulation property Treasures equals zero because all the treasure agents have been collected by the Traveler. **Note you must write @Treasures in the test action!**
2. If the treasure agents are all gone, do the win actions. Remember to **stop** the simulation or **reload** the World to end the game!

Treasure behavior changes: There are two behavior changes required for the treasure agent.

- The first step is to have the treasure be collected by the traveler. We can simulate this by **erasing** the treasure agent when the Traveler moves on top of it.

- The second behavior change for the treasure agent is to respond to the Controller's broadcast by evaluating the rule in its **Count** method, which updates the Treasures simulation property.

This second change is in the form of a separate method; it is not part of the continually running "While Running" method, since it only runs when called by the controller agent.

During **Count**, each remaining treasure agent will increase (or **increment**) the value of the Treasures simulation property. If no treasure agents remain in the World, then the value of the Treasures property will be zero, which the controller agent will detect and declare the game won.

3-D Journey (Continued)

Before you test this, check your world. In the picture, we have placed **one** Controller tile in the right corner of the World.

In addition, we have placed several additional treasures on the worksheet, so that the traveler must "collect" – that is, move on top of each of them in order to win the game.

You can also add more Chasers to make the game more difficult!

Here is the behavior for the Treasure Agent:

The rule in the while running method erases the Treasure agent when a Traveler is on top of it.

The action in the Count method adds one to the value of the Treasures simulation property. **Note you must write @Treasures in the set action!**

The Bigger Picture: Communication between Agents

Polling introduces a technique that allows agents to create a complex behavior by cooperating: a particular set of conditions cause one type of agent to send a message to another type of agent to do a named method that contains a special set of rules. Count was the special method in the polling example.

This type of communication between different types of agents can be used to create interesting games. For example, if PacMan eats a power pill, then PacMan can broadcast a message to all ghosts to “Get_Scared”. The Get_Scared method can change the ghosts’ appearance so that they look different as they run away instead of chasing PacMan. Or the Traveler in Journey can fire ice arrows at Chasers. When an ice arrow hits a Chaser, it sends the Chaser agent a message that makes it freeze if it is unfrozen. Frozen chasers cannot move so the Traveler can collect the treasures without being caught by the Chaser.

You learned polling so that you would understand how to make different types of agents communicate. But there is usually more than one solution to a programming problem so now you have seen an alternate way to keep track of the number of any kind of agent.

When the goal is just to count up the number of agents and stick the value in a simulation property, AgentCubes Online has a simpler method for the controller to do this:

3-D Journey (Continued)

These two actions have been deleted from the controller's while running method:

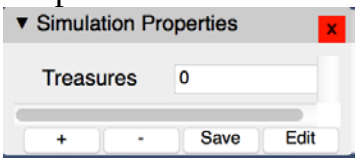
And replaced by this action:

The set action contains a specialized communication between the controller and the Treasure agents, the **agents_of_type("Treasure")** message, which makes the Treasures count themselves without the need for us to code a separate count method.

Student Handout: Troubleshooting Guide for Polling and Broadcast

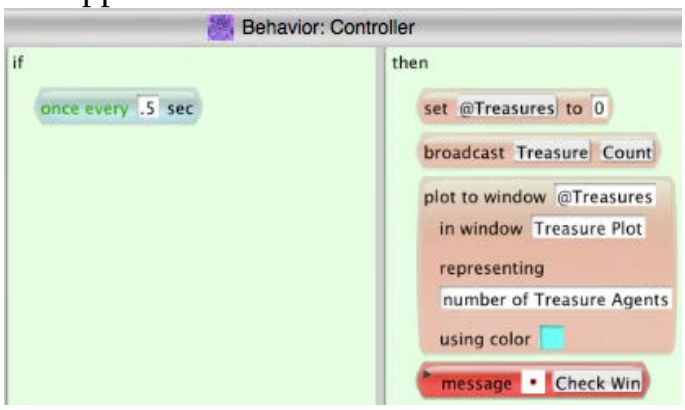
Make a quick check on how many Treasures are in the World:

Click on the AgentCubes Online gear button  and select "Show simulation Properties". This window will appear:



The correct number of Treasures will not appear in this window until you have single-stepped (click on the black triangle next to the stop and go buttons) or briefly run the game. If your programming is correct, the value of Treasures will decrease by 1 each time your Traveler collects (erases) a Treasure. When the value of Treasures is equal to 0, you should win the game.

More detailed troubleshooting: To determine what is happening in your game, it is helpful to look at how the simulation property changes over time. Add the **plot to window** action to the rule in the Controller's **while running** method. Fill it out as it appears below:

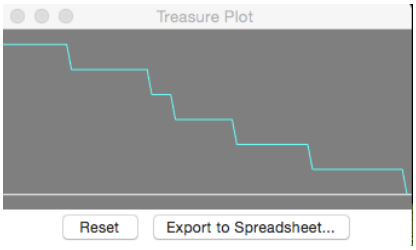


In the **plot to window** action, you must name the simulation property to be plotted (Treasures), name the window where it will appear (Treasure Plot), say what it represents (number of Treasure Agents) and pick the color of the line that will appear on the graph. **Note that you must put "@" before the Treasures in the plot to window action!**

The Treasure Plot window will appear as soon as you run the game. Move the Treasure Plot window somewhere where you can watch it while you run the game.

3-D Journey (Continued)

In this window, you will see a graph that shows you what's happening 'behind the scenes' while you play the game.



This information will help you determine where a mistake may be. For example, if the number of treasures never goes above 0, there is a problem with the method Count or the broadcast. If the number of treasures goes to zero but the game doesn't end, there is a problem with the game ending rule in the Controller.

End of Unit Review Sheet - Journey

A) The main computational thinking patterns we reviewed were:

1) **Cursor Control**: intentionally moving an agent.
    a. Using keyboard keys to move an agent.
    b. Example is moving the Traveler.

2) **Absorb**: deleting agents on the screen.
    a. Use the "Erase" action in AgentCubes Online.
    b. Examples are erasing the treasure agents.

3) **Collision**: when 2 agents collide (run into each other).
    a. Use the "See" condition
    b. Use the "Stacked" condition, OR
    c. Use the "Next to" condition.
    d. Examples are the collecting treasures and winning the game.

B) The main NEW computational thinking patterns we learned were:

1) **Diffusion**: emitting the scent (smell) of an agent. We used an agent attribute (S) on the agent with the smell, and we diffuse the smell by diffusing the attribute using the average of the 4 smells around it; s = (s[left]+s[right]+s[up]+s[down])*.25.

2) **Hill Climbing**: following the strongest scent. It only works if there is diffusion done with it, so they go hand in hand. Example is the method we created on the chaser to move towards the highest value of the scent "s" around him.

3) **Polling**: is when an agent "shouts out" (broadcasts) to all agents of a certain type requesting them to execute a specific method in response and perhaps change a global variable so that the originator agent can make a decision and take action. Example is the broadcast by the Controller of the method "Count" to the treasure agents in order to discover whether the game should end.

C) Other concepts we covered in AgentCubes Online are:

1) Troubleshooting the simulation, and considering rule order.
2) Using sounds and messages in the game.
3) Timing our actions using the "Once every" condition.

Student Challenge 1: Adding Levels

Before your start this challenge:

You must have a complete basic journey game with a Traveler who wins if s/he reaches the treasure and Chasers who chase the Traveler using a hill climbing search. The Traveler loses if a Chaser gets too close. The worksheet should have walls that the Traveler and Chasers cannot cross.

Levels

The Next Level

Build additional levels to increase the challenge!

Description of the Challenge:

- Your Traveler must shoot ice arrows up in all four directions (up, down, left and right).

Adding Levels to Your Game:

Now that you have made your project more like a real game with Chasers that really chase the Traveler and multiple treasures that must be collected to win, it is fun to make several Worlds so that your player can try to win multiple levels.

How can you make one World more challenging than another?

1. Thank about the arrangement of the Walls. Is it easier or harder for the Traveler to escape the Chasers in a more open maze with fewer walls?
2. Think about the number of Treasures. Is it easier or harder for the Traveler to win when s/he must collect a larger number of Treasures?
3. Think about the number of Chasers. What number of Chasers would make it harder but not absolutely impossible for your Traveler to win?

How do you make your Traveler move automatically from one level to another?

- AgentCubes Online has a condition that checks which World the agent is in right now 

and an action that lets the agent switch Worlds 

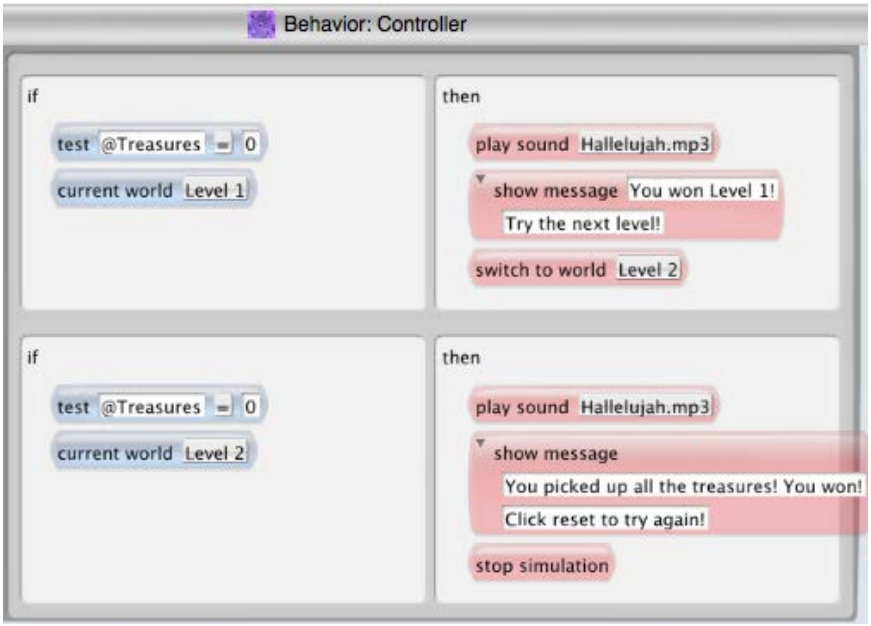
. The example Worlds were named “Level 1” and “Level 2” when they were created.

Where would you put rules that use these actions?

3-D Journey (Continued)

- Think about when the player should switch levels: not in the middle of exploring a World but after winning a particular level.

Go to the Controller and replace the single win rule in Check Win with these 2 rules (or more if you have more than two Worlds):



Test your Worlds on your friends! How many Levels can they win?

Student Challenge 2:

Ice Arrows Challenge

Before your start this challenge:

You must have a complete basic journey game with a Traveler who wins if s/he reaches the treasure and Chasers who chase the Traveler using a hill climbing search. The Traveler loses if a Chaser gets too close. The worksheet should have walls that the Traveler and Chasers cannot cross.

Description of the Challenge:

- Your Traveler must shoot ice arrows up in all four directions (up, down, left and right).
- A Chaser hit by a moving ice arrow freezes and cannot move.
- A frozen Chaser hit by a moving ice arrow unfreezes and can move again.
- Ice arrows should not go through walls or stack up in piles.

Design Activity:

In the description above, circle nouns to identify the agents and underline the verbs to identify actions associated with each agent. Mark adjectives to identify new shapes for an agent.

Create new agent: ice arrow

- Make it be an inflatable icon so you can draw your own picture.
- The picture for the ice arrow agent may face in any of the four directions.
- The point of the arrow should be a different color from the tail so that you can easily recognize which arrow you are seeing in the tiny pictures in the conditions.
- After you have drawn the first arrow shape, select the ice arrow agent and click on the +Shape button at the lower left corner of the AgentCubes Online window so you can draw an additional shape for the basic ice arrow.
- Draw 4 ice arrow shapes that so that the shapes face upwards, downwards, left and right.
- The ice arrow's shape stores its **Direction**. We can tell which way an ice arrow should move by checking its shape. The image saves the direction instead of an agent attribute.

Create a second shape for the Chaser: a frozen Chaser

- Select your chaser agent and click on the **+Shape** to create the frozen Chaser
- Make sure the frozen Chaser looks different enough that you can see it in the tiny pictures in the conditions.
- The Chaser's picture stores its **state**: frozen or unfrozen.

3-D Journey (Continued)

Traveler Design Challenges

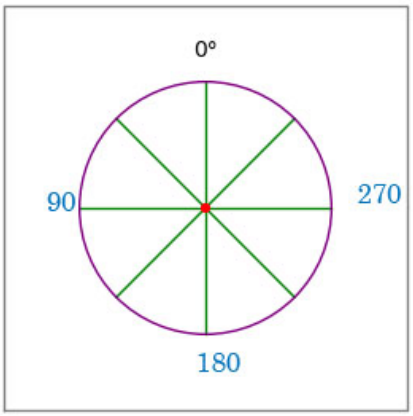
How do you know which ice arrow the Traveler should shoot?
The Traveler should shoot an ice arrow in the direction that s/he is facing.

How do you know which way the Traveler is facing?
The Traveler must have an Agent Attribute (or local variable) called Direction which keeps track of which way the Traveler is facing.
Initialize the Direction Agent Attribute in a when-creating-new-agent method.
When an arrow key is typed, set the Direction attribute and rotate the Traveler to face in that direction of the arrow key before the Traveler moves.

How can you avoid putting 4 rules for the 4 different ice arrows in the Traveler's while running method?
When the space bar is typed, make the Traveler send itself a message to do a method called Shoot Arrow that will shoot an arrow in the direction the Traveler is facing.

How are Directions named in AgentCubes Online?

Directions must be named using degrees on a circle as in the picture below.

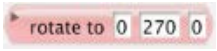


Up arrow key = move in 0 degrees direction
Left arrow key = move in 90 degrees direction
Down arrow key = move in 180 degrees direction
Right arrow key = move in 270 degrees direction

Traveler Rules

How to Create and Initialize the Traveler's Direction variable

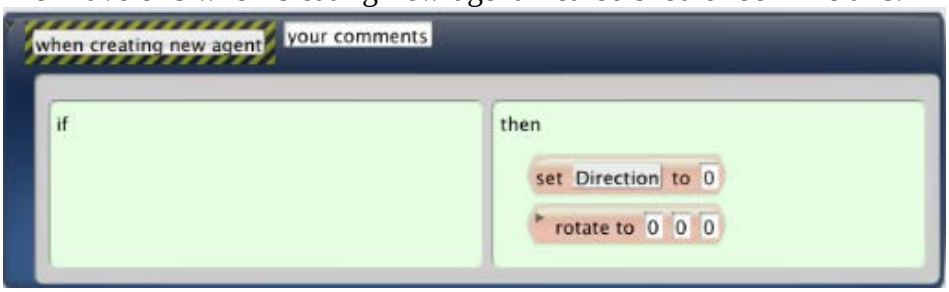
1. Click on the +Method button
2. Click on the word on in the method's black and yellow tag and change it to "when-creating-new-agent"
3. Drag in the set action 

and make a new variable name, Direction.
4. Set Direction to 0.
5. Drag in the 

action and set all 3 numbers to 0.
6. Erase the Traveler and redraw the Traveler on the World, then save it.

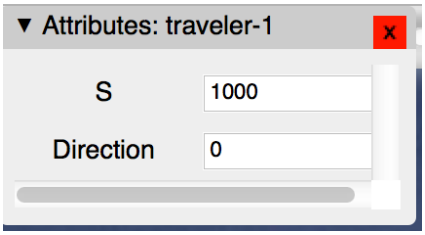
3-D Journey (Continued)

The Traveler's when-creating-new-agent method should look like this:



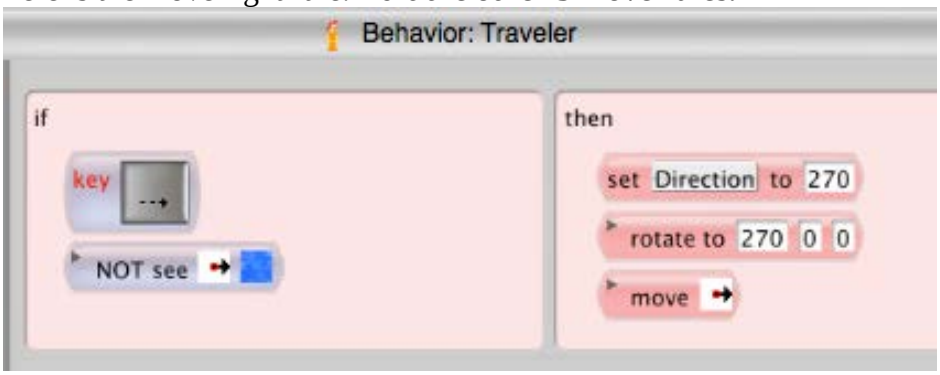
Test the value of the Traveler's Direction attribute

1. Double click on the Traveler with the big Arrow tool.
2. You should see this:



Update the Traveler's move rules

Add the **set** and **rotate to** actions to the Traveler's move rules so that the Traveler's Direction attribute changes as the player types the different arrow keys.
Here is the move right rule. **Edit the other 3 move rules.**



Note: the number for the direction goes in the <u>first</u> box of the rotate to action!

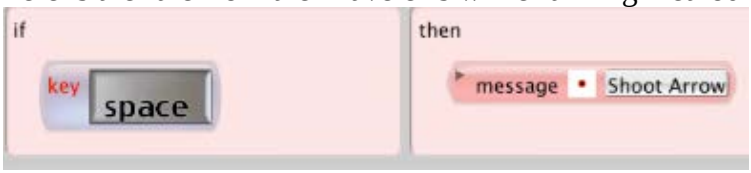
Test the Traveler's move rules to make sure the value of the Direction attribute matches the Traveler's movements

1. Double click on the Traveler with the big Arrow tool.
2. Type the right arrow key. Does Direction change to 270 when the Traveler moves right? Does the Traveler rotate to face right?
3. Check the other arrow keys. **Remember move up is 0, move left is 90 and move down is 180.**

3-D Journey (Continued)

Making the Traveler Shoot Arrows

Here is the rule from the Traveler's while running method that generates ice arrows:



Where should this rule appear in the while running method box?

- Above or below the win rule?
- Above or below the move rules?

Remember: special cases and less common events appear above default behavior like moving!

Here is the first rule in the Traveler's Shoot Arrow method:

Direction refers to the Traveler's agent attribute Direction, which keeps track of which way the Traveler is facing. **Remember 0 is up, 90 is left, 180 is down and 270 is right.**

Make 3 more rules like this one for the remaining 3 directions.

Test your code:

- Does your Traveler generate ice arrows in all 4 directions?
  If not, make sure that your ice arrow pictures and the arrows in the new actions match the directions.

Ice Arrow Rules

Ice Arrow Design Challenges

How do you know which direction the ice arrow should move?

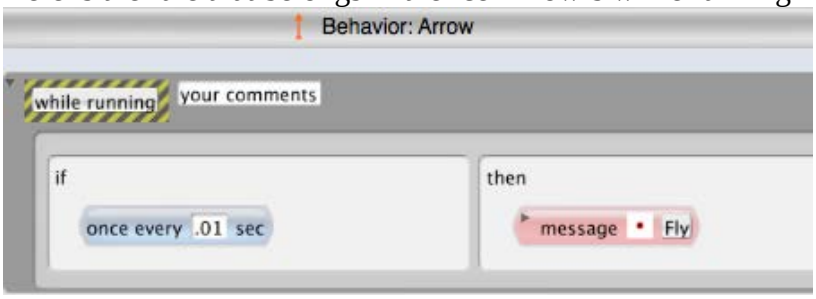
Ice Arrows should move whichever direction they are pointing.

How can you avoid putting 4 rules for the 4 different ice arrows in the Ice Arrow's while running method?

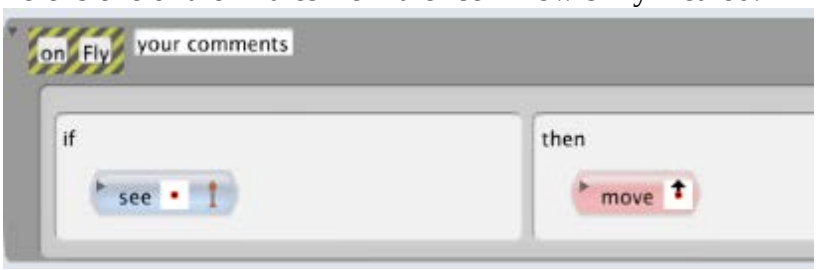
Make the Ice Arrow send itself a message to do a method called Fly that will make an ice arrow move in the direction it points.

3-D Journey (Continued)

Here is the rule that belongs in the Ice Arrow's while running method:



Here is one of the 4 rules from the Ice Arrow's Fly method:



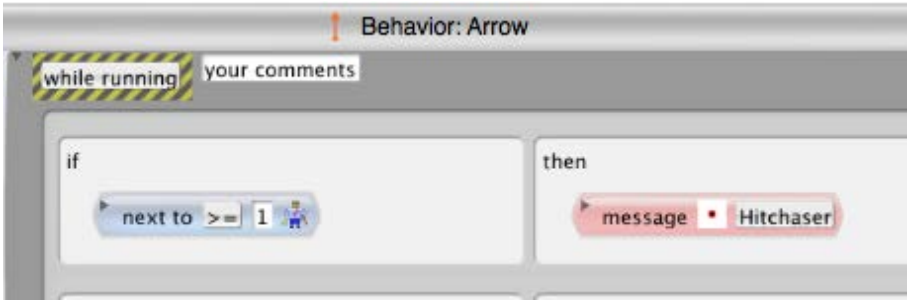
Add 3 more rules for the other 3 directions to the Fly method.

Test your code:

- Do your ice arrows move in all 4 directions?
  If not, make sure that your ice arrow picture and the move arrow match in each rule.

How do you make the point of the ice arrow hit the Chaser?
Only the point of the arrow can freeze or unfreeze the Chaser. It does not look convincing if the Chaser is hit by the side of an arrow. We need another method called HitChaser to see whether the arrow point will actually hit the Chaser.
What rules call HitChaser?
Use the **next to** condition to test whether the arrow is near a **frozen or unfrozen** Chaser, then use the HitChaser method to find out whether the Chaser is in front of the arrow point.

Here is the rule that calls HitChaser when the ice arrow is next to an ***unfrozen*** Chaser:



Does this rule belong above or below the rule that makes the ice arrow do it's Fly method?
It's a special case and special cases should always be higher than other rules!
Make a second rule that will call HitChaser when the ice arrow is next to a *frozen* Chaser.

3-D Journey (Continued)

What do Ice Arrow's HitChaser rules do?

1. Test the shape of the arrow.
2. Test whether there is a Chaser on the grid square that the arrow points at.
3. Send a message to the Chaser that it has been Hit.

Here is a rule from the HitChaser method:

Make 3 more rules like this for the other 3 ice arrow shapes.
Then make the same 4 rules for the Frozen Chaser, so that it unfreezes if hit by an arrow.

Chaser Rules

How does the Chaser Freeze and Unfreeze?
The Hit method freezes an unfrozen Chaser and unfreezes a frozen Chaser.

Here is the rule from the Chaser's Hit method that **freezes** a Chaser:

Note: The blue Chaser is frozen.

Make a second rule that will unfreeze a frozen Chaser.

3-D Journey (Continued)

Test your code:

- What happens when the Traveler shoots an ice arrow at the Chaser? You may need to temporarily delete the action in the Chaser's rules that makes it hill climb in search of the Traveler. It is much easier to test your ice arrows if the Chaser holds still!
- Does the ice arrow sit in front of the Traveler while the Traveler blinks back and forth between frozen and unfrozen shapes?
- The problem here is that the ice arrow keeps sending Hit messages to the Chaser so that the Chaser freezes and unfreezes over and over.
- How do you make the ice arrow stop sending messages to the Chaser?
- Erase it!
- Change the rules in the Ice Arrow's HitChaser method so that the ice arrow erases itself after it sends the Hit message to the Chaser.

Here is what the rules in HitChaser should look like now:

Test your code: does the Chaser stay frozen now until a 2nd ice arrow hits it?

One last step:

Do your ice arrows jump over walls?
Add rules to the Ice Arrow's Fly method so that ice arrows erase themselves if their points run into a wall.

3-D Journey (Continued)

Here is one of the rules:

Where should the rules about walls appear in the Ice Arrow's Fly method?

Special cases come before default behaviors!

1. Running into a wall happens less often so it is a special case and belongs above the regular fly rules.
2. The Ice Arrow's default behavior is to move forward so the call to the Fly method should be the last rule.

Where does this rule go? Look →

Make 3 more rules like this for the other 3 ice arrow shapes.

Test your program!

- Do your ice arrows stack up on the edges of your world?
- Put walls around the edges to absorb the ice arrows.
- Or create special ground agents that absorb ice arrows and put them around the edges of your world. If you choose this option, you will need 4 more rules in the Ice Arrow Fly method that look just like the wall rules but use the special ground agent instead of the wall. Put these rules just below the wall rules.
- Test that your Traveler is able to fire ice arrows in all 4 directions.
- The ice arrow should shoot in the Direction the Traveler is facing. If necessary, double click on the Traveler with the big arrow tool, to check which way the Traveler is facing and make sure that the ice arrow is fired the same direction. If there is a problem, go back through this tutorial and check that your rules exactly match the pictures.