

# TABLE OF CONTENTS

---

	<b>LEARNING GOALS</b>	<b>PAGE</b> <b>4</b>
	<b>ROBOT MASTER PROTOCOL AND ROOM SETUP</b>	<b>PAGE</b> <b>6</b>
	<b>LESSON LAYOUT</b>	<b>PAGE</b> <b>8</b>
<b>LESSON 1</b>	<b>I KNOW THAT!</b> Data Storage and Vision Recognition	<b>PAGE</b> <b>14</b>
<b>LESSON 2</b>	<b>GUESS WHAT!</b> Comparing Integers	<b>PAGE</b> <b>18</b>
<b>LESSON 3</b>	<b>LET ME QUIZ YOU!</b> Lists and Indices	<b>PAGE</b> <b>22</b>
<b>LESSON 4</b>	<b>ROCK, PAPER, SCISSORS!</b> Algorithms	<b>PAGE</b> <b>26</b>
<b>LESSON 5</b>	<b>LET'S PLAY!</b> Lists and Scrolling	<b>PAGE</b> <b>30</b>
	<b>BOX GLOSSARY</b>	<b>PAGE</b> <b>35</b>
	<b>PACKING INSTRUCTIONS</b>	<b>PAGE</b> <b>52</b>
	<b>TAKE HOME QUESTIONS</b>	<b>PAGE</b> <b>47</b>
	<b>VISION CARDS</b>	<b>PAGE</b> <b>49</b>



# LEARNING GOALS

## OVERVIEW

The overall focus of this module is to make students into real software developers! Successful developers and engineers must be independent learners. They must be able to deconstruct existing code, uncover the underlying structure and apply that abstraction to their own work. The learning goals detailed below break this target down into achievable pieces.

The names of these goals does not reflect their difficulty to achieve, it reflects a grouping of Bloom's five levels of cognitive behavior.

Knowledge is the lowest level of cognitive behavior. Knowledge requires recalling and applying previously learned information.

Analyze is the middle level of cognitive behavior. Analyzing requires understanding how any why something is true, rather than merely recalling facts and definitions.

Problem Solving and Evaluation is the highest level of cognitive behavior. Problem solving and evaluation requires synthesizing a myriad of information to devise a plan to solve a particular problem. Additionally it requires evaluating that procedure's effectiveness and revising it to become better.

## KNOWLEDGE GOALS

- ▶ Be able to have the robot recognize an image.
- ▶ Associate words with their meanings in a program.
- ▶ Read a Choregraphe program sequentially.
- ▶ Use loops, switch case, and other previously learned computer science concepts in own program.
- ▶ Understand and use new programming concepts in own program.

## ANALYZE GOALS

- ▶ Explain how and why a program works in writing.
- ▶ Write up a coherent multi-step plan.
- ▶ Write a clear reflection of work.
- ▶ Explain how and why new programming concepts work both verbally and in writing.

## EVALUATION GOALS

- ▶ Break down an example program into pieces of previously learned material and new material.
- ▶ Use context clues from previously learned CS content to deduce functionality of new material.
- ▶ Write a full explanation of the decontextualization process.

# LEARNING GOALS

- ▶ Identify and explain similarities in structure.
- ▶ Write clear reflection of own work and use this reflection to revise.
- ▶ Iterate the reflect and revise process until satisfied with result.

## COMMON CORE STANDARDS

**CCSS ELA-Literacy.CCRA.W.2** Write informative/explanatory texts to examine and convey complex ideas and information clearly and accurately through the effective selection, organization, and analysis of content.

**CCSS ELA-Literacy.CCRA.W.5** Develop and strengthen writing as needed by planning, revising, editing, rewriting, or trying a new approach.

**CCSS ELA-Literacy.CCRA.W.10** Write routinely over extended time frames (time for research, reflection, and revision) and shorter time frames (a single sitting or a day or two) for a range of tasks, purposes, and audiences.

**CCSS.ELA-LITERACY.W.5.2** Write informative/explanatory texts to examine a topic and convey ideas and information clearly.

**CCSS.ELA-LITERACY.W.5.2.A** Introduce a topic clearly, provide a general observation and focus, and group related information logically; include formatting (e.g., headings), illustrations, and multimedia when useful to aiding comprehension.

**CCSS.ELA-LITERACY.W.5.2.B** Develop the topic with facts, definitions, concrete details, quotations, or other information and examples related to the topic.

**CCSS.ELA-LITERACY.W.5.2.C** Link ideas within and across categories of information using words, phrases, and clauses (e.g., in contrast, especially).

**CCSS.ELA-LITERACY.W.5.2.D** Use precise language and domain-specific vocabulary to inform about or explain the topic.

**CCSS ELA-Literacy.RST.6-8.4** Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 6-8 texts and topics.

**CCSS MATH.PRACTICE.MP1** Make sense of problems and persevere in solving them.

**CCSS MATH.PRACTICE.MP2** Reason abstractly and quantitatively.

**CCSS MATH.PRACTICE.MP7** Look for and make use of structure

**CCSS MATH.PRACTICE.MP8** Look for and express regularity in repeated reasoning.

## REFERENCES

Bloom, B.S. (Ed.). Engelhart, M.D., Furst, E.J., Hill, W.H., Krathwohl, D.R. (1956). *Taxonomy of Educational Objectives, Handbook I: The Cognitive Domain*. New York: David McKay Co Inc.

# ROBOTMASTER PROTOCOL

As most classes are organized around a group of students and shared robots, getting participants used to sharing, and using a methodology for taking and giving over control of the robot is often useful for class management. Moreover, at any given point in time, only one computer can control to the robot. Failing to disconnect gracefully, will cause the robot to reject all other computer requests to connect.

We suggest taking turns with an answer and response methodology like is used in extreme sports and the military, so robot handover is treated as a team activity.

## THE "ROBOT MASTER" HANDOVER:

The person who holds the little robot is the Robot Master and the only one who is allowed to connect to the NAO robot.

- ▶ When you want to connect, you ask the Robot Master, "Can I connect to the robot?"
- ▶ The current Robot Master will disconnect from the NAO in Choregraphe
- ▶ The current Robot Master, will say, "I am disconnected!" and hand over the little robot.
- ▶ The receiver is the new Robot Master!
- ▶ Only after the little robot is received, can the Robot Master connect to the NAO robot with Choregraphe

While a student is connected to a NAO robot, no one else will be able to connect, as Choregraphe will return an error if connection is attempted.



# ROOM SETUP

## EACH CLUSTER COMES IN FIVE TRANSPORT CASES WITH THE FOLLOWING ITEMS:

- ▶ Three NAO Robots (named: Mickey, Minnie and Goofy)
- ▶ Sixteen laptops (15 for the students and one for the teacher, stickers can identify them)
- ▶ A wireless router
- ▶ One thumb-drive
- ▶ Three Robot-Master puppets
- ▶ Chargers for the robots and the laptops.

A proper packing of the transport case with eight laptops, chargers and a router.



## THE EVENING BEFORE THE CLASS:

- ▶ Charge all the laptops
- ▶ Make a backup of the thumb-drive with the students' stories.
- ▶ Charge the robots.

## CLASSROOM PREPARATION:

- ▶ Set up three island tables in the classroom, around each table, place 5 chairs for the students.
- ▶ Place a NAO on each table. (From the second lesson, try to keep the same robot for the same group of students. They become attached to their robot based on the name...)
- ▶ Plug in the NAO chargers, and use duct tape to secure them to the floor/table/wall.
- ▶ Plug in the router to the power, there's no need for an internet connection. Make sure it's in the same room with the robots and the laptops.
- ▶ Place five computers on each table around the robot. There's no need to plug in the laptops, the battery life should be good enough for more than 4 hours.
- ▶ Locate the Teacher's computer, (a sticker on the lid says "teacher") and keep it on your table, next to the projector/TV/big screen.
- ▶ Place a robot-master puppet on each table.
- ▶ Locate the thumb drive, and keep it next to your computer. At the beginning of the lesson, each student will copy his story to his/her computer.
- ▶ Make sure you did not forget this workbook!

# LESSON OUTLINE

## OVERVIEW

In this module, each lesson plan is the same, but the content changes. For this reason the only lesson plan you are provided is this one. There are reference sheets that are specific to each lesson that you should review, and work through, before each class.

There are five two-hour lessons in this module. These two-hour lessons are broken down into two one-hour sections. In each lesson, students will deconstruct an existing game on the robot (first hour) and use the new knowledge from deconstructing an existing game to create their own game (second hour). These stages are called Decontextualize and Recontextualize respectively.

Prior to running this course, you should make sure that the Module 3 vision recognition library is loaded on each robot. This library is what allows NAO to recognize the cards used in each lesson. To do this, follow the instructions on page 10 of this book.

**Aside:** When referring to the robot this book alternates between he and she. We encourage you to do the same in your classroom. This promotes gender equality in the classroom and helps encourage all students to participate.

## BEFORE CLASS

Read the reference page for the lesson and review this lesson plan. Put the example game program on each computer. The folders are titled Lesson\_1\_Example\_Game for lesson 1, etc. Choose one student from each group to be the Investigation Robot Master. Each student should get a chance to play this role throughout the class. Connect this computer to the robot to run the example game. This way the students can explore what the game does.

## DECONTEXTUALIZE

As students work in their student book, walk around the classroom and ask guiding questions to get students to engage more deeply with the content. Be sure to have students read instructions out loud.

- ▶ Demonstrates the example game for that lesson. **(5 minutes)**
- ▶ Have a student volunteer read through the questions on the **PLAY THE GAME!** page in the student book out loud to the class. Discuss the meaning of each question to ensure students understand what they are being asked. **(5 minutes)**
- ▶ Have students play the game and answer the questions. They should work as a group to do this investigation. Only the Investigation Robot Master should control the robot. **(10 minutes)**
- ▶ Discuss student answers as a class. **(5 minutes)**
- ▶ Have students then write full user-side walk through of the game on the **EXPLAIN THE GAME!** page in the student book. **(5 minutes)**
- ▶ Have a few students volunteer to read their write-ups to the class. It might be wise to pre-select

# LESSON OUTLINE

students as they are writing by reading a little over their shoulders and selecting write-ups that will spur discussion. (5 minutes)

- ▶ Students open the game file on their own computers and look through the code while answering questions on the **READ THE CODE!** page in the student book. (10 minutes)
- ▶ In each lesson there is one Black Box, a Choregraphe box that has an unknown function. These boxes are indicated by a ? icon. Students must identify these boxes and list the inputs, outputs and deduce the function of this box in the **WHAT DO THESE DO?** page in the student book. This is the central focus of learning in each lesson. Students must reason through the program to figure out what exactly these boxes do. They **MUST** understand the box before moving on to the next stage. (10 minutes)
- ▶ Discuss functions as a class. (5 minutes)

## RECONTEXTUALIZE

- ▶ Students brainstorm alternative uses of the new Black Box and write/draw their ideas on the **BRAINSTORM!** page in the student book. Give students **2-3 minutes** to brainstorm silently. Then have them discuss their ideas with a partner for **2-3 minutes** and finally discuss ideas as a group for **5 minutes**. This discussion method is called Think-Pair-Share.
- ▶ Have the students pick one of their ideas and construct a full plan on the **BUILD IT OUT!** page in their student book. (10 minutes)
- ▶ Have the students carry out the plan by programming it on their own computers. They should test whatever they can on the virtual robot before running it on the real robot. (20 minutes)
- ▶ When students finish making their program, have them complete the **REFLECT AND REVISE!** page in their student book. They should consistently reflect on their work and revise until they have a program that they are happy with (or they run out of time). (15 minutes)
- ▶ Have the students complete the **TODAY I LEARNED...** page in their student book. (5 minutes)

## AFTER CLASS

Save the students' work on the flash drive.

Pack up everything (Check the packing list on page 41 of this book).

Provide feedback to RobotLAB on how the lesson went at <http://content.robotslab.com/championslessonfeedback>





# VISION RECOGNITION

## A WORD ABOUT NAO'S VISION RECOGNITION

This curriculum relies heavily on NAO's vision recognition engine. To use vision recognition you must first make a database of images to recognize (we have done this for you with the cards on page 49 of this book). When you use the Vision Reco. box in choregraphe it takes 30 seconds or so to load the database. The robot will not do anything else during this time.

Some modifications have been made to this process for this particular curriculum. First, a vision recognition database has already been created for the cards. **No additional databases are necessary to implement this curriculum**, though you can learn how to create one on the next page. Additionally, the Vision Reco. box has been modified to wait 5 seconds between recognitions. This modification was made to make the games run more smoothly. You can modify this wait time using the process shown below.

## LOADING THE DATABASE

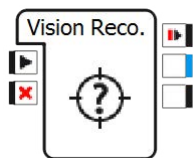
To load the Module 3 Vision recognition database onto your robot, follow steps 1-3 on page 7. Then, click the  button and navigate to module3.vrd file. Select, then press the  button to load the database onto the robot. Repeat for all robots.

## MODIFYING THE WAIT TIME

This process refers to the modified box available in the new Module 3 box library. If this library is not already loaded on your Choregraphe, you can add it by going to Edit → Preferences, pressing the plus sign next to the User's Box Libraries drop down menu and navigating to the appropriate file (module3-1.cbl. Press OK in the Preference pop-up box and a new Module 3 tab should appear in your box library.

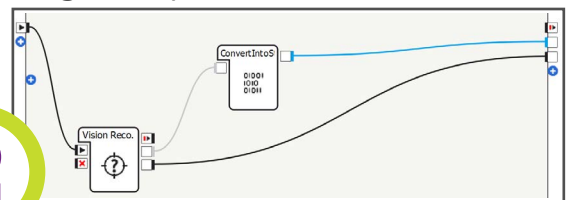
Drag the Vision Reco. box out from the new Module 3 box library. The box should appear as depicted below.

1



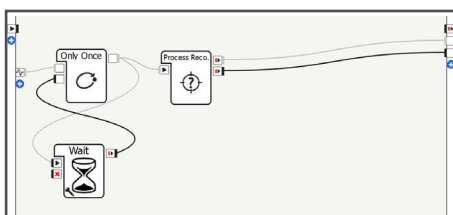
Double Click the box. You should see the following workspace:

2



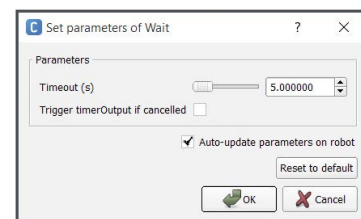
Double click the new Vision Reco. box. You should see the following workspace:

3



Press the wrench on the Wait box and adjust the time as desired.

4



# VISION RECOGNITION

## CREATING YOUR OWN VISION DATABASE

Let me reiterate, you DO NOT need to create your own vision recognition database to run this curriculum. This section is here merely to provide guidance for those curious about how it is done, or if you want to add cards and objects of your own.

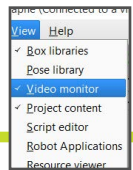
First we will show how to “learn” an object, then how to load a database onto the robot, then how to save that database and finally we will discuss which objects and images are good candidates for the robot to learn.

Connect to a real robot.

1

From the view dropdown menu, select Video Monitor

2



You should see a live stream from the robot's front camera. Additionally you should see the following menu bar:

3



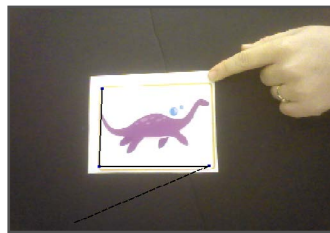
4

The menu bar options are play/pause, Learn, Import Vision Database, Export Current Database, New Database and Send Current Database To Robot.

To learn a new object, place the object in front of the robot so you can see it in the video monitor. Press the learn button and wait for Choregraphe to take the picture. When you have the picture, click around the object

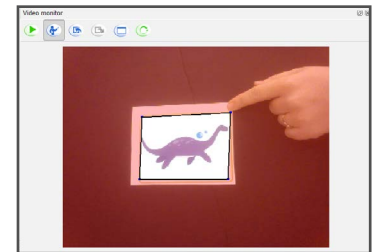
5

you want to learn.



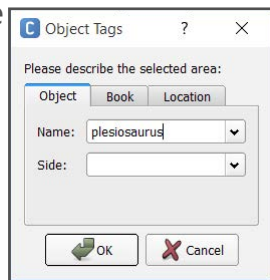
6

When you are done, the object should be outlined and everything outside the object should turn red.



An Object Tags pop-up box should appear automatically. This is where you type the name of the object. Be sure to note if you capitalize anything. To be safe, all words in this curriculum have been stored lowercase.

7



Continue learning objects until you have everything you want in the database. To push the database onto the robot, press the refresh button.

To save the database on your computer, press the export button.

8

To load a previously saved database, press the import button.

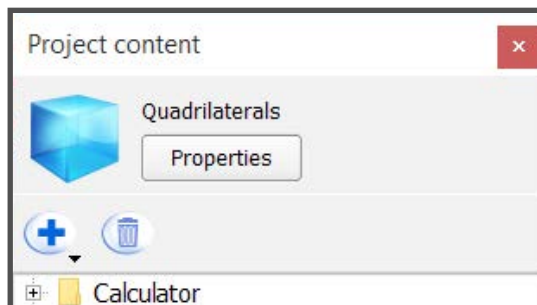
# PROJECT CONTENT

## WHAT IS PROJECT CONTENT?

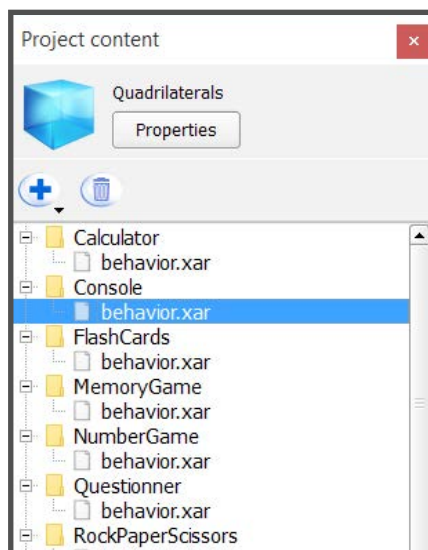
Another new attribute to this curriculum is using the project content view. This allows your students to have only one Choregraphe folder for the entire course, rather than a new folder for each lesson.

## HOW TO USE THE PROJECT CONTENT PALETTE

► From the menu bar go to View → Project Content. The following palette should appear:



- By default there is one behavior in your project. You can add a new one by pressing the plus button in the project content palette and entering a name. We recommend using a name without spaces.
- Once you have more than one behavior, you can switch between the two by double clicking the behavior.xar file within each behavior folder (highlighted below).

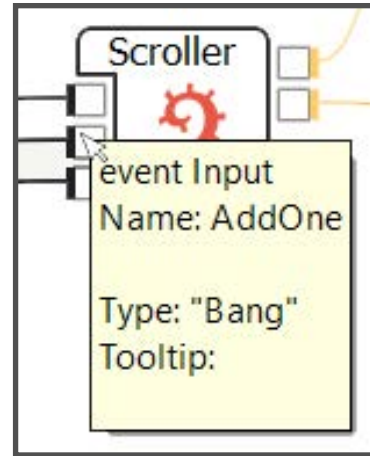


Your students should make a new behavior for each lesson rather than creating a new choregraphe file/folder for each lesson. This will help them keep track of their work. Additionally the final lesson will not work unless students have done this.

## TOOLTIPS

One useful feature of Choregraphe are the tooltips. If you hover over a box, input or output port a box will pop up to show you

1. The type of object
2. The name of the object
3. The type of the input (ports only)
4. The tooltip associated with the object.



Tooltips are useful for figuring out the black boxes and reminding yourself of what a box does.

# LESSON 1 REFERENCE

## BEFORE WE GET STARTED...

If you are reading this, we assume that you have already loaded the Module 3 vision recognition database on all robots. Additionally we assume that you have added the Module 3 box library to Choregraphe on all computers used in this lesson. If you have not done this, do so now. Follow the instructions on page 10 of this book.

## EXAMPLE GAME WALKTHROUGH

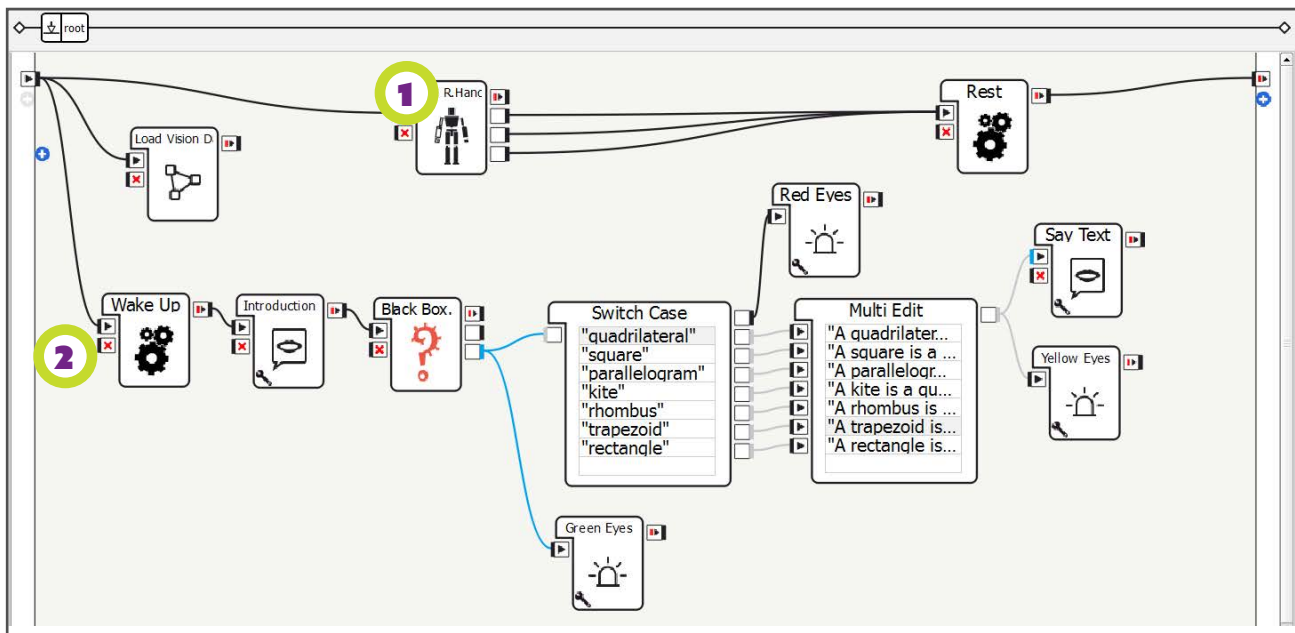
This program uses the quadrilateral cards located on page 49 of this book.

### WHAT THE USER EXPERIENCES

When you run the program, the robot will wait a few seconds before introducing the game. This wait is to allow some time for the vision recognition database to load on the robot. The robot will then say "Show me a quadrilateral card and I will tell you a little about that shape. Touch my head if you want me to stop".

If you then show the robot one of the quadrilateral cards, she will give the definition of that quadrilateral. If you show her a different card, her eyes will turn red and she won't say anything. You can repeat this process as many times as you want. To end the program, touch any of the robot's three head sensors.

### WHAT THE CODE DOES



# LESSON 1 REFERENCE

First the vision recognition database loads. This takes a few seconds. Meanwhile, the robot then wakes up, engaging the motors so that the robot moves while he talks (animated say boxes). Then two boxes are signaled.

- 1** The first is the tactile head box. This box makes the robot sense her head and if any of the three sensors is touched, the program flows to the rest box which then ends the program.
- 2** The second path leads to the introduction of the game, then to the vision recognition box (explained in more detail below). When this box is triggered, the robot begins looking for objects. When an object is recognized, the blue output port is signaled and the name of the image is passed to both the switch case box and the green eyes box. The next box is Multi Edit. The input ports correspond to each line within the box. If the third input port is signaled the third line of text ("A parallelogram is a quadrilateral with both pairs of opposite sides parallel" in this case) is sent through the output port. This leads to the say text so the robot will say the appropriate definition.

## BLACK BOX EXPLAINED: VISION RECO.

This box is stored in the Module 3 Box Library as Vision Reco.

### INPUTS

- ▶ onStart - begins the vision recognition process
- ▶ onStop - makes the robot stop looking for objects to recognize. Without signaling this port, the robot will never stop looking for objects. This is why the robot will recognize and define quadrilaterals more than once. Think of onStart as opening a door and onStop as closing that door.

### OUTPUTS

- ▶ onStopped - triggered when the box behavior finishes. In this case this port will only be triggered if the onStop input port was signaled.
- ▶ onPictureLabel - This port is signaled when the robot recognizes an object. The label given to that object is passed along the blue string.
- ▶ onNoPicture - This port is signaled when the robot has gone through its entire database, but does not see a picture.

### FUNCTIONALITY

This box uses the robot's camera to search for objects. When an object is recognized from the database, the robot passes that object's label through the onPictureLabel port. The robot will continue to search and pass on labels until the onStop input port is triggered.

## POSSIBLE USES OF BLACK BOX

This box could be used to make flashcards (like this program), tell a story about dinosaurs, quiz students on vocabulary definitions, play games etc.

# LESSON 1 TEACHING TIPS

## INTRODUCING THE LESSON

This is the first lesson in the module! So when you introduce the lesson, remember to introduce the course. This course is all about programming human-robot interaction through games. The interaction will primarily be through vision recognition. This first lesson focuses on programming association between the cards the robot recognizes and information about those cards.

## INTRODUCING THE GAME

This game is a study helper! The robot defines the different quadrilaterals that you show her. Watch and see how it works.

## LEADING THE DISCUSSION

In general, students should be the main contributors to discussion. As the teacher, you should ask questions to facilitate discussion and guide the discussion in the correct direction. Sample questions are provided below.

There are some concepts in this lesson, however, that students will struggle with and may need extra guidance to understand fully. In particular, we anticipate students having difficulty understanding what the Load Vision Database box does. Without this box, the vision database would be loaded when the black box is signaled. This would stall the game and make it less fun to play. It may be worth while showing the students what happens if you remove this box from the program.

Additionally students may not have seen the Multi Edit box before. This box is useful because it acts as a lot of text edit boxes, just all put int one. If the fifth input port is signaled, the fifth line of text in the box is sent through the box output port.

Additionally, during the brainstorm session, students will come up with ideas that they cannot yet implement. Their ideas may even be beyond the scope of this course. We recommend letting students' imaginations run wild during brain storming, then, before they start fleshing out their ideas, have them choose a portion of their idea that they can implement. In future lessons they can build on these projects if they so choose. Help them understand what is feasible and what is not. The memory game included as a demo is not feasible at this point.

## QUESTIONS TO PROMPT STUDENT LEARNING

### DECONSTRUCTION QUESTIONS

- ▶ Break it down! What happens if the Tactile Head box is signaled?
- ▶ What does the blue color mean?
- ▶ What information is passed through the blue thread?
- ▶ Which cards work with this program?

# LESSON 1 TEACHING TIPS

- ▶ Could we show the robot a dinosaur card? What would happen if we did?
- ▶ How do you stop the program once it is running?
- ▶ Where in the program does it tell the robot to do that?
- ▶ Is there any other way to stop the program?
- ▶ What happens if you double click the Black Box?
- ▶ Have you tried hovering over the boxes, input ports or output ports to read the tooltips?
- ▶ Why is the Black Box signaled at the same time as the wait and introduction thread?

## RECONSTRUCTION QUESTIONS

- ▶ What do you want to happen first in your game? Then what?
- ▶ What cards are you going to use in your game?
- ▶ What do you want the robot to say when she recognizes this card?
- ▶ Are you planning any custom movements, sounds or LEDs?



# LESSON 2 REFERENCE

## EXAMPLE GAME WALKTHROUGH

This program uses the number cards located on page 61 of this book.

### WHAT THE USER EXPERIENCES

When you run the program, the robot will wait a few seconds before introducing the game. This wait is to allow some time for the vision recognition database to load on the robot. The robot will then say "I'm thinking of a number between 1 and 10. Try to guess what it is!".

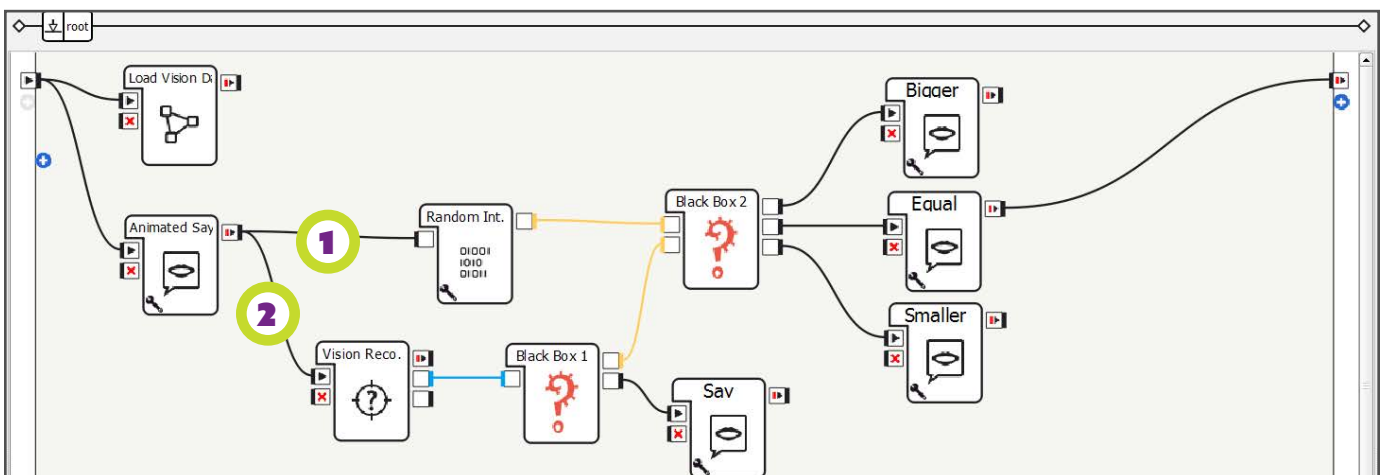
If you then show the robot one of number cards the robot will tell you if the number she is thinking of is bigger, smaller or if you got it right. The game continues until you guess the correct number.

### WHAT THE CODE DOES

First the vision recognition database loads. This takes a few seconds. When the database is loaded, the Animated Say box is signaled and the robot introduces the game. From there the path splits into two branches

- 1 The robot randomly selects a number from 1 to 10. This number is then passed to black box 2.
- 2 The robot simultaneously looks for images to recognize. When it recognizes an image, it feeds the name to black box 2, which attempts to convert the label into a number. If the recognized image is not a number, the robot says "Sorry, I didn't recognize a number. Show me again." If a number was recognized this box will convert the text label into an int type and pass this to black box 2.

From there, the robot compares the user selected number to her randomly generated number and responds accordingly. These responses are in the three Animated Say boxes at the end of the program. Only one of the "correct" box is linked to the universal stop port on the right of the workspace. This way, the program will continue until the user guesses the correct number.



# LESSON 2 REFERENCE

## BLACK BOX 1 EXPLAINED: STRING TO INT

This box is stored in the Module 3 Box Library as Vision Reco.

### INPUTS

- ▶ input - the text to be converted.

### OUTPUTS

- ▶ output - the integer converted from the string.
- ▶ notNumber - the string input was not a number.

### FUNCTIONALITY

This box converts the text given by the vision reco box to a number. If it cannot do the conversion (say the robot recognized "red monster") notNumber is signaled. Otherwise the text will successfully be converted to an integer.

## BLACK BOX 2 EXPLAINED: COMPARISON

This box is stored in the Module 3 Box Library as Vision Reco.

### INPUTS

- ▶ first vale - the value on the left of the operator.
- ▶ second value - the value on the right of the operator

### OUTPUTS

- ▶ greater\_than - the first value is greater than the second value.
- ▶ equal\_to - the two values are equal.
- ▶ less\_than - the first value is less than the second value.

### FUNCTIONALITY

This box takes in two values, compares them and splits the path based on how they compare.

## POSSIBLE USES OF BLACK BOXES

String to Int is necessary in all number based games.

Comparison could be used in war, race to 10, min, calculator, etc.

# LESSON 2 TEACHING TIPS

## INTRODUCING THE LESSON

This lesson is an introduction to all number-based games. Before running the program, it might be a good idea to refresh students' memory of the comparison operators (<, > and =). Some students remember this by thinking of the operators as an alligator who wants to eat the bigger number.

## INTRODUCING THE GAME

This game is a normal number guessing game. The robot will think of a number and you have to try to guess that number in as few tries as possible. Spend a few minutes having students play this game with each other before they play with the robot. This way they have a better understanding of, and recent experience with, the game.

## LEADING THE DISCUSSION

In general, students should be the main contributors to discussion. As the teacher, you should ask questions to facilitate discussion and guide the discussion in the correct direction. Sample questions are provided below.

There are some concepts in this lesson, however, that students will struggle with and may need extra guidance to understand fully. In particular, we believe that students will struggle with understanding why you need to convert a string type variable to a number. Shouldn't the computer just know that "9" means 9? You have to explain to them that it doesn't. There are many different variable types: strings (text), integers (counting numbers and their negatives), floats (decimals) and boolean (True, False). The robot needs to be told what type of data it is dealing with so that it deals with it appropriately. In Choregraphe there is one more, "bang" that signals boxes to begin their function. Most other programming languages do not have this type.

## QUESTIONS TO PROMPT STUDENT LEARNING

### DECONSTRUCTION QUESTIONS

- ▶ Break it down! What happens when the path splits?
- ▶ What would happen if you showed the robot stegosaurus instead of a number card?
- ▶ Can the robot pick a number not between 1 and 10?
- ▶ When does the program end?
- ▶ What are the three possible outcomes when you play this game?
- ▶ Play the number guessing game with a friend/peer. What do you think about when your friend guesses a number?

### RECONSTRUCTION QUESTIONS

- ▶ What do you want to happen first in your game? Then what?

# LESSON 2 TEACHING TIPS



- ▶ What cards are you going to use in your game?
- ▶ What do you want the robot to say when she recognizes this card?
- ▶ Are you planning any custom movements, sounds or LEDs?

# LESSON 3 REFERENCE

## EXAMPLE GAME WALKTHROUGH

This program uses the quadrilateral cards located on page 49 of this book.

### WHAT THE USER EXPERIENCES

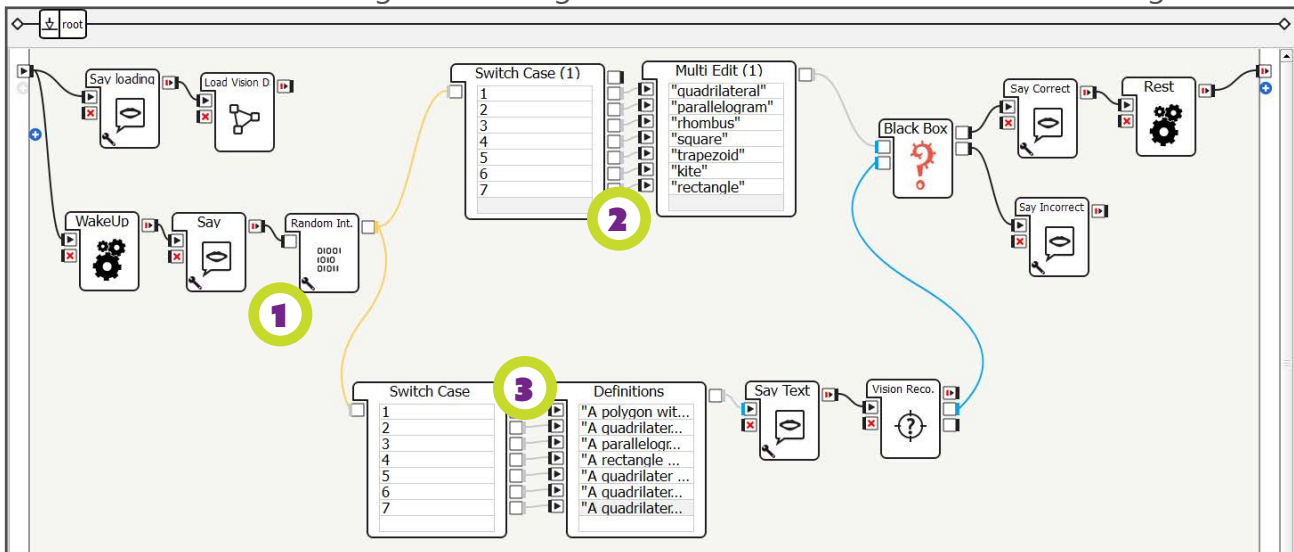
When you run the program, the robot will wait 3 seconds before introducing the game. This wait is to allow some time for the vision recognition database to load on the robot. The robot will then stand up and say "Show me the shape that best fits the following definition". The robot will then say a definition at random of one of the seven quadrilaterals. When you show the robot a card he will either say "Correct!" if you selected the correct card or "Sorry, try again" if you selected an incorrect card. If you got it correct the program will end, otherwise you have to try again until you get it right.

### WHAT THE CODE DOES

First the vision recognition database loads. This takes a few seconds. Meanwhile, the robot then wakes up, engaging the motors so that the robot moves while he talks (all the say boxes are animated say).

- 1 A random number is chosen between 1 and 7. This is because there are seven quadrilaterals in this game.
- 2 This number is sent to two switch case boxes. The switch case matches the input to a number and signals the appropriate output. On the top, this output signals the name of one of the quadrilaterals.
- 3 On the bottom, the switch case output signals one of the quadrilateral definitions. It is important that the definitions are in the same order as the labels in 2. This definition is then passed to a say text box so the NAO says the definition.

Next the robot waits to recognize an image with the Vision Reco. box. When it recognizes a card,



# LESSON 3 REFERENCE

the name of this card is passed to the black box.

The black box compares the two inputs and if they are the same that means you were correct, if they are different this means your guess was incorrect. If you get it right, the program is over. If you got it wrong, the program does not end. This leaves the Vision Reco. "door" open for you to try again.

## BLACK BOX 1 EXPLAINED: ENTRY IN LIST

### INPUTS

- ▶ List - takes in a list delimited by semicolons
- ▶ Index - takes in a number greater than 0 and less than the length of the list. The index is the number of the list entry to return.

### OUTPUTS

- ▶ Entry - The entry in the list that corresponds to the index.

### FUNCTIONALITY

This box picks an entry in a list based on the index given. For example if you have the list Apples;Oranges;Grapes;Pears and the index 2, Oranges would be passed through the Entry output port.

## BLACK BOX 2 EXPLAINED: STRING EQUALITY

### INPUTS

- ▶ string1 - takes in a string, or text.
- ▶ string2 - takes in a string.

### OUTPUTS

- ▶ equal - signaled if the two strings are the same.
- ▶ notequal - signaled if the two strings are different.

### FUNCTIONALITY

This box compares two bits of text. In the case of our program, it compares the name of the quadrilateral card you showed the robot, with the entry from the Labels list. If they match you got it correct, if they are different you are incorrect.

## POSSIBLE USES OF BLACK BOXES

Interactive story telling, study companion.

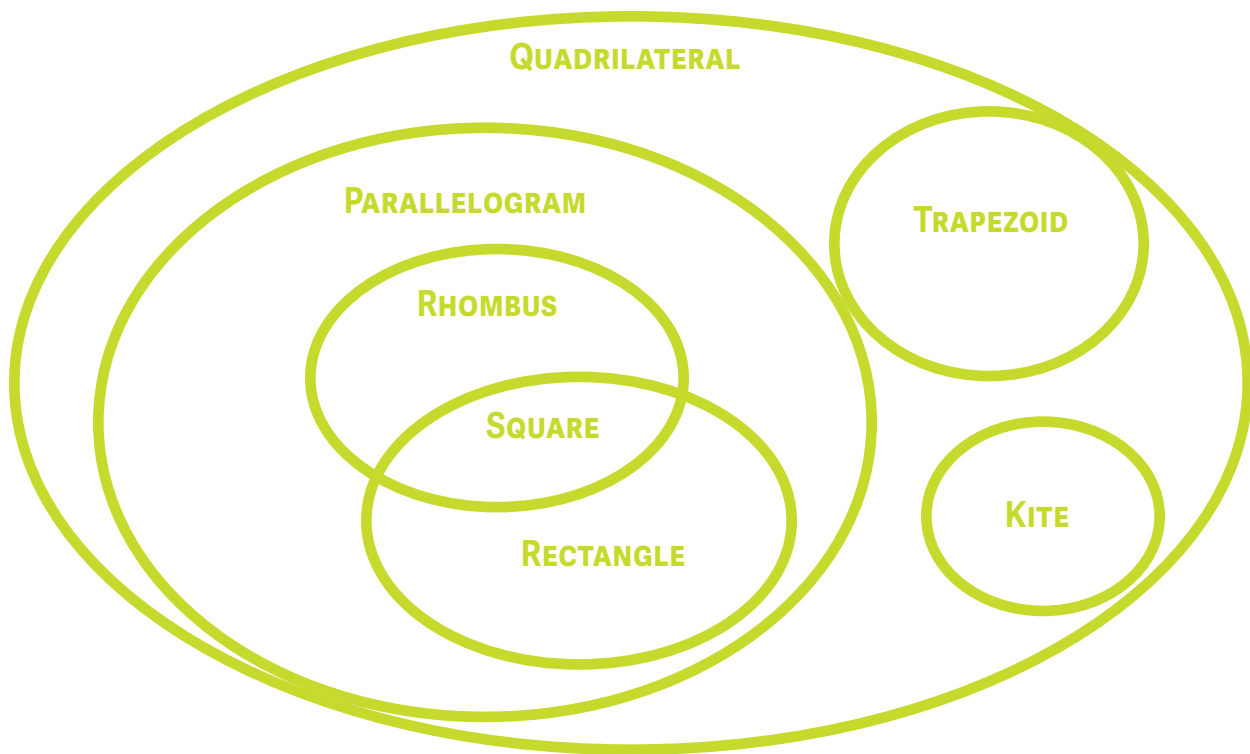
# LESSON 3 TEACHING TIPS

## INTRODUCING THE LESSON

This lesson builds on the game from lesson 1. In lesson 1 the robot was recognizing a card and matching it with the definition. In a sense you were quizzing the robot on the different quadrilaterals. This time the roles are reversed. The robot is quizzing you on the different quadrilaterals!

## INTRODUCING THE GAME

This game is a study helper! The robot quizzes you on the different quadrilaterals. He will say a definition and you have to pick the quadrilateral that best fits that definition. To help you keep them straight, here is a Venn diagram of quadrilaterals. If a quadrilateral is both a rhombus and a rectangle, it must be a square.



## LEADING THE DISCUSSION

In general, students should be the main contributors to discussion. As the teacher, you should ask questions to facilitate discussion and guide the discussion in the correct direction. Sample questions are provided below.

The flow of this program will be difficult for most students to get through. You have to help them walk through the program. Most importantly, help them understand the indexing of the two lists. The answers to a portion of the "Read The Code!" activity are given on the next page. If the random number selected is 3, the quadrilateral label is rhombus and the definitions is A parallelogram with all four sides congruent. Using as many of these examples as possible will help students understand this structure.

# LESSON 3 TEACHING TIPS

## QUESTIONS TO PROMPT STUDENT LEARNING

### DECONSTRUCTION QUESTIONS

- ▶ If the random number is 6, which label is selected? Which definition?
- ▶ What information is being passed into the Black Box?
- ▶ What does the blue color represent in choregraphe? Yellow?
- ▶ How does the program know if you got the correct card?
- ▶ What are the output ports of the vision recognition box?
- ▶ Why are there no spaces between the words in the labels text box? (this is because there are no spaces stored in the card vision database labels and when you check for equality, it checks character by character).

### RECONSTRUCTION QUESTIONS

- ▶ What do you want to happen first in your game? Then what?
- ▶ What cards are you going to use in your game?
- ▶ What do you want the robot to say when she recognizes this card?
- ▶ How are you using Black Box 1, the entry in list box?
- ▶ How are you using Black Box 2, the string equality box?
- ▶ Are you planning any custom movements, sounds or LEDs?

## READ THE CODE ANSWERS

### LABELS

1. quadrilateral;
2. parallelogram;
3. rhombus;
4. square;
5. trapezoid;
6. kite;
7. rectangle

### DEFINITIONS

1. A polygon with four sides;
2. A quadrilateral with both pairs of opposite sides parallel;
3. A parallelogram with all four sides congruent;
4. A quadrilateral with all four sides congruent and all four angles right angles;
5. A quadrilateral with exactly one pair of opposing sides parallel;
6. A quadrilateral with two pairs of opposing sides parallel;
7. A quadrilateral with all four angles right angles



# LESSON 4 REFERENCE

## EXAMPLE GAME WALKTHROUGH

This program uses the rock, paper, scissors cards located on page 54 of this book.

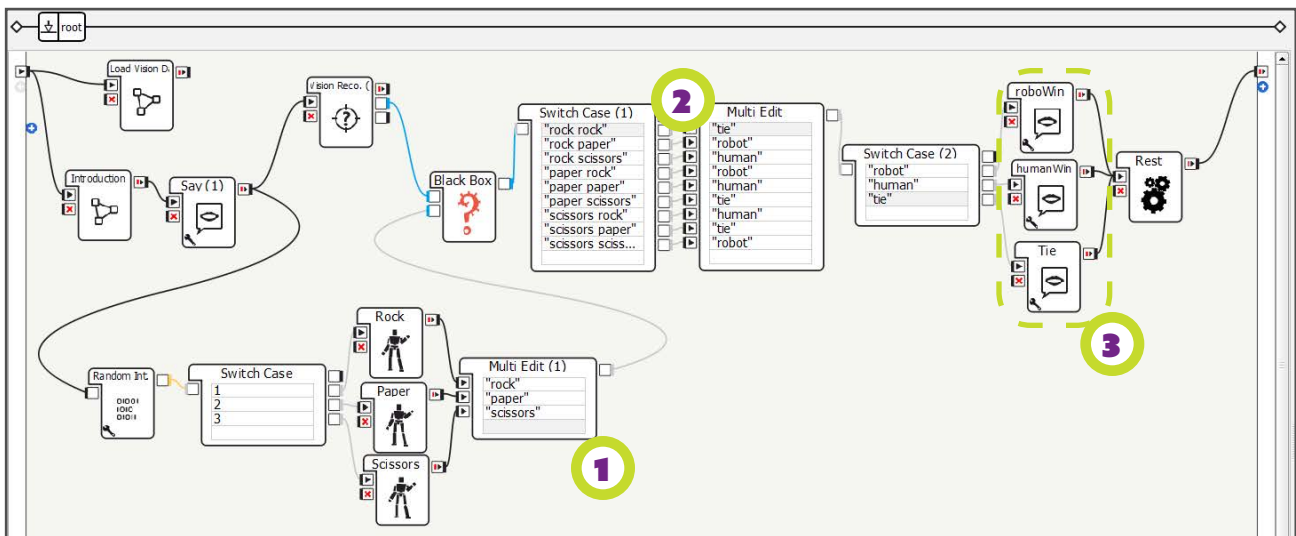
### WHAT THE USER EXPERIENCES

When you run the program, the robot will wait 3 seconds before introducing the game. This wait is to allow some time for the vision recognition database to load on the robot. The robot will then stand up and say "Let's play rock paper scissors. This is how I make them". She will then proceed to show you the hand gestures she uses for each option (rock, paper and scissors). The robot will then say "rock, paper scissors shoot". At this point the robot is expecting you to select one of the rock, paper, scissors cards and show it to her. She will select an action to do at random. When she recognizes your card, she will determine a winner. The program is now over.

### WHAT THE CODE DOES

First the vision recognition database loads. This takes a few seconds. Meanwhile, the robot then wakes up, engaging the motors so that the robot can make the rock, paper, scissors motions. The Introduction box is a diagram box containing Nao's whole introduction of the game. The next say box is where Nao says "Rock, paper, scissors shoot". After this the path splits in two directions.

- 1 In this path the robot selects a number at random from 1 to 3. The switch case splits the path based on the number. If the number is 1, the robot chooses rock, does the animation and signals the input of the Multi Edit box corresponding with "rock". The top path is Vision Reco. where the robot waits to recognize an image then passes it on to the black box.
- 2 The black box joins the two pieces of text together so if the vision reco gave the box "rock" and the robot chose "paper" the black box would output "rock paper". This text then gets passed to



# LESSON 4 REFERENCE

the switch case, which signals the multi edit and determines the winner. So "rock paper" connects to "robot" meaning that the robot wins.

- 3 These say boxes are where the robot either says "You win", "I win" or "tie". Then the program is over.

## BLACK BOX EXPLAINED: DECIDER

### INPUTS

- ▶ text1 - takes in text. In the example program, the text is the human's choice.
- ▶ text2 - takes in text. In the example program, the text is the robot's selection.
- ▶ rules - takes in a list of rules. Each rule is separated by a new line. Each entry in a rule follows the following format: player 1's selection,player2's selection,winner

### OUTPUTS

- ▶ text1text2 - outputs the concatenated text (with a space in between).

### FUNCTIONALITY

This takes two pieces of text and "glues" them together (concatenates).

## POSSIBLE USES OF BLACK BOX

Rock, Paper, Scissors with different dinosaurs/monsters. Rock, Paper, Scissors with more options or different rules (maybe rock always wins). War. Any two player selection game.

# LESSON 4 TEACHING TIPS

## INTRODUCING THE LESSON

In this lesson, students will investigate how to make NAO play rock, paper scissors. The learning goal through doing this is to have students make a systematic list of all possible outcomes (see Read The Code answers on next page).

## INTRODUCING THE GAME

This game is rock, paper, scissors. Most students should already know this game, but they may have forgotten. So start by introducing the game. There are two players and they simultaneously choose either rock paper or scissors. To choose rock, you make your hand into a fist. To choose paper you make your hand flat and to choose scissors you put your pointer finger and middle finger out in the shape of scissors. To determine who wins, follow these simple rules. Rock beats scissors, scissors beats paper and paper beats rock.

## LEADING THE DISCUSSION

In general, students should be the main contributors to discussion. As the teacher, you should ask questions to facilitate discussion and guide the discussion in the correct direction. Sample questions are provided below.

This program can be a little overwhelming as there is a lot going on. Reenforce the idea that even though it looks complicated, students can break down smaller pieces then build it back up to see what the program does.

Additionally, students might be overwhelmed by the multi edit box. Try to have them deduce which player is first in the list (i.e. human selection is first, then robot selection).

## QUESTIONS TO PROMPT STUDENT LEARNING

### DECONSTRUCTION QUESTIONS

- ▶ Double click the timeline boxes to see what is inside.
- ▶ What is the range of the random int box?
- ▶ What is inside the Say(1) box?
- ▶ Whose selection is first in the rules?
- ▶ How many rules are there?
- ▶ Why are there 9 rules?

### RECONSTRUCTION QUESTIONS

- ▶ What do you want to happen first in your game? Then what?
- ▶ What cards are you going to use in your game?

© RobotLAB 2015 By Ellen McCullagh

# LESSON 4 TEACHING TIPS

- ▶ What are the rules in your game?
- ▶ How do these rules translate to the lines you need to type into the rules text box?
- ▶ How are you using the decider box in your game?

## READ THE CODE ANSWERS

### WRITE OUT EACH RULE FROM THE TEXT BOX MARKED WITH

- ▶ rock rock = tie
- ▶ rock paper = robot
- ▶ rock scissors = human
- ▶ paper rock = human
- ▶ paper paper = tie
- ▶ paper scissors = robot
- ▶ scissors rock = robot
- ▶ scissors paper = human
- ▶ scissors scissors = tie

# LESSON 5 REFERENCE

## EXAMPLE GAME WALKTHROUGH

This program lets students combine all the games they have created thus far.

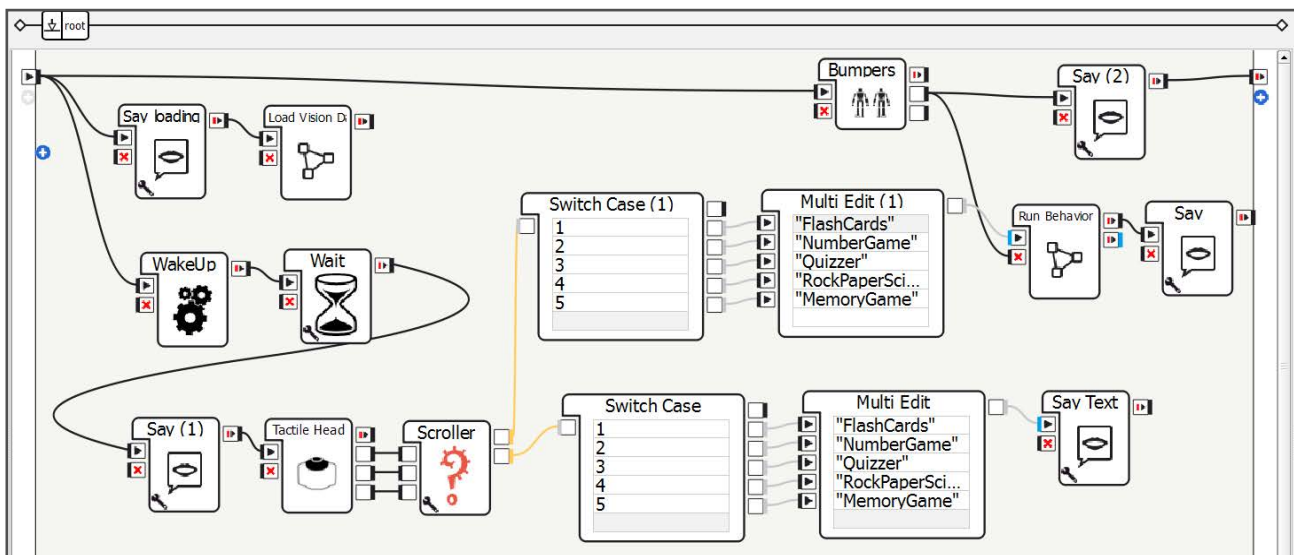
### WHAT THE USER EXPERIENCES

The robot will scroll through all the names of the programs students have created. Scrolling uses the front and rear head touch sensors. When they select the game they want they can press the middle head sensor and the game will start.

### WHAT THE CODE DOES

Three things happen off the bat here. First, the bumper sensor is signaled. This is the failsafe switch for the entire program. Also the vision database loads. Even though vision recognition is not used in this program, this program runs all the other games so if you load it here the other loads will go faster and the games will be more fun to play. Lastly, a wakeup box is signaled then a wait box. This is timed so that the robot waits for the vision database to load before moving on.

Then the robot says "select a game by pressing the buttons on my head". When you press the buttons on the robot's head, a count is established and changed depending on the button. If you press the front button, the count goes up one. If you press the back button the count goes down one. This



count does loop around (i.e. after 5 comes 1). All of this work is done in the scroller box.

Scroller has two outputs, one output is signaled every time the count changes. This is so that the robot can tell you what program you would select if you pressed the middle head button. The second (top) output is the one that is triggered when you make a selection.

If you select when the count is at 4, for example, the text "RockPaperScissors" will be given to the Run Behavior box from the Multi Edit box. Run Behavior finds the behavior with that given name and runs it. It is important that the list in this multi edit uses the **EXACT** titles of all of the behaviors in the particular project (Must be in project content). Check the Project Content palette ([more info on page 30 of this book](#)) to see the names of your behaviors.

# LESSON 5 REFERENCE

## BLACK BOX EXPLAINED: SCROLL

### INPUTS

- ▶ AddOne - adds one to the current value (initial value in wrench).
- ▶ Select - selects the current number
- ▶ SubtractOne - takes one away from the current value.

### OUTPUTS

- ▶ CurrentNumber - outputs the current number each time there is a change.
- ▶ NumberSelected - outputs the current number when the Select input port is signaled.

### FUNCTIONALITY

This box effectively allows you to scroll through a set of numbers. It does not stop at the top, rather it jumps back to the beginning. Similarly it doesn't stop at the bottom, it jumps back to the top. In other words, the numbers loop. This lets you scroll around and around the different project names.

## RUN BEHAVIOR BOX

### INPUTS

- ▶ onStart - Takes in a string representing the desired behavior.
- ▶ onStop - stops the behavior

### OUTPUTS

- ▶ onStopped - signal sent when behavior finishes.
- ▶ onError - outputs a string explaining the error when an error arises.

### FUNCTIONALITY

This box runs a behavior of a given name. Again this only works if that behavior is part of your project content.

# LESSON 5 TEACHING TIPS

## INTRODUCING THE LESSON

In this lesson, students will create a console that connects all of the games that they have made thus far together. This will not take the whole two hours. We recommend using the remaining time to have students perfect their games and then show them off to each other.

## INTRODUCING THE GAME

This game uses the tactile sensors to “scroll” through all the games in that particular project. Explain to students that this is like a playlist of music or a gaming console.

## LEADING THE DISCUSSION

In general, students should be the main contributors to discussion. As the teacher, you should ask questions to facilitate discussion and guide the discussion in the correct direction. Sample questions are provided below.

This program should be pretty quick for the students. They will most likely replicate the example without adding a ton of flourish. This is OK. They can spend the remaining time perfecting their previous games and showing them off to one another.

One thing they might struggle with is the syntax they need to use in the text box to list their program names. Here is what you need to know:

- ▶ All of the games have to be visible in the project content palette (page 12 of this book). If the projects are all in different files, have the students copy paste the boxes into a new menu file.
- ▶ The names of the programs must be typed identically to how they are typed in the project content palette. (You can right click to rename within the project content palette)

## QUESTIONS TO PROMPT STUDENT LEARNING

### DECONSTRUCTION QUESTIONS

- ▶ What does the Entry in List box do?
- ▶ What goes into the scroll box?
- ▶ Click on the wrench of the scroll box. What are the parameters?
- ▶ What variable type does the scroll box output? What does yellow symbolize?
- ▶ Why is the blue string looped back to the onStop port of the Tactile Head box?
- ▶ Why are there two entry in list boxes?
- ▶ What did the front/middle/back head sensor do in the program?
- ▶ Hover over the input and output ports of the scroll box. What are their names?

# LESSON 5 TEACHING TIPS

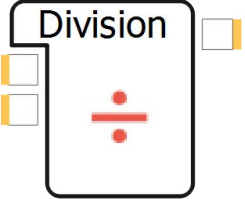
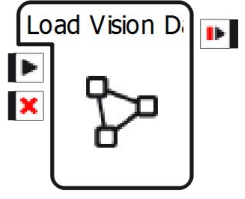
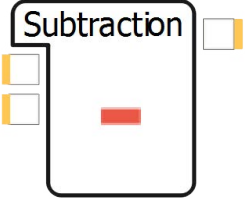
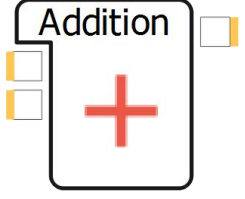
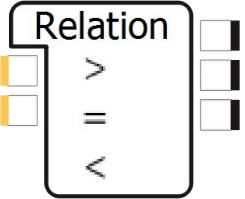
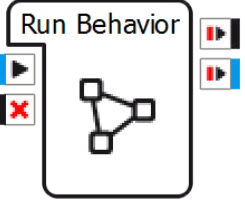
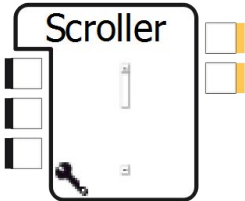
## RECONSTRUCTION QUESTIONS

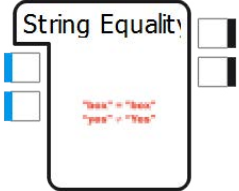
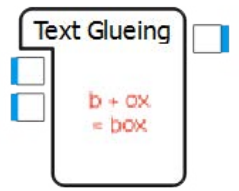
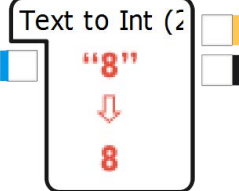
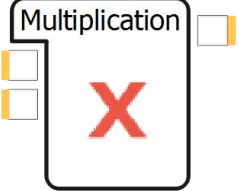
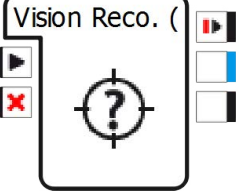
- ▶ Are all of your games in the same project?
- ▶ What are the names of your games?



# Box Glossary

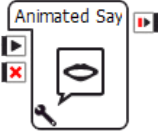
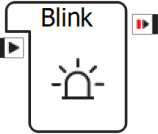
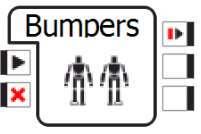

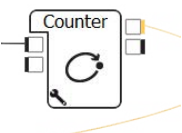
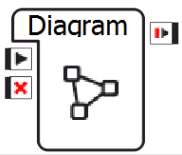

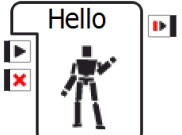
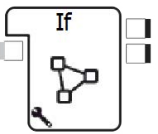
## MODULE 3 NEW BOXES

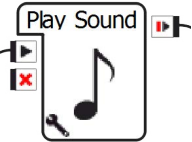
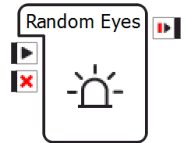
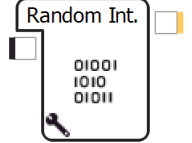
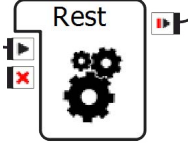
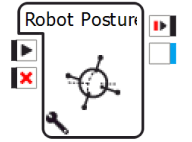
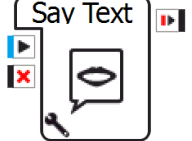
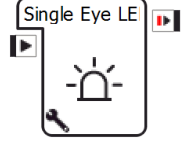


Box	NAME	FUNCTION
	<b>DIVIDE</b>	Divides the top number by the bottom number
	<b>LOAD VISION DATABASE</b>	Loads the vision recognition database currently on the robot.
	<b>MINUS</b>	Subtracts the bottom number from the top number
	<b>PLUS</b>	Adds both inputs.
	<b>RELATION</b>	Takes in two numbers and compares them. The top output fires if the top number is larger than the bottom, etc.
	<b>RUN BEHAVIOR</b>	Looks for a local file with the same name as the text input. If it finds one, it runs the behavior.
	<b>SCROLLER</b>	Scrolls through numbers. Top input increases the number, bottom input decreases the number. Middle input selects the number.

Box	NAME	FUNCTION
 <p>String Equality</p> <p>"Bob" = "Bob" "yes" = "Yes"</p>	<p><b>TEXT EQUALITY</b></p>	<p>Checks to see if two string-type variables are equal.</p>
 <p>Text Glueing</p> <p>b + ox = box</p>	<p><b>TEXT GLUING</b></p>	<p>Takes two bits of text and glues them together. Top input first.</p>
 <p>Text to Int (2)</p> <p>"8" ↓ 8</p>	<p><b>TEXT TO INT</b></p>	<p>Converts a string type variable to an integer. Top output fires if conversion can happen, bottom output fires if cannot convert.</p>
 <p>Multiplication</p> <p>X</p>	<p><b>TIMES</b></p>	<p>Multiplies the two numbers.</p>
 <p>Vision Reco. (</p> <p>?</p>	<p><b>VISION RECO</b></p>	<p>Recognizes an image and outputs the name of that image from the blue output port.</p>

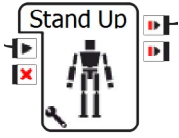
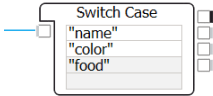
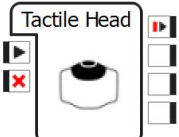
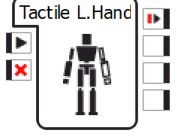
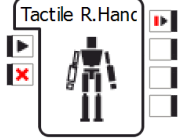
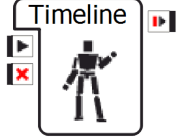


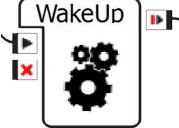
# Box Glossary

OLD BOXES

Box	NAME	FUNCTION
	<b>ANIMATED SAY</b>	Makes the NAO speak while he moves his arms and head
	<b>BLINK</b>	Makes the robot blink once
	<b>BUMPERS</b>	Waits for one, or both, of the bumpers to be pressed
	<b>COLOR EDIT</b>	Sets a color and codes it as a number
	<b>COUNTER</b>	Makes NAO repeat an action a certain number of times
	<b>DIAGRAM</b>	A custom box to group other boxes
	<b>EYE LEDs</b>	Sets the LED color of both eyes
	<b>HELLO</b>	Makes NAO wave hello (He does not say hello)
	<b>IF</b>	Checks to see if the input is equal to a preset value

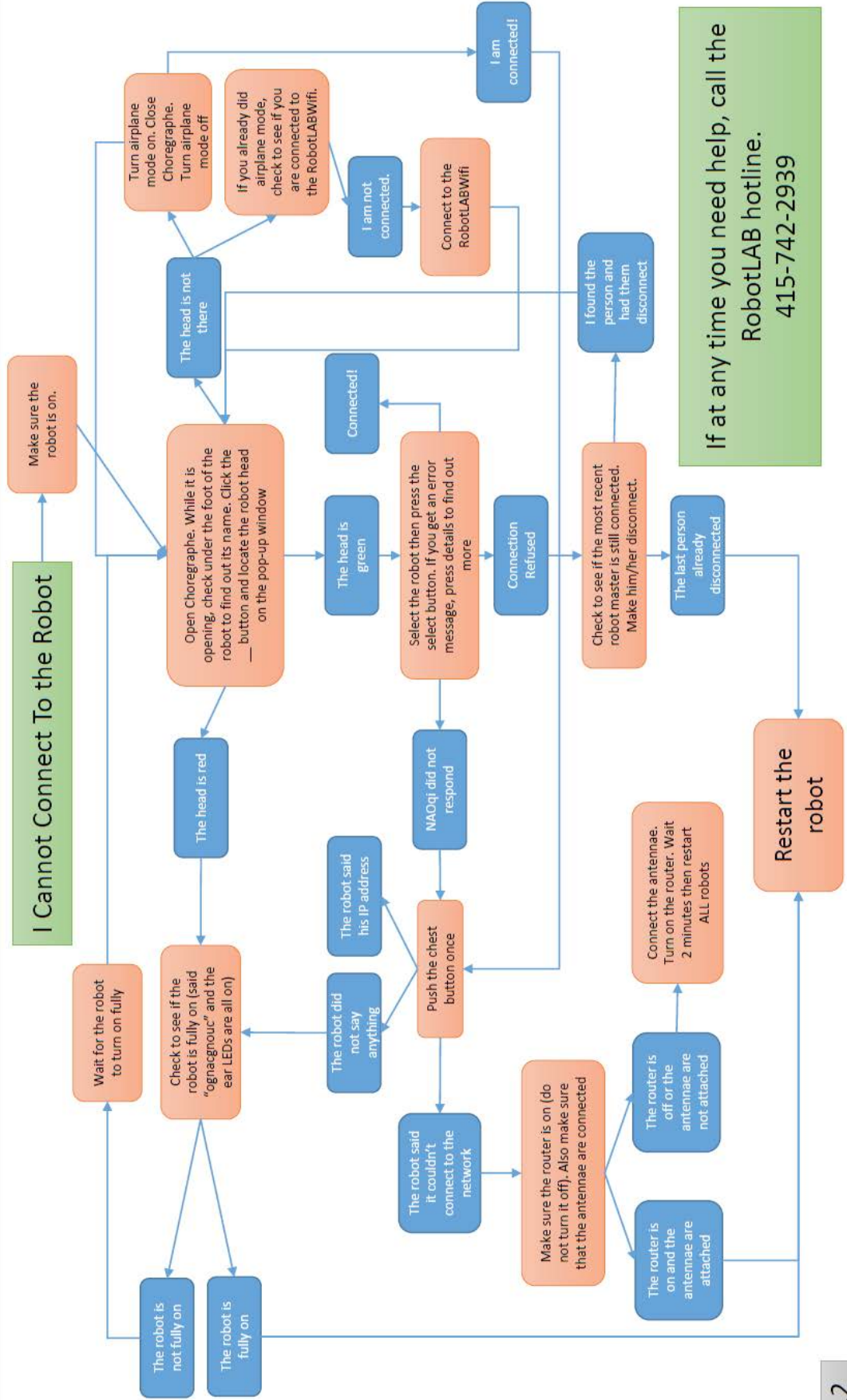
Box	NAME	FUNCTION
	<p><b>PLAY SOUND</b></p>	<p>Plays a sound file through NAO's speakers</p>
	<p><b>RANDOM EYES</b></p>	<p>Changes the color of the eyes randomly</p>
	<p><b>RANDOM INT.</b></p>	<p>Chooses a random number from a range of numbers</p>
	<p><b>REST</b></p>	<p>Turns the motors off after putting him in a safe position</p>
	<p><b>ROBOT POSTURE</b></p>	<p>Outputs the name of the robot's current posture</p>
	<p><b>SAY TEXT</b></p>	<p>NAO says the text given to it through the blue input port</p>
	<p><b>SINGLE EYE LED</b></p>	<p>Sets the color of a single eye</p>
	<p><b>SIT DOWN</b></p>	<p>Makes the NAO sit down smoothly from any position</p>
	<p><b>SPEECH RECO.</b></p>	<p>Recognizes the word a human says from a particular list of words</p>

# Box Glossary

Box	NAME	FUNCTION
	<b>STAND UP</b>	Makes the NAO stand up smoothly from any position
	<b>SWITCH CASE</b>	Changes tasks based on an input
	<b>TACTILE HEAD</b>	Detects a touch on the head
	<b>TACTILE L.HAND</b>	Detects a touch on the left hand
	<b>TACTILE R.HAND</b>	Detects a touch on the right hand
	<b>TIMELINE</b>	Makes NAO do a custom motion
	<b>WAIT</b>	Waits a certain amount of time
	<b>WAIT FOR SIGNALS</b>	Waits for both input ports to be signaled
	<b>WAKE UP</b>	Turns all of NAO's motors on

# NAO CONNECTION-TROUBLE FLOW CHART

## CHART



## I AM HAVING CONNECTIVITY ISSUES WITH THE ROBOT.

See the flow chart on the previous page.

## A ROBOT IS BROKEN / MAKING GRINDING NOISES. WHAT SHOULD I DO?

If a robot is broken or damaged, it can be sent in for repair. Please contact the RobotLAB hot-line at (415) 742-2939. Additionally, you should report any damage to your manager.

## A COMPUTER IS BROKEN, WHAT CAN I DO?

Call the RobotLAB hot-line at (415) 742-2939. We can help you determine the extent of the damage and if the computer can be fixed with a factory reset.

## A CASE, OR ANY OF ITS CONTENTS, IS MISSING. WHAT SHOULD I DO?

The complete system consists of five cases. Three of the cases contain a NAO robot. The other two are used to carry the 16 laptops, chargers and router and power source. Determine what is missing and promptly report it to your supervisor. Have them contact RobotLAB at (415) 742-2939.

## I'M GETTING A WINDOWS UPDATE POP-UP ON MY SCREEN. WHAT SHOULD I DO?

Some of the computers will ask you to update or activate windows. Since the computers are not connected to the internet, they occasionally complain about this problem. The best way to deal with this issue is to just accept it, and then click the windows icon-key and re-select the desktop environment.

## HOW DO I CONNECT THE TEACHER LAPTOP TO A PROJECTOR OR SCREEN?

On the side of the computer there is an HDMI port. Most modern projectors and screens use HDMI connectors. If your screen uses a different connection, it is probably VGA. You will need an adapter for this. Contact your supervisor for an adapter.

## THE ROBOT SAID THAT SOME OF ITS MOTORS ARE HOT. WHAT SHOULD I DO?

If the motors are engaged for too long, they overheat. Eventually, the robot will complain about it. At this point you can either press the chest button twice to release the motors, or press the moon button in Choregraphe (it must be connected to that particular robot). Sit NAO down and wait until he cools, 5-10 minutes.

## THE ROBOT SAID ITS HEAD WAS HOT. WHAT SHOULD I DO?

Place NAO in an area with lots of ventilation, like near a window. This should help his head cool down. If this does not work, turn NAO off for a few minutes.

## MY STUDENTS DID SOMETHING TO CHOREGRAPHE AND I CAN NO LONGER SEE ROBOT VIEW OR THE BOX LIBRARY. WHAT SHOULD I DO?

From the View menu, select Reset Views.

**If you have any additional issues, please contact the RobotLAB hot-line at (415) 742-2939.**

# PACKING INSTRUCTIONS

At the end of each lesson you need to pack everything back into the pelican cases. Below is a checklist to help you remember everything. There are four black spots in the checklist for you to add any additional items you use. Packing the NAO can be difficult. He is fragile and you can break his fingers or other body parts in the packing process. For that reason We have included step by step instructions on packing NAO (located on the next page).

**3 NAO ROBOTS**

**3 NAO CHARGERS**

**16 BLUE COMPUTERS**

**16 COMPUTER CHARGERS**

**1 ORANGE ROUTER (INCLUDING BOTH ANTENNAE)**

**1 ROUTER POWER CORD**

---

---






---

---


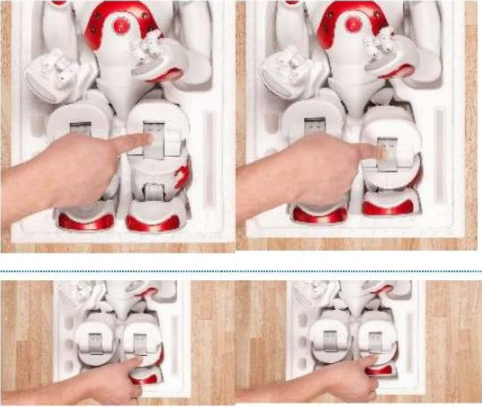

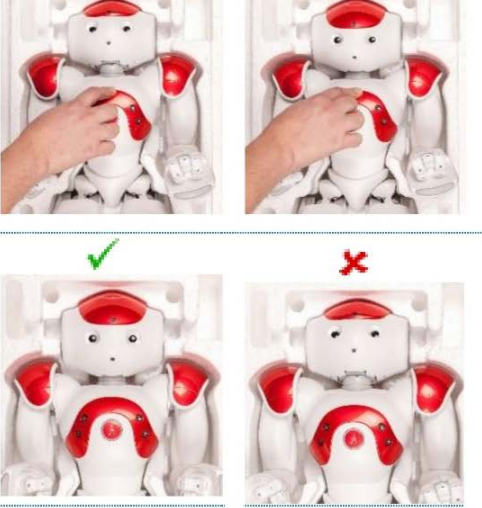


# PACKING INSTRUCTIONS

## PACKING NAO IN FOAM

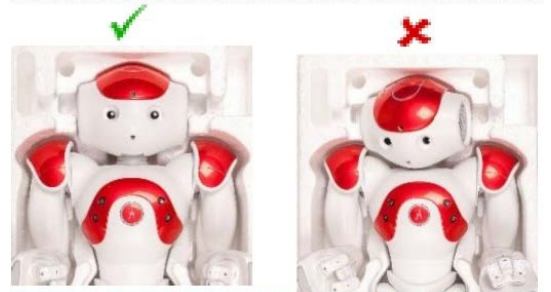
Step	Action
1	<p>Make sure NAO's hands are closed. Otherwise, NAO's fingers could get damaged.</p> <p><b>X</b>  <b>✓</b> </p> <p>To Close NAO's hands, try these actions:</p> <ul style="list-style-type: none"><li>▶ Turn the NAO on then off.</li><li>▶ Connect NAO to Choregraphe and close his hands by moving joints in robot view.</li><li>▶ If the hands seem broken, use a rubber band to keep them closed.</li></ul>
2	<p>Wait for NAO to cool down before storing it!</p> <p>Placing a warm NAO in the polystyrene casing can alter the plastic finish.</p>
3	<p>Put NAO in the Crouch Posture</p> 
4	<p>Take the polystyrene casing out of the box.</p> 
5	<p>Grab and lift NAO, keeping it in the crouch posture.</p> <p>Tip: Place one hand on the back, and the other under the feet.</p> 

# PACKING INSTRUCTIONS





Step	Action
6	<p>Starting with the head, gently place NAO into the polystyrene casing with the face up.</p> 
8	<p>Gently push each knee to the bottom of the polystyrene casing.</p> 
9	<p>For each leg, bend the ankle so the foot and knee are as close as possible. The bottom of the feet should be completely touching the polystyrene casing.</p> 
10	<p>Press the chest until the back is in contact with the polystyrene casing.</p> <p>The head should not be falling backward.</p> 

# PACKING INSTRUCTIONS

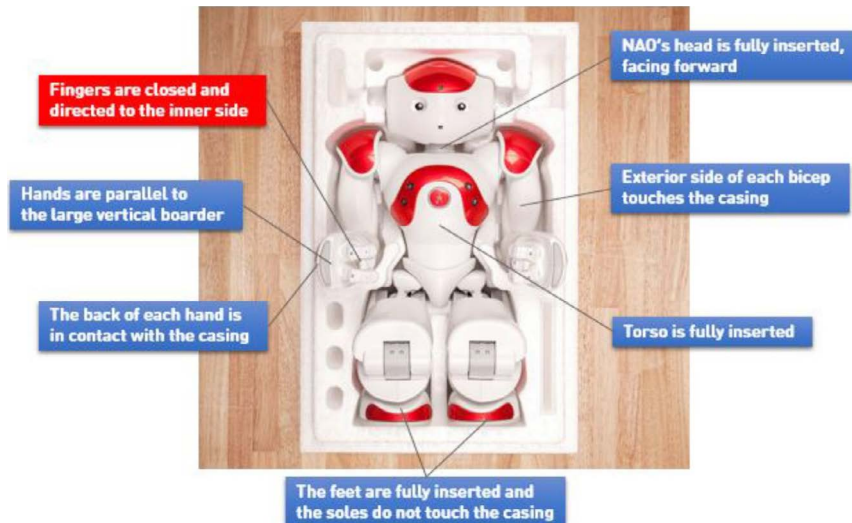
Step	Action
11	<p>Press the head back until it is in contact with the polystyrene casing.</p> <p>NAO should be facing forward.</p>
12	<p>NAO's hands should...</p> <ul style="list-style-type: none"> <li>▶ Be parallel to the vertical border of the polystyrene casing.</li> <li>▶ Have the fingers directed inwards.</li> </ul>
13	<p>Pull the hands gently to the exterior.</p> <p>Each hand should be in contact with the polystyrene casing.</p> <p>Each elbow should be in contact with the inner polystyrene casing.</p>
14	<p>Push the biceps gently with one finger until they are in contact with the back of the polystyrene casing.</p>



# PACKING INSTRUCTIONS

Step	Action
	<p>The exterior of each bicep should touch the polystyrene casing.</p> <p>The back of each hand should be in contact with the polystyrene casing.</p> 
15	<p>To close the polystyrene casing, place the top casing so that its bottom edge touches the bottom casing.</p> 
16	<p>With one hand maintaining the position of the bottom edge, use the other hand to gently lower the top casing onto the NAO.</p> <p><b>DO NOT PUSH THE CASING CLOSED</b> - this breaks the NAO's fingers.</p> 
17	<p>Check all edges to make sure that the top and bottom casings are flush.</p> 

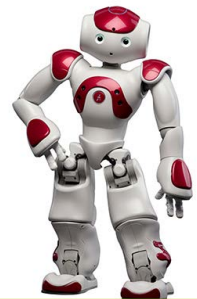
You are done! Below is a quick reminder guide.



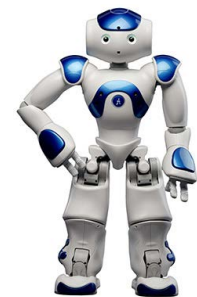
# NAME CARD TEMPLATE

To make name cards. Photocopy this page until you have enough name cards for your class. Cut along the dotted lines and fold along the solid lines to make tents.

MY NAME IS...



MY NAME IS...





# TAKE-HOME QUESTIONS

## LESSON 1: I KNOW THAT!

### MAIN TOPICS IN TODAY'S LESSON:

Today the students made the NAO learn flash cards. They had the robot say a few sentences when the robot recognized a card.

### QUESTIONS TO ASK YOUR CHILD:

What cards did you use in your program?

What did the robot say when he saw one of the cards?

What is your favorite card.

## LESSON 2: GUESS WHAT!

### MAIN TOPICS IN TODAY'S LESSON:

The NAO picked a number between 1 and 10 and asked the students to guess the number.

### QUESTIONS TO ASK YOUR CHILD:

How many tries did it take for you to guess NAO's number?

What was your game?

How was it different from the game NAO played at the beginning of the lesson?

## LESSON 3: LET ME QUIZ YOU!

### MAIN TOPICS IN TODAY'S LESSON:

Lists and indexes.

NAO picked a definition at random and the students had to show him the corresponding quadrilateral.

### QUESTIONS TO ASK YOUR CHILD:

How is a square different from a kite?

What cards did you use in your game?

Tell me what your game did step by step.

## LESSON 4: ROCK, PAPER, SCISSORS!

### MAIN TOPICS IN TODAY'S LESSON:

NAO played rock, paper, scissors with the students using rock paper scissor cards. The students then had to make a game similar to rock, paper, scissors with their own twist on the rules.

### QUESTIONS TO ASK YOUR CHILD:

Which cards did you use in your game?

What were the rules of your game?

# TAKE-HOME QUESTIONS

## LESSON 5: LET'S PLAY!

### MAIN TOPICS IN TODAY'S LESSON:

The students programmed a menu that let them choose between all the games they made in this module.

They presented their games to the class.

### QUESTIONS TO ASK YOUR CHILD:

Which games were your favorite?

How did the menu work? What buttons did you have to press?



**QUADRILATERAL**  
**PARALLELOGRAM**  
**RECTANGLE**  
**RHOMBUS**  
**SQUARE**



**QUADRILATERAL**  
**PARALLELOGRAM**



**QUADRILATERAL**  
**PARALLELOGRAM**  
**RHOMBUS**



**QUADRILATERAL**  
**KITE**





**QUADRILATERALS**



**QUADRILATERALS**



**QUADRILATERALS**



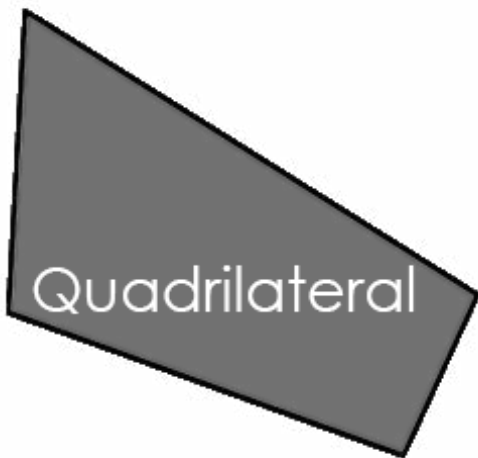
**QUADRILATERALS**



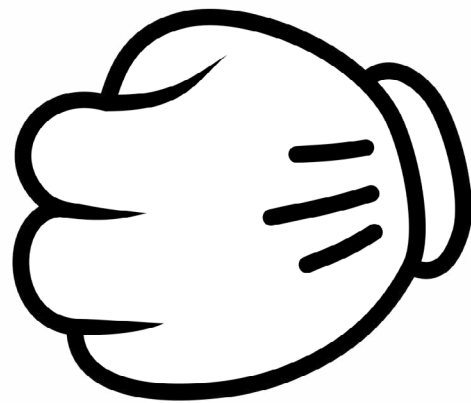
**QUADRILATERAL  
TRAPEZOID**



**QUADRILATERAL  
PARALLELOGRAM  
RECTANGLE**



**QUADRILATERAL**



**ROCK**



**QUADRILATERALS**



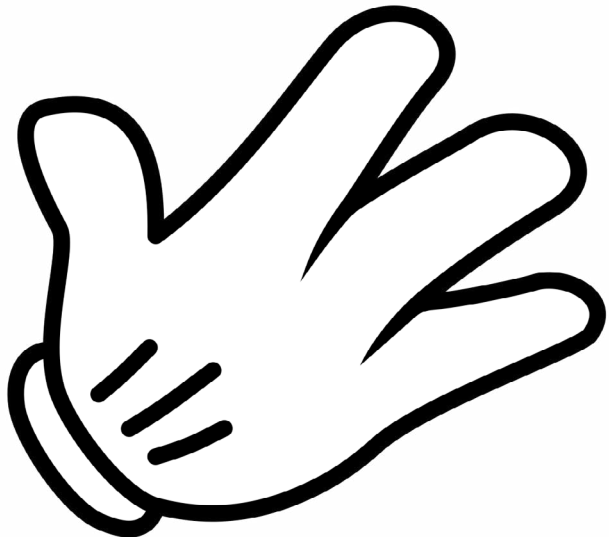
**QUADRILATERALS**



**ROCK, PAPER, SCISSORS**



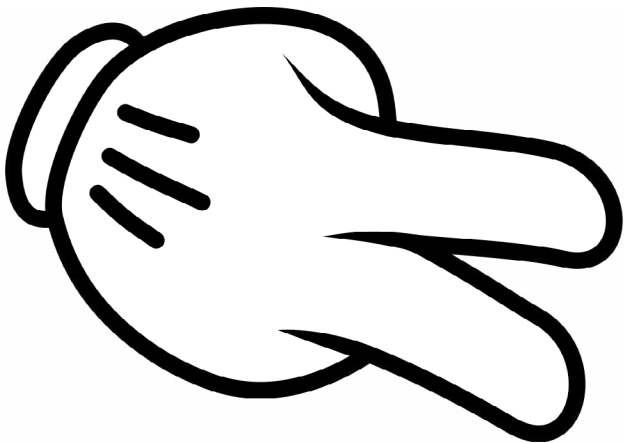
**QUADRILATERALS**



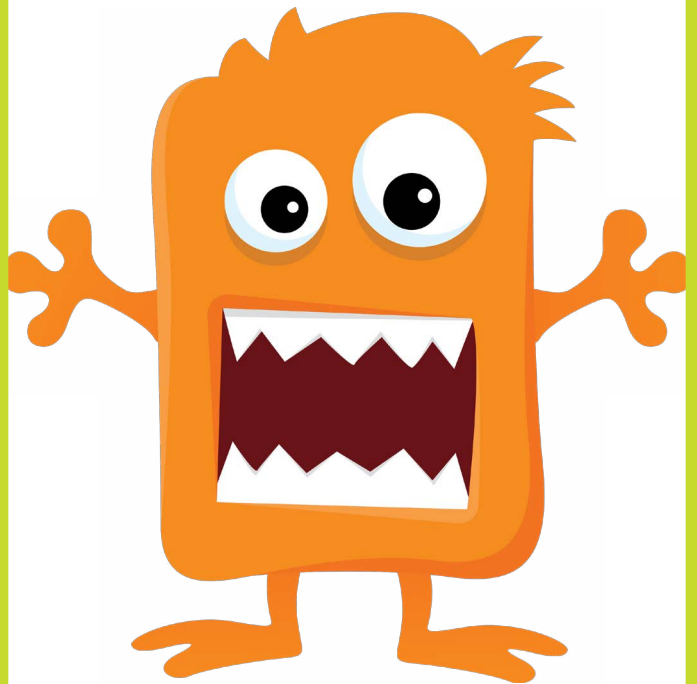
**PAPER**



**BLUE MONSTER**



**SCISSORS**



**ORANGE MONSTER**



**MONSTERS**



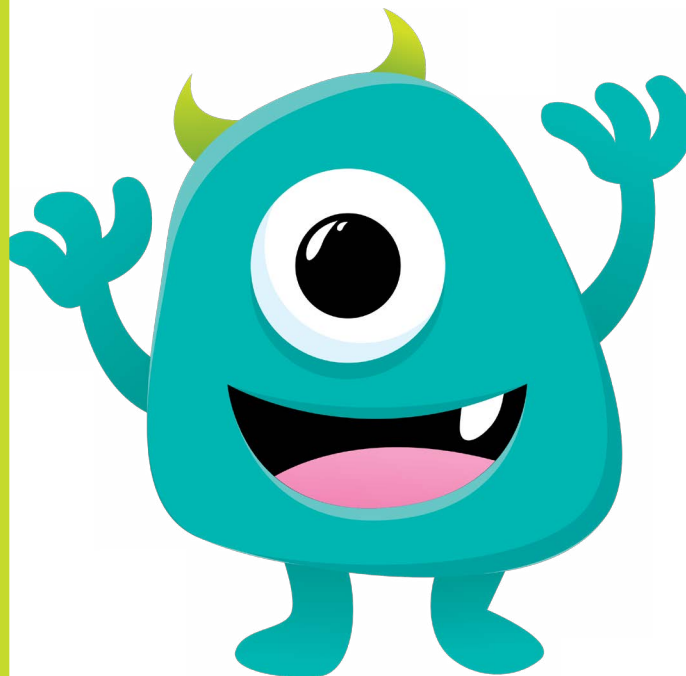
**ROCK, PAPER, SCISSORS**



**MONSTERS**



**ROCK, PAPER, SCISSORS**



**TEAL MONSTER**



**PURPLE MONSTER**



**GREEN MONSTER**



**RED MONSTER**



**MONSTERS**



**MONSTERS**



**MONSTERS**

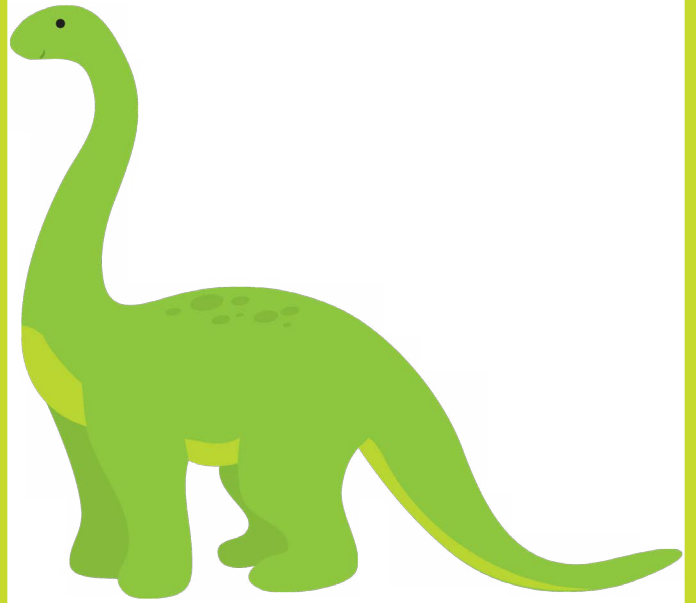


**MONSTERS**

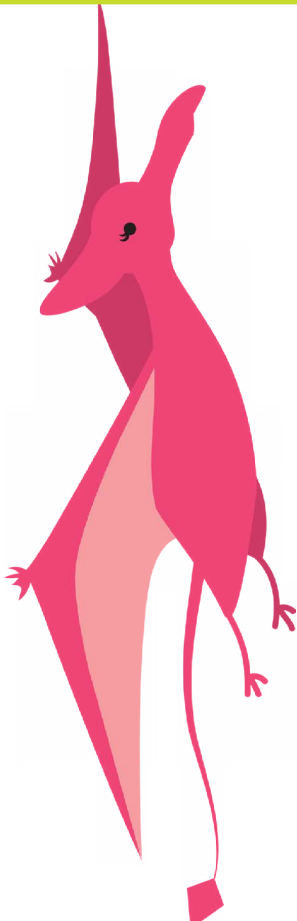




**T-REX**



**PLESIOSAURUS**



**PTERODACTYL**



**PALM  
TREE**





**DINOSAURS**



**DINOSAURS**



**DINOSAURS**



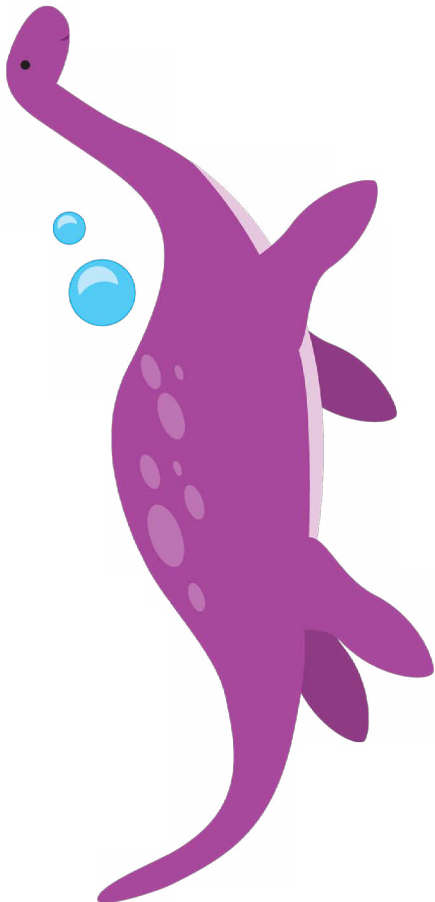
**DINOSAURS**



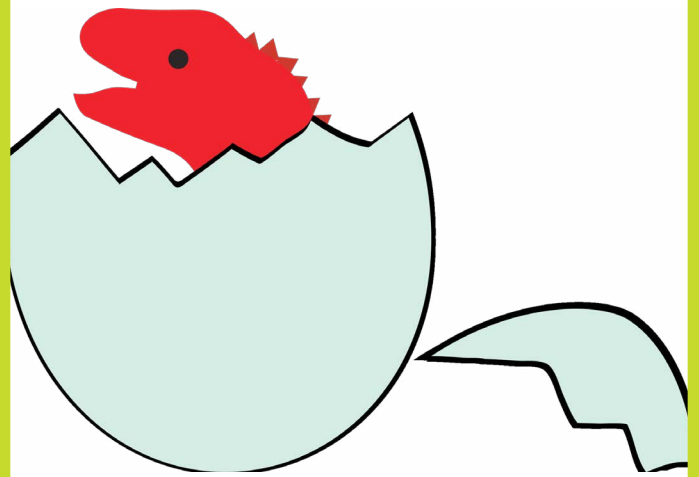
**TRICERATOPS**



**STEGOSAURUS**



**PLESIOSAURUS**



**T-REX EGG**



**DINOSAURS**



**DINOSAURS**



**DINOSAURS**



**DINOSAURS**

1 .

• •  
2

• •  
•  
3

• •  
• •  
4



**NUMBERS**



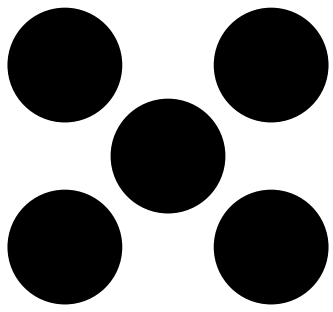
**NUMBERS**



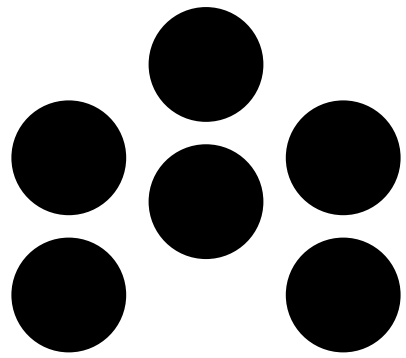
**NUMBERS**



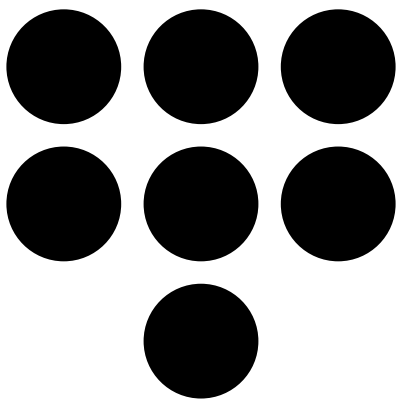
**NUMBERS**



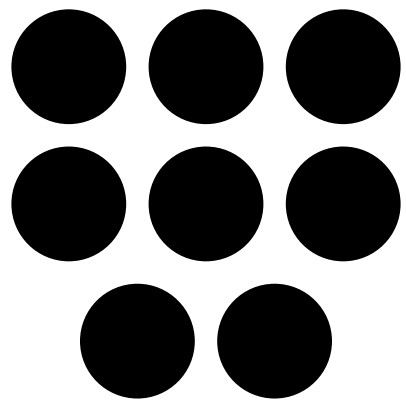
5



6



7



8



**NUMBERS**



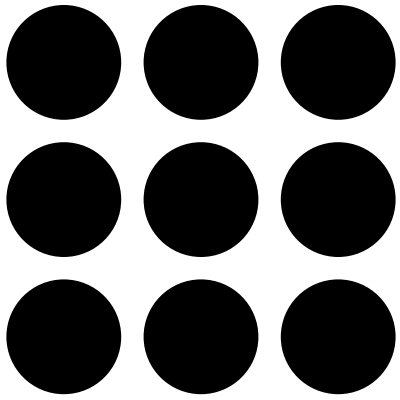
**NUMBERS**



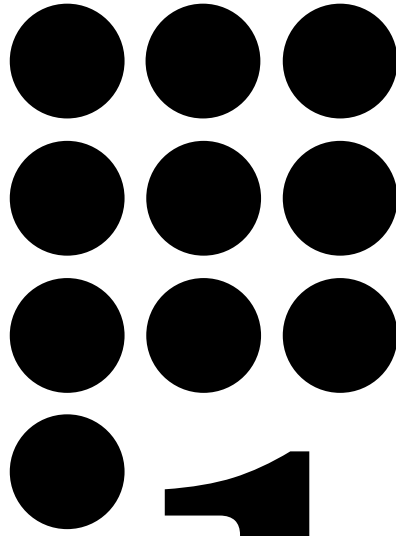
**NUMBERS**



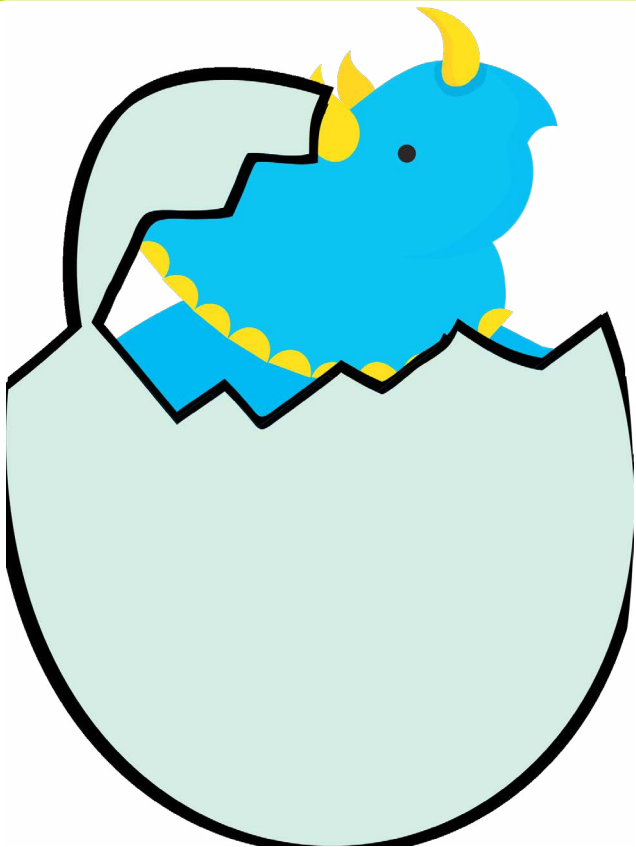
**NUMBERS**



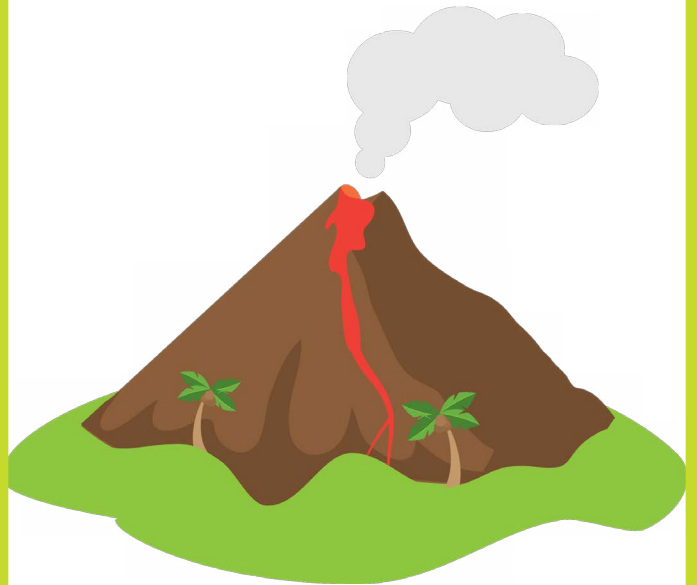
9



10



**TRICERATOPS EGG**



**VOLCANO**





**NUMBERS**



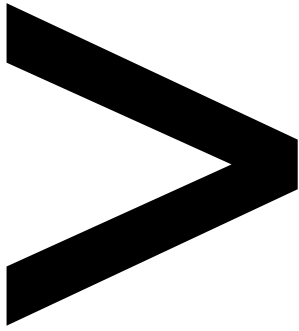
**NUMBERS**



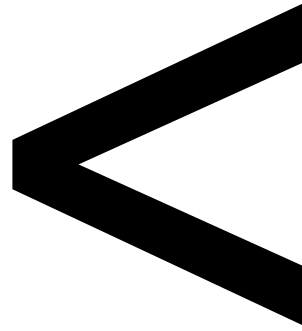
**DINOSAURS**



**DINOSAURS**



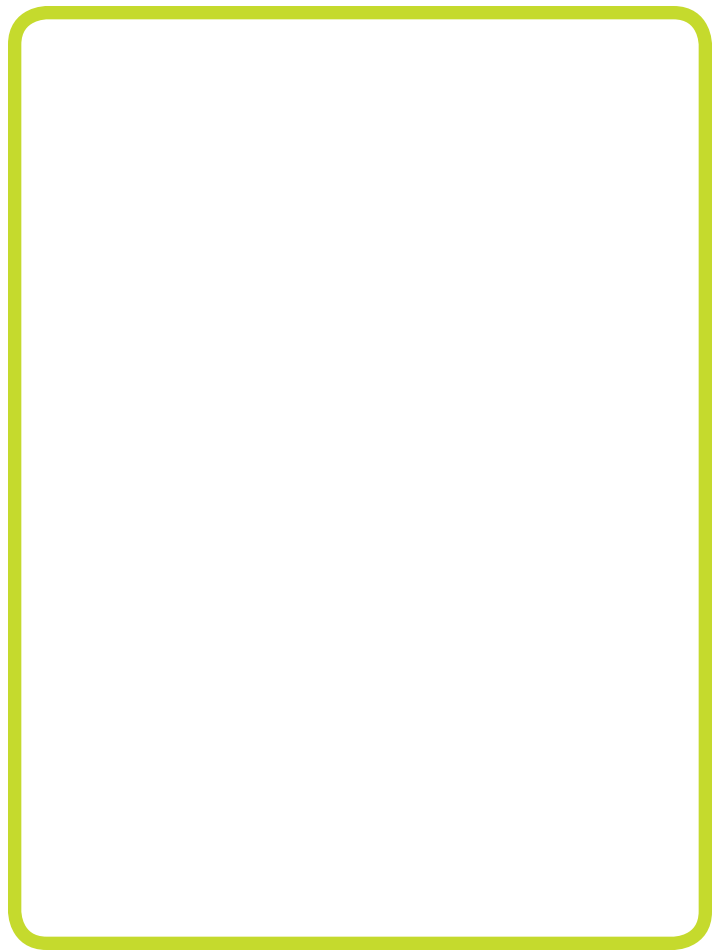
**GREATER**



**LESS**



**EQUAL**





**BONUS CARDS**



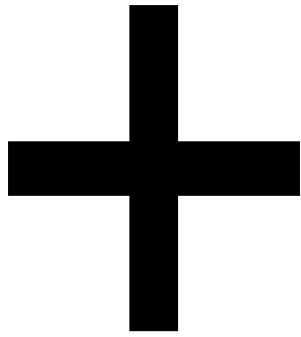
**BONUS CARDS**



**BONUS CARDS**



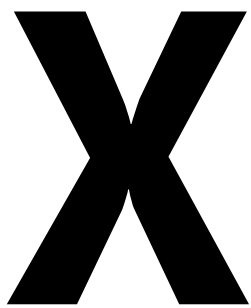
**BONUS CARDS**



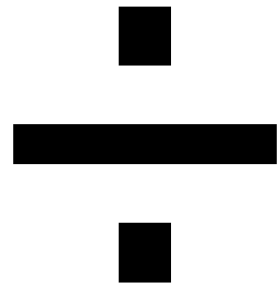
**PLUS**



**MINUS**



**TIMES**



**DIVIDED BY**



**BONUS CARDS**



**BONUS CARDS**



**BONUS CARDS**



**BONUS CARDS**