

Using the Camera

- **Key Concepts:**
 - Understand and use the camera
- **Pacing:**
 - Over several class periods

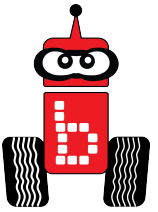


Table of Contents

[Introduction to Camera: Activity 1](#)

[Step 1](#)

[Step 2](#)

[Step 3](#)

[Camera Functions: Activity 2](#)

[I See Green: Activity 3](#)

[Tracking Left, Right, and Straight: Activity 4](#)

[Understanding New Functions: Activity 5](#)

[Where is the Object's Center?: Activity 6](#)

[Is the Object Left or Right? Activity 7](#)

[Camera Function Activity 8](#)

Standards

Goal:

- Students will familiarize themselves with the functions `msleep()` and `motor()`
- Students will understand how to move their robots in the following manner: forwards, backwards, straight,, circles, right and left turns

Standards:

Common Core State Standards Math Practices

CCSSMP1: Make sense of problems and persevere in solving them

CCSSMP2: Reason abstractly and quantitatively

CCSSMP4: Model with mathematics

CCSSMP6: Attend to precision

CCSSMP8: Look for and express regularity in repeated reasoning

Next Generation Science and Engineering Practice

1: Asking questions and defining problems

2: Developing and using models

3: Planning and carrying out investigations

4: Analyzing and interpreting data

5: Using mathematics and computational thinking

6: Constructing explanations and designing solution

7: Engaging in argument from evidence obtaining, evaluating, and communicating information

Standards Continued

Standards Continued:

2016 ISTE Standards

Empowered Learner

1c: Students use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.

1d: Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

Knowledge Constructor

3d: Students build knowledge by actively exploring real-world issues and problems, developing ideas and theories and pursuing answers and solutions.

Innovative Designer

4a: Students know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.

4b: Students select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.

4c: Students develop, test and refine prototypes as part of a cyclical design process.

4d: Students exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

Computational Thinker

5a: Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.

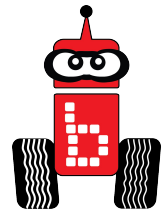
Introduction to Camera

Activity 1

Goal: Students will utilize the camera to track objects.

1. Follow steps 1 through 3 in the activity 1 slides that follow.

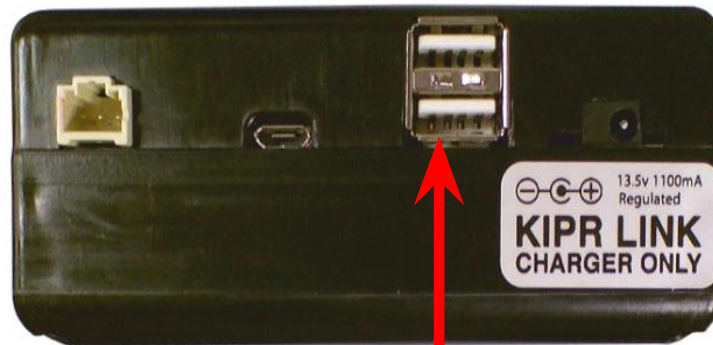
Introduction to Camera- Activity 1



Step 1

You will need the USB Camera

- The USB camera plugs into one of the USB (type A) ports on the back of the KIPR Link
- Unplugging the camera while it is being accessed will usually freeze the system, requiring a reboot

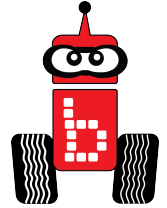


USB
Ports



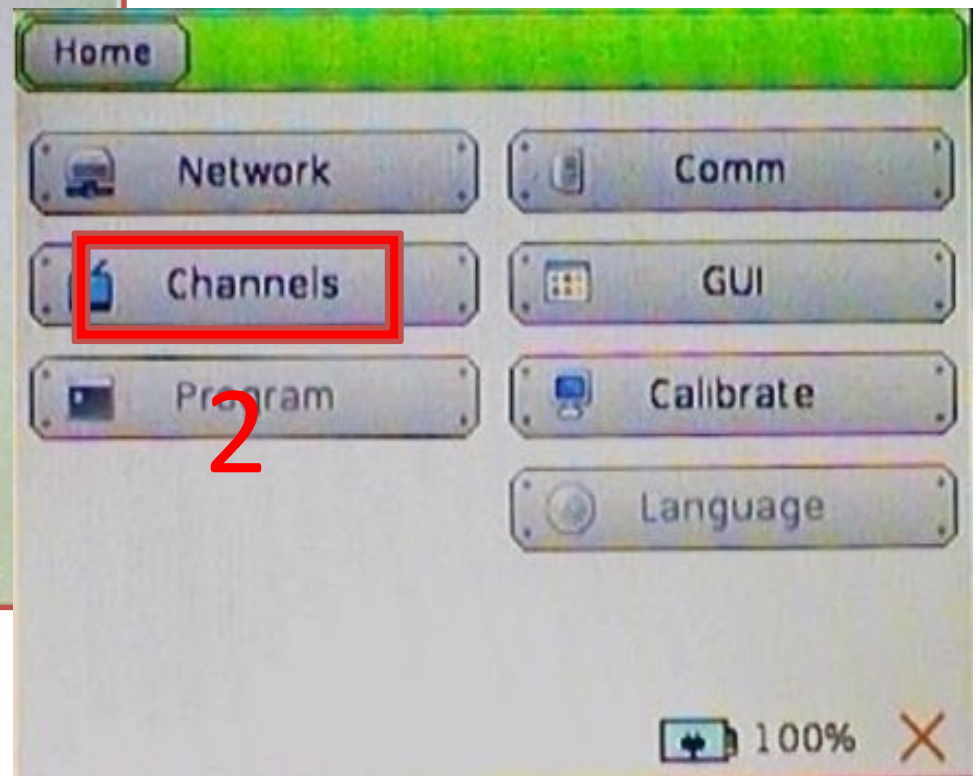
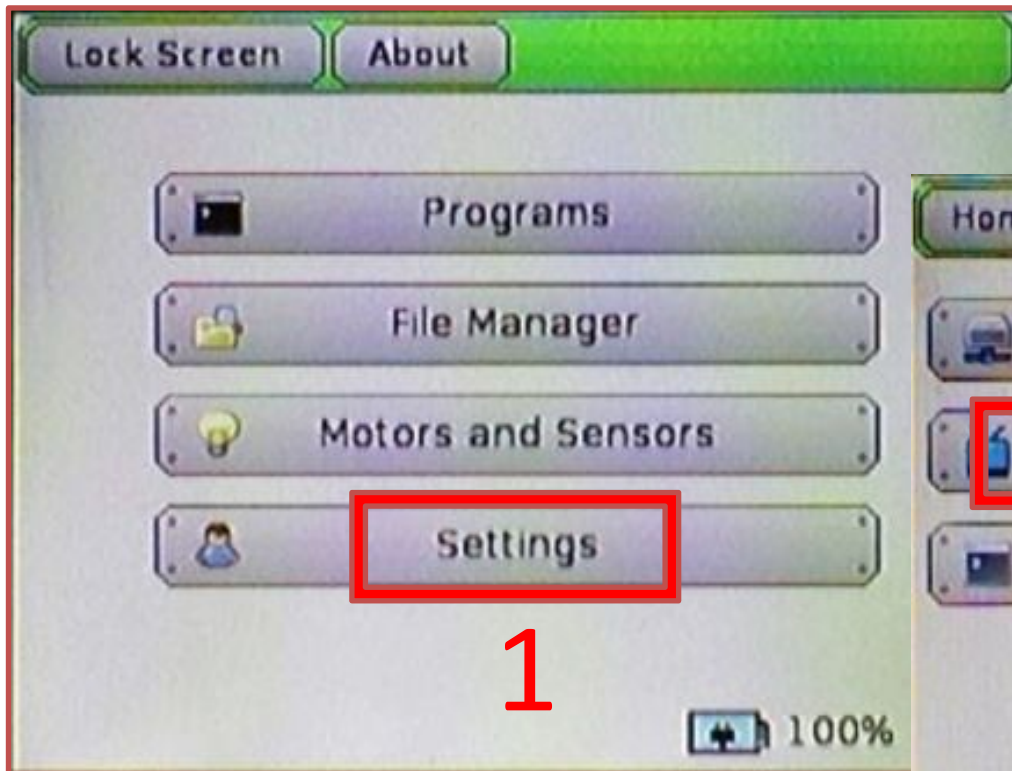
Intro. to Camera- Activity 1

Setting the Color Tracking Channels



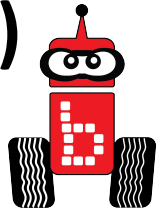
Step 2

1. Select *“Settings”*
2. Select *“Channels”*

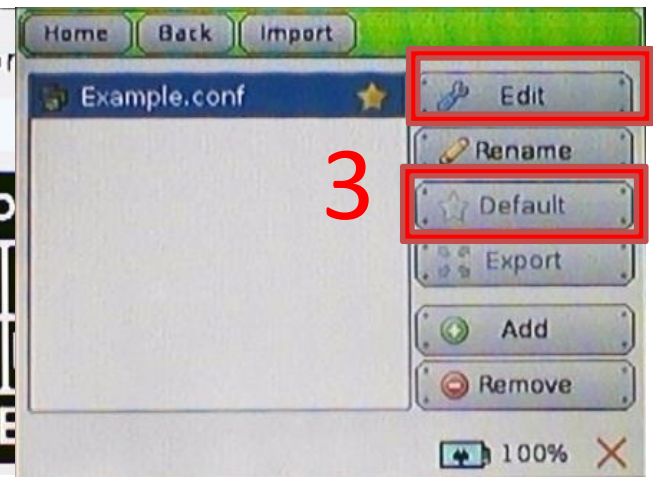
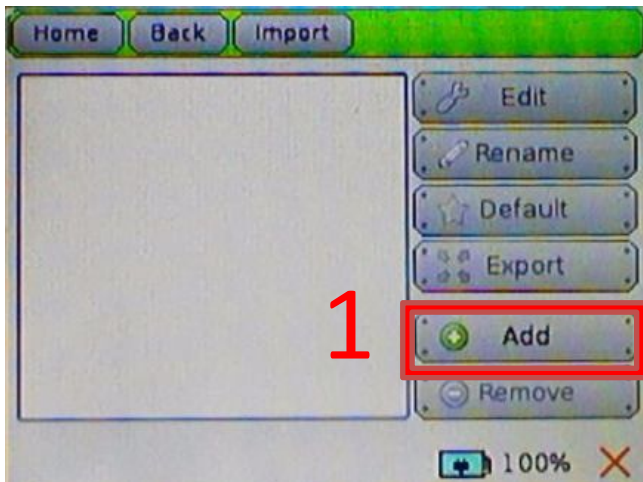


Intro. to Camera- Activity 1

Setting the Color Tracking Channels (Continue)

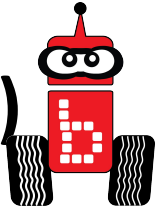


1. To specify a camera configuration select *“Add”*
2. Enter a configuration name such as *“find_green”* then press enter
3. Press *“Default”* and highlight the new configuration. This tells the camera to use this channel. *“Default”* is indicated by the gold star.
4. Press the *“Edit”* button

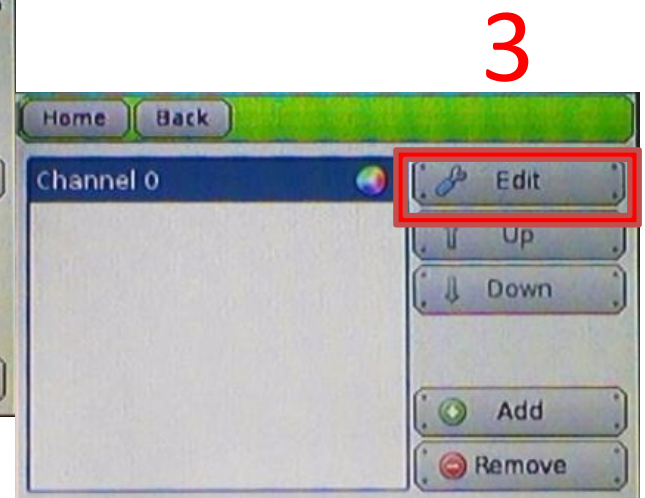
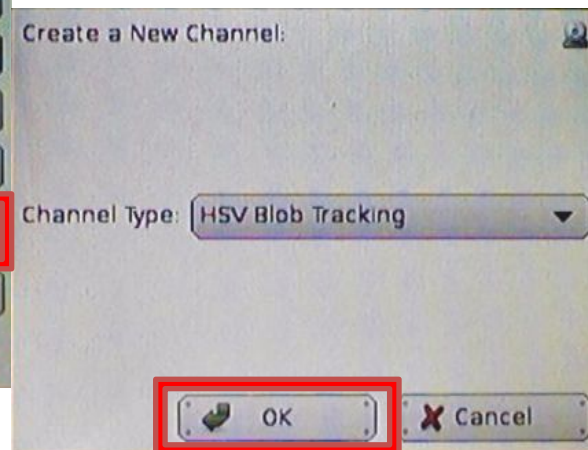
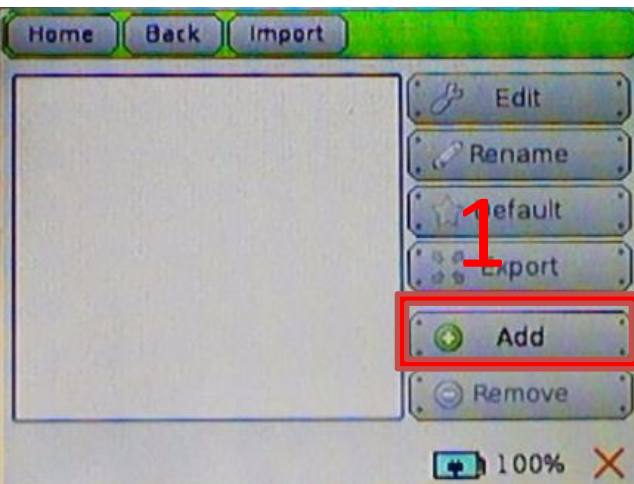


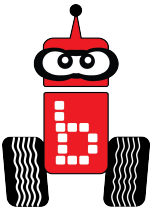
Intro. to Camera- Activity 1

Setting the Color Tracking Channels (Continued)



1. Press the *“Add”* button to add a channel to the configuration
2. Select *“HSV Blob Tracking”* then *“OK”* to make this track color
3. Highlight the channel name you just created and press the *“Configure”* button to edit settings
 - First channel is 0 by default you can add three more: 1,2,3

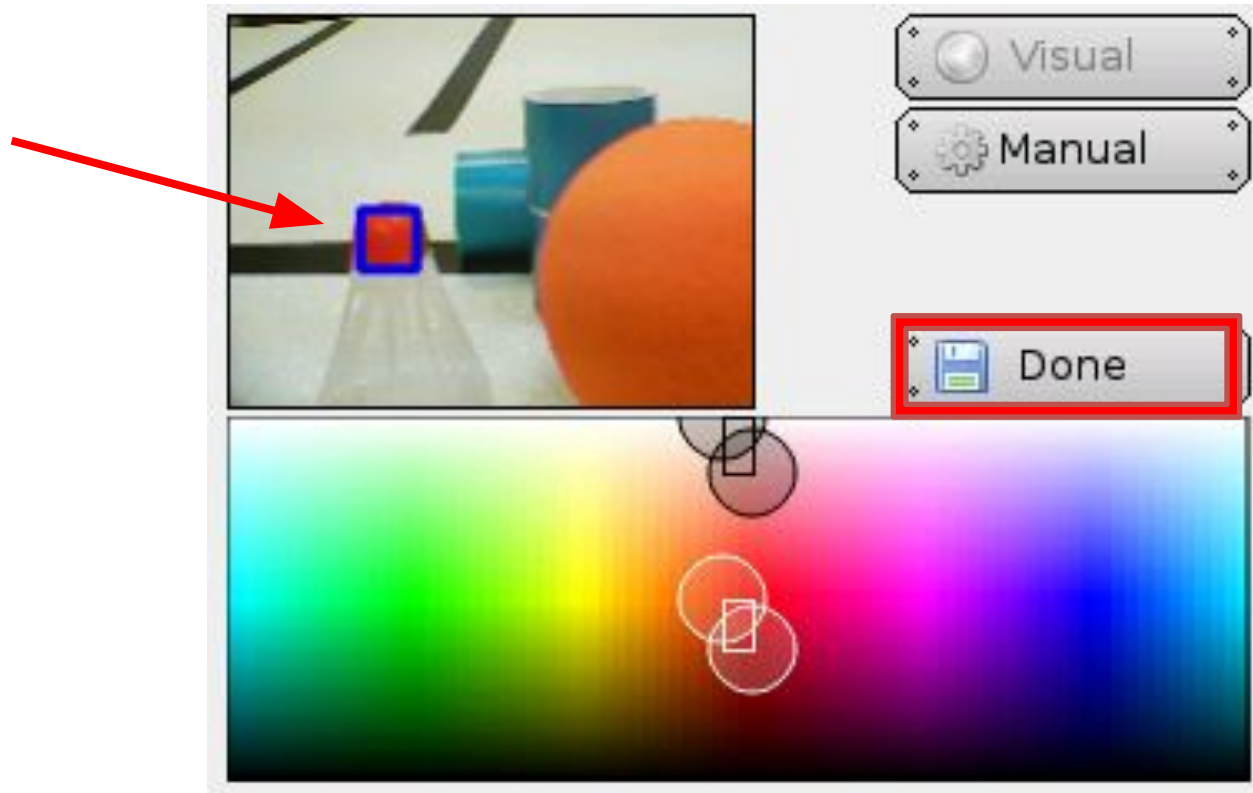




Intro. to Camera- Activity 1

Setting the Color Tracking Channels (Continued)

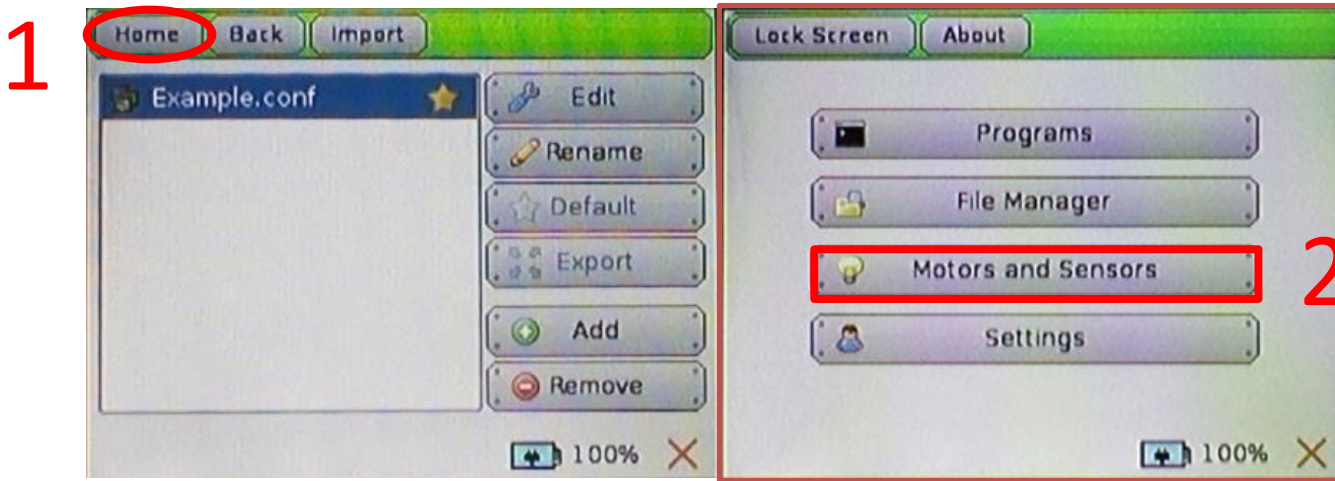
1. Place the colored object you want to track in front of the camera and touch it on the touch screen
 - The program will put a bounding box (dark blue) around the selected object then hit *“Done”*



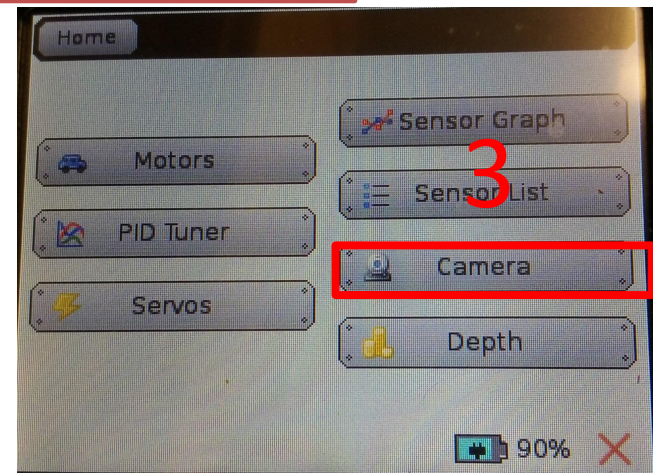
Intro. to Camera- Activity 1

Checking Color Tracking

Step 3

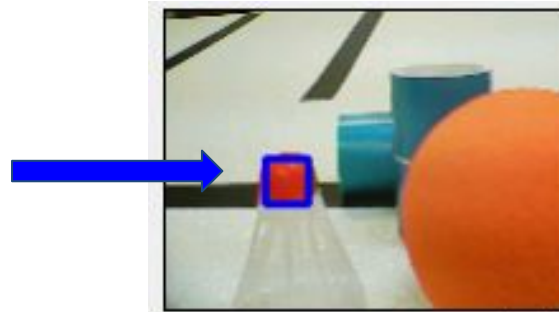


1. Press the home button.
2. Go to motors and sensors.
3. Go to camera.



Intro. to Camera- Activity 1

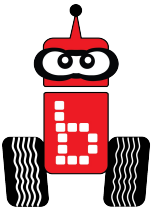
Checking Color Tracking (Continued)



1. Place your object in front of the camera again.
2. A **blue box** should appear around the object. This lets you know that the Link can “see” the object.
3. If the **blue box does not appear** (or appears on too many different objects) go back to [Setting the Color Tracking Channel Step 2](#) to reset your color tracking.

Proceed to the next slide!

Intro. to Camera- Activity 1



Congratulations! You
have set the color
tracking for the
camera.



Camera Functions

Activity 2



We know that:

1. All functions have **names**
2. All functions have a set of parentheses (the “**argument list**”)
 - Some functions need more information (“**arguments**”), such as `motor(0, 100)` and `msleep(3000)`
 - Some do not need more information, such as `ao()` and `enable_servos()`

The camera functions are no different.

Next slide...

Camera Functions Continued



1. Read and discuss these functions with your elbow partner.

```
camera_open(); // opens camera  
camera_close(); // closes camera  
camera_update(); // retrieves current image
```

Programming statements commonly used with camera

```
while(a_button() == 0) // repeats a block of code while  
                        the a-button is not pressed
```

```
if(get_object_count(channel#) > 0) // does the camera see at least  
                                   one "color you specified" object
```

Channel #: 0,1,2,3

- We chose 0 as our default

Boolean logic:

> Greater than

>= Greater than or equal

< Less than

<= Less than or equal

== Equal to

!= Not equal to

Number of objects

Camera Functions - Activity 2

Write the answers to the following questions. Share with your “elbow partner”.

1. Which function updates the camera image?
2. Which function turns the camera on?
3. When would you need to update the camera image? Before or after finding the object?
4. Which function is looking for the colored object?
5. What is the function that prints something to the screen?

Camera Functions - Activity 2

Answer

1. `camera_update (); // retrieves current image`
2. `camera_open ();`
3. Before
4. `get_object_count(channel#) > 0;`
5. `printf("Hi");`

I See Green

Activity 3

Goal: Write a program that will allow you to check to see if the camera is tracking the color that you want it to see.

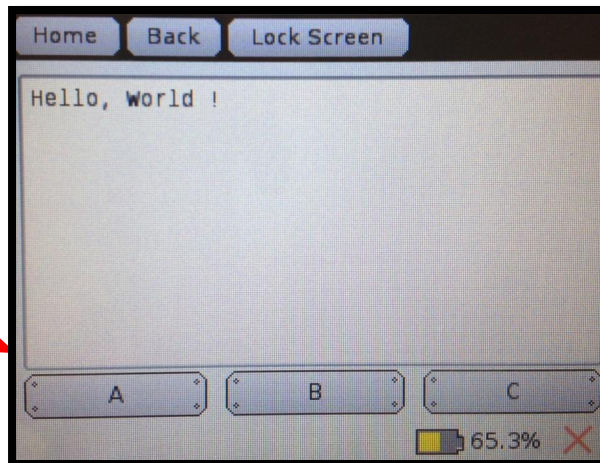
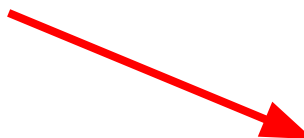
1. Use [code planning paper](#) to plan the steps in your code. [Color Tracking Review](#).
2. Your program must include:
 - prints the words “I see green” when the object comes into view.
 - finds the color you are tracking

I See Green: Activity 3 Continued

Example of code planning sheet:

1. Opens the Camera
 - (This should always be the first step when using the camera.)
2. Checks the status of the a_button
 - (Remember to use this step to create the **loop** that will keep your camera checking for images)
3. Updates camera image
4. Gets an object count
5. Prints “I see green.”
6. Remember if you want to stop the program you must press the a button: because you had a while loop that exits when a_button is pressed

Buttons



I see Green Activity 3

Program Example

```
#include <kipr/botball.h>

int main()
{
    camera_open(); // opens camera
    while (a_button() == 0) //Creates a loop
    {
        camera_update(); // retrieves current image

        if (get_object_count(0) > 0) //does the camera sees at least 1 green object
        {
            printf("I see green.\n");
        }
    }
    return 0;
}
```

(get_object_count(0) > 0)

channel # (0 is
always default)

number of objects

I See Green

Activity 3 Continued

Why does this work?

For color vision tracking, images are processed by the controller to identify "blobs" matching the color specification you set in the channel configuration.

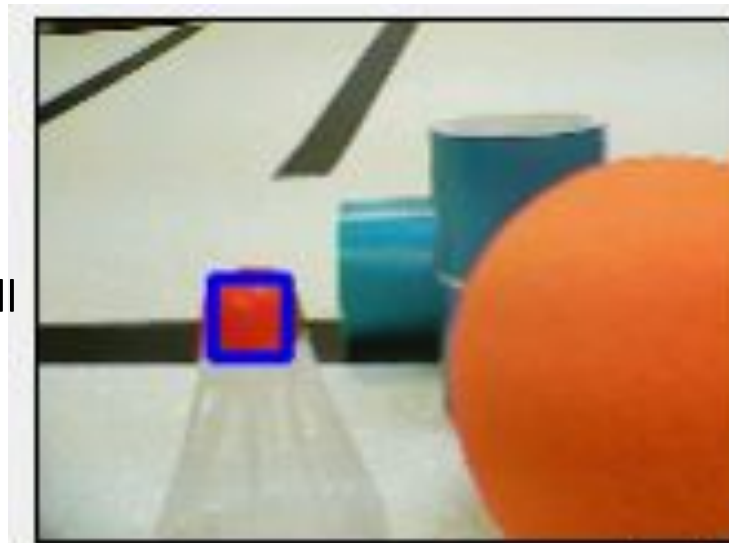
- A blob is a set of contiguous pixels in the image matching your channel color specification

The camera image size is in pixels 160 (columns) X 120 (rows)

- Remember we start counting with column 0 and end at column 159 (remember you start counting at 0)

160 wide

120 tall



I See Green

Activity 3 Continued

The camera image size is in pixels 160(columns) X 120(rows)

*Remember we start counting at 0

Imagine a row
and column
spreadsheet.



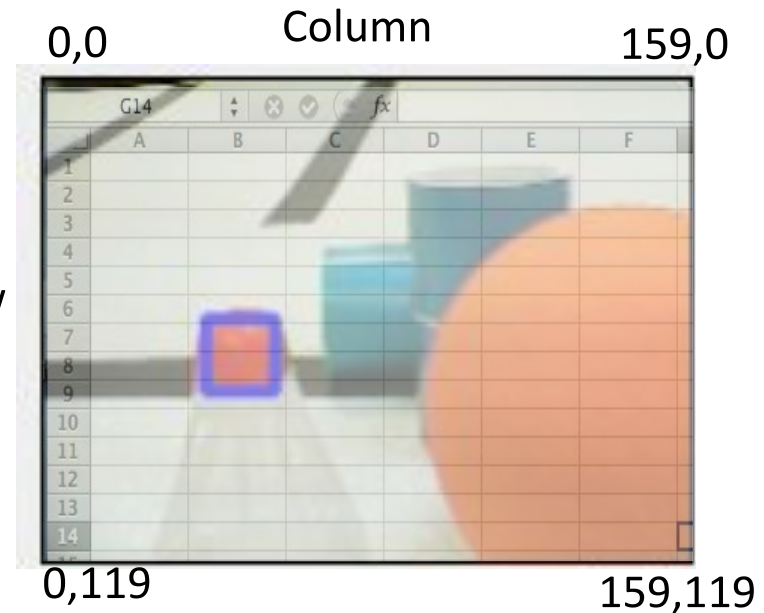
0,0 Column 159,0

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

Row

0,119

159,119



Tracking Left, Right, and Straight:

Activity 4

Part 1

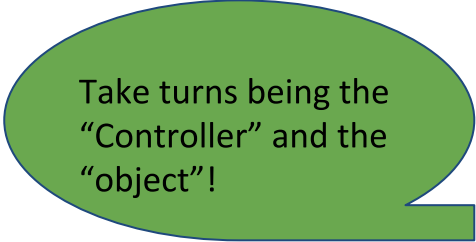
Form groups of 2 – one person will be the object and the other person will be the Controller (each person can hold signs or objects to tell which one they are)

Have the “object” choose a position (to the left, right, or in front of the “Controller”)

Have the “Controller” state where the “object” is (left, right, or in front of) and which way the “Controller” would need to turn to reach the object (is to the left, is to the right, center).

Part 2

Repeat Part 1 but instead of having the “object” move, the “object” person will call out a number between 0-159. The “Controller” would state which side the number is on (right, left, or in front of) and which way the “Controller” would turn (turn left, turn right, go straight)



Take turns being the “Controller” and the “object”!

Understanding New Functions:

Activity 5

```
get_object_center_column(channel #, object #);  
// retrieves the center column coordinate value
```

```
get_object_center_row(channel #, object #);  
// retrieves the center row coordinate value
```



Object

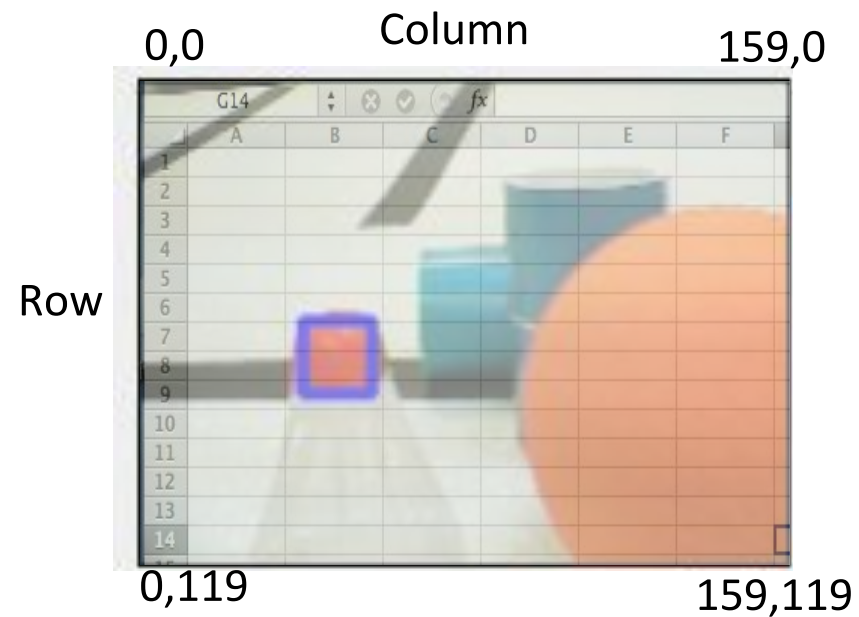
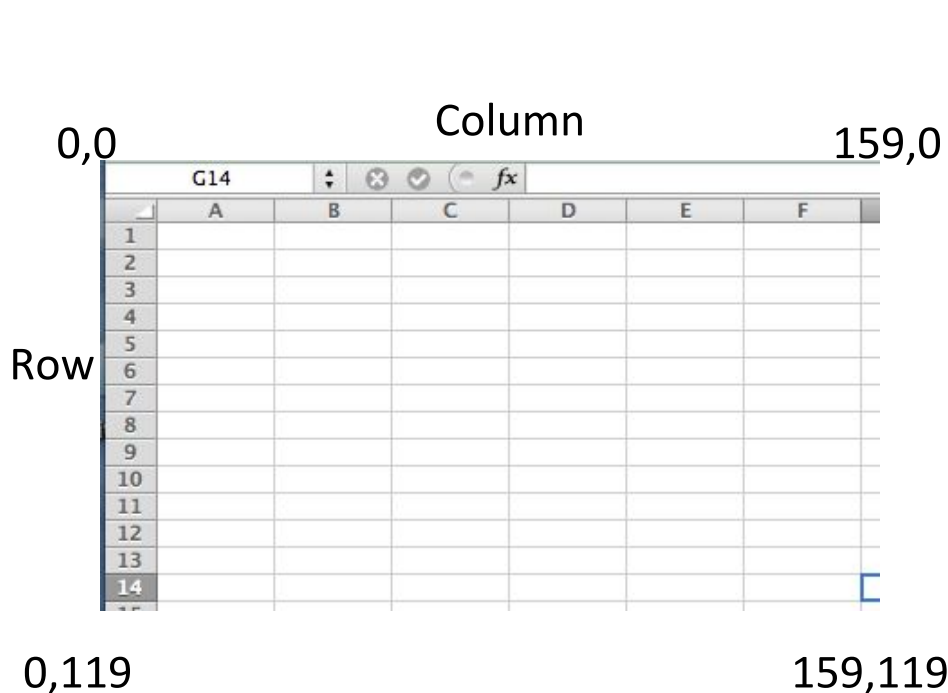
Objects are numbered from largest to smallest starting at zero. Zero being the largest number.

0,1, 2,...

Understanding New Functions:

Activity 5

- Your controller will try to find the center of the object you tell it to look for, the object number.
- It does this by finding the center column and row value on the display. This will always be written as **(column,row)**.
- This value is hidden from us, but the controller sees it.

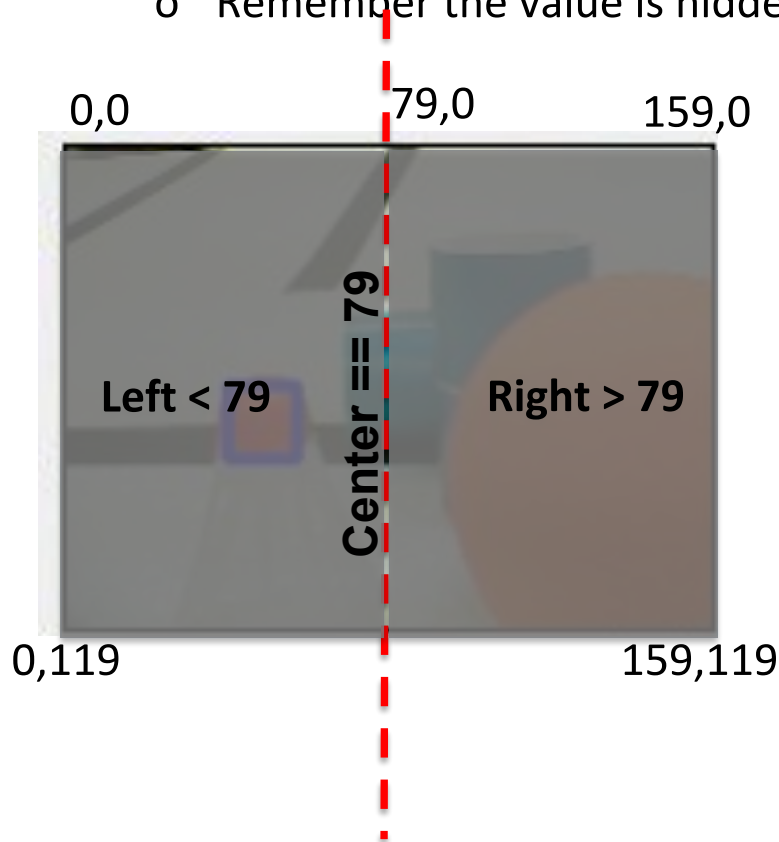


Understanding New Functions:

Activity 5

In your code you will use the position of the object in relation to the center of the image to tell if it is to the left or right

- And if you know that the image is 159 (columns) wide, then the center is 79
 - If the value returned is between 0 and 78 the object will be to **left**
 - If the value returned is between 80 to 159 the object will be to the **right**
 - If the value returned is 79 the object will be in the **center** of the image
 - Remember the value is hidden we can't see it!



Channel #

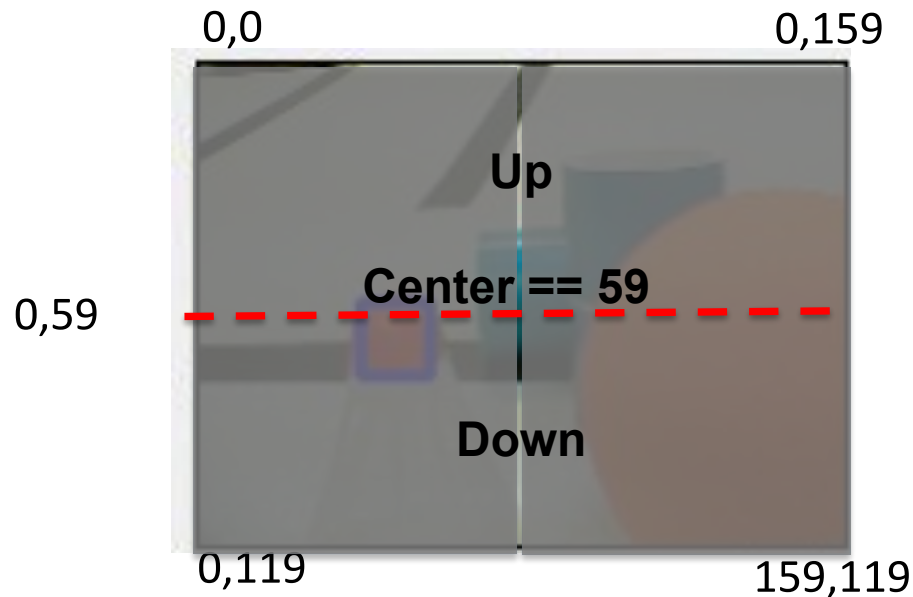
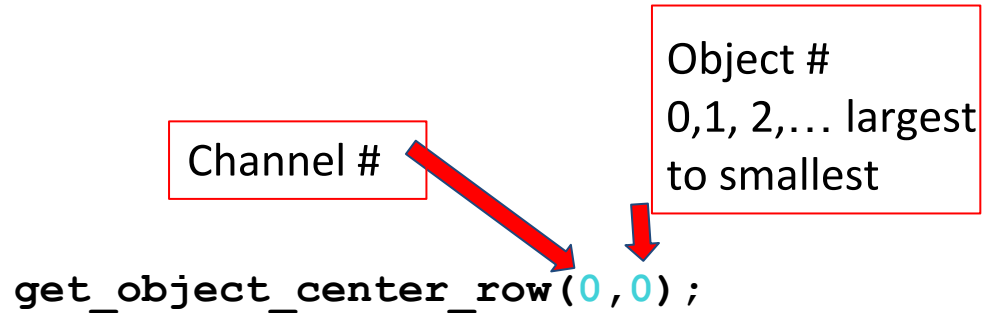
Object #
0,1, 2,... largest
to smallest

```
get_object_center_column(0,0);
```

The program is asking the controller what value between 0 and 159 is the center of the object. This value will be seen and used by the robot.

Understanding New Functions: Activity 5

The program is asking the controller what value between 0 and 119 is the center of the object. This value will be seen and used by the robot.



Understanding New Functions:

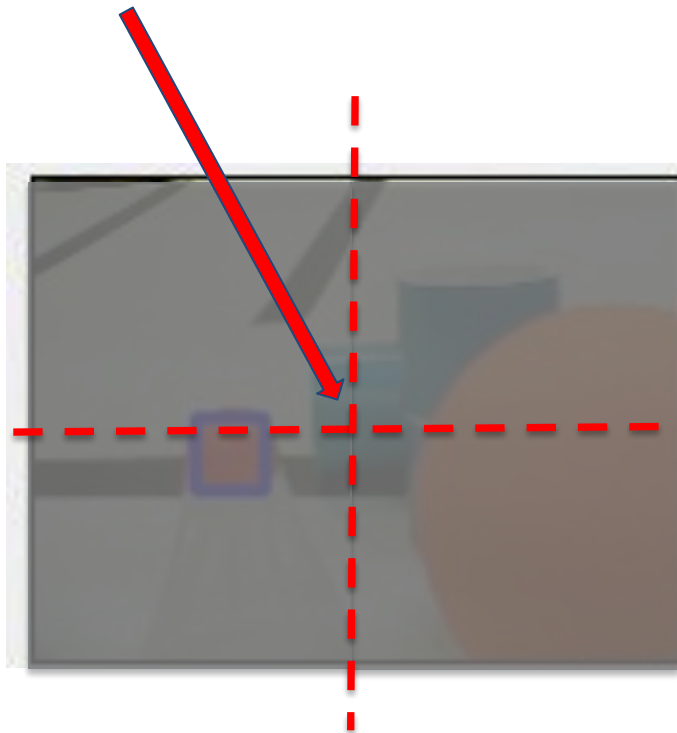
Activity 5

To find center you need a column and a row.

```
get_object_center_column(0,0);
```

```
get_object_center_row(0,0);
```

79,59



Center

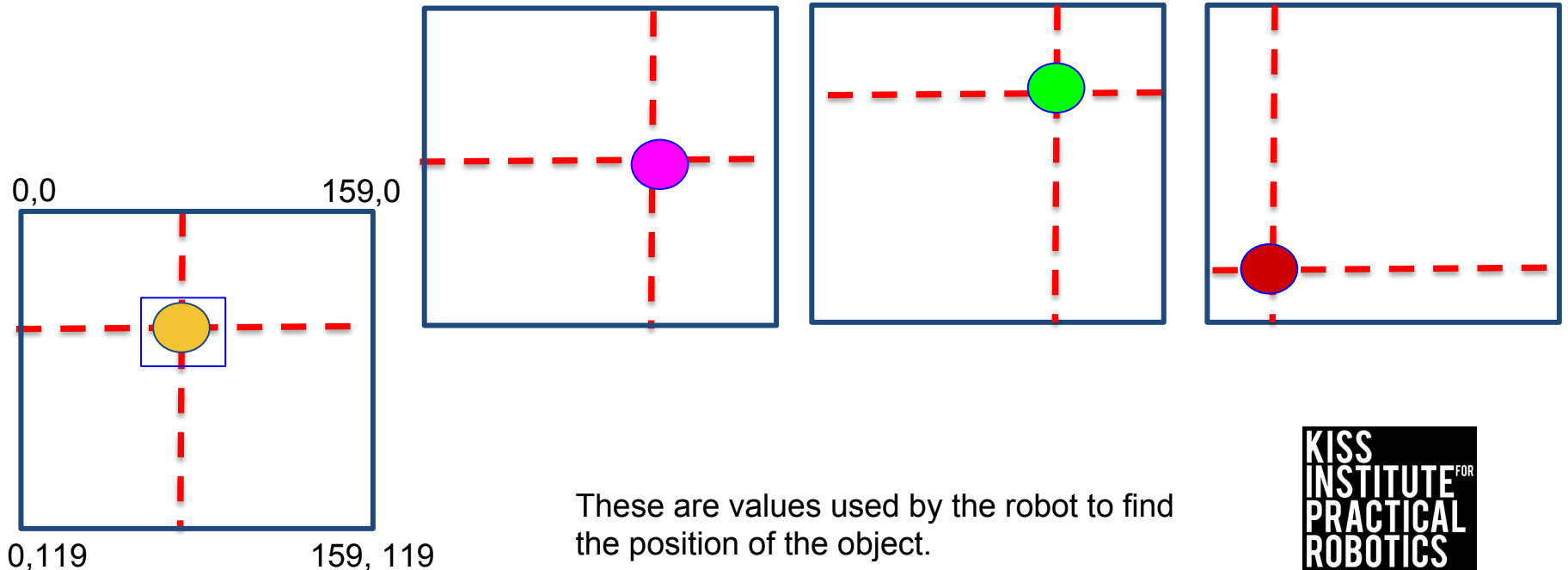
The program is asking the controller what value is between 0 and 159, (column) and what value is between 0 and 119 (row) to find the center of the object.

Remember the numbers at the top of the spreadsheet are smaller than at the bottom.

Where is the Object's Center?

Activity 6

1. Where is the yellow object on your spreadsheet? Write down the approximate column and row cell locations.
2. Where is the purple object on your spreadsheet? Write down the approximate column and row cell locations.
3. Where is the green object on your spreadsheet? Write down the approximate column and row cell locations.
4. Where is the red object on your spreadsheet? Write down the approximate column and row cell locations.



These are values used by the robot to find the position of the object.

Approximate answers

Yellow- 69,40

Pink- 110, 59

Green- 123,37

Red-25,100

Is the Object on the Left or Right

Activity 7

```
get_object_center_column(0,0) // tells the  
controller to find the column center of the object
```

- What would the code look like to determine if the object is to the left of the camera?
- What would the code look like to determine if the object is to the right?

Remember Boolean logic:

> Greater than

>= Greater than or equal

< Less than

<= Less than or equal

== Equal to

!= Not equal to

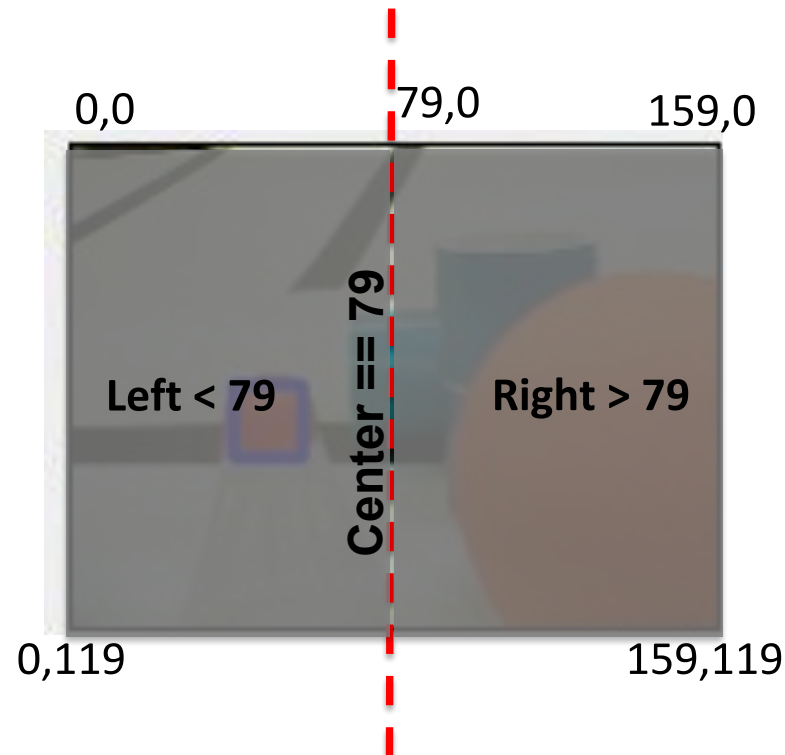
Is the Object on the Left or Right

Activity 7

Solution and Explanation:

```
get_object_center_column(0,0) < 70 //object on left
```

```
get_object_center_column(0,0) > 90 //object on right
```



Camera Functions Matching:

Activity 7

Match the functions to what they do.

<pre>if (get_object_center_column(0,0) < 70)</pre>	Gets the most recent image from the camera
<pre>while (a_button() == 0)</pre>	Turns on the camera
<pre>if (get_object_center_column(0,0) > 90)</pre>	Checks to see if the object is in the center of the image (straight ahead)
<pre>camera_open();</pre>	Checks to see if objects of the specified color are in the camera image
<pre>if ((get_object_center_column(0,0) >= 70) && (get_object_center_column(0,0) <= 90))</pre>	Checks to see if the object is to the right of the center of the image (to the right)
<pre>if (get_object_count(0) > 0)</pre>	Checks to see if the object is to the left of the center of the image (to the left)
<pre>camera_update();</pre>	Waits for the a_button to be pushed

Activity 7– Camera Functions Matching

Answers

<pre>if (get_object_center_column(0,0) < 70)</pre>	Gets the most recent image from the camera
<pre>while (a_button() == 0)</pre>	Turns on the camera
<pre>if (get_object_center_column(0,0) > 90)</pre>	Checks to see if the object is in the center of the image (straight ahead)
<pre>camera_open();</pre>	Checks to see if objects of the specified color are in the camera image
<pre>if ((get_object_center_column(0,0) >= 70) && (get_object_center_column(0,0) <= 90))</pre>	Checks to see if the object is to the right of the center of the image (to the right)
<pre>if (get_object_count(0) > 0)</pre>	Checks to see if the object is to the left of the center of the image (to the left)
<pre>camera_update();</pre>	Waits for the a_button to be pushed

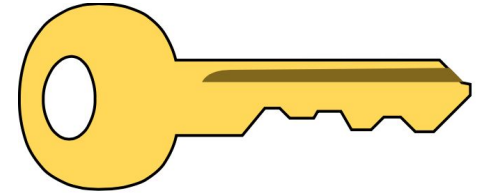
Find the Green Object - Activity 7

Goal: Use the camera to drive to the green object.

1. Use [code planning paper](#) to plan the steps in your code. [Color Tracking Review](#).
2. Your program must:
 - a. Find the green object
 - b. must go to the green object and stop
 - c. Your code uses an **if** statement with the Boolean value of < 79
 - d. Your code uses an **if** statement with the Boolean value > 79

Camera Activity Using **while** and **if**

Example of code planning sheet:



1. Open the camera
2. Create a while loop that checks the status of the a_button
3. Updates camera image
4. Is the object count greater than 0 (does it see the object?)
5. Motor turns left toward object (is the object on the left?)
6. Motor turns right toward object (is the object on the right?)
7. Else statement (no object found)
8. Stops when the A-button is pressed

*This is the same type of program as the line follow activity, but instead of the reflectance sensor, it is using the camera. Because it knows that 80 is the center of the image anything <70 is to the left, so turn left, anything ≥ 90 is to the right, so turn right, if it is in the middle 71-89 it won't turn.

Find the Green Object - Activity 7 Solution

```
#include <kipr/botball.h>

int main()
{
    camera_open(); // open the camera

    while (a_button() == 0) // checks the status of the a_button; stops when the a_button is pressed
    {
        camera_update(); // updates the camera image

        if (get_object_count(0) > 0) // is the object count greater than 0? (does it see the object?)
        {
            if (get_object_center_column(0,0) < 79) // is the object on the left?
            {
                motor(0,-50); // turn left
                motor(3,50);
            }

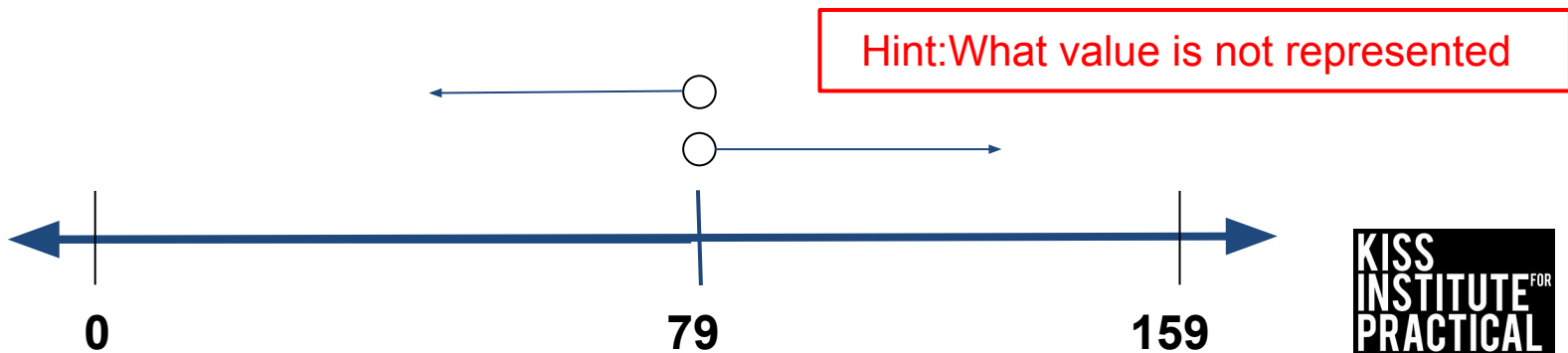
            if (get_object_center_column(0,0) > 79) // is the object on the right?
            {
                motor(0,50); // turn right
                motor(3,-50);
            }
        }
        else // the object count was not greater than 0 (no object found)
        {
            ao(); // turn off motors
        }
    }

    ao(); // turn off motors
    camera_close(); // open the camera
    return 0;
}
```

Drive to the Green Object - Activity 8

Goal: Use the camera to find the green object.

1. Use [code planning paper](#) to plan the steps in your program. [Color Tracking Review](#).
2. Your program must:
 - a. Find the green object
 - b. Your code should use 3 `if` statements.



Drive to the Green Object - Activity 8 Solution

```
#include <kipr/botball.h>

int main()
{
    camera_open(); // open the camera

    while (a_button() == 0) // checks the status of the a_button; stops when the a_button is pressed
    {
        camera_update(); // updates the camera image

        if (get_object_count(0) > 0) // is the object count greater than 0? (does it see the object?)
        {
            if (get_object_center_column(0,0) < 79) // is the object on the left?
            {
                motor(0,-50); // turn left
                motor(3,50);
            }

            if (get_object_center_column(0,0) > 79) // is the object on the right?
            {
                motor(0,50); // turn right
                motor(3,-50);
            }

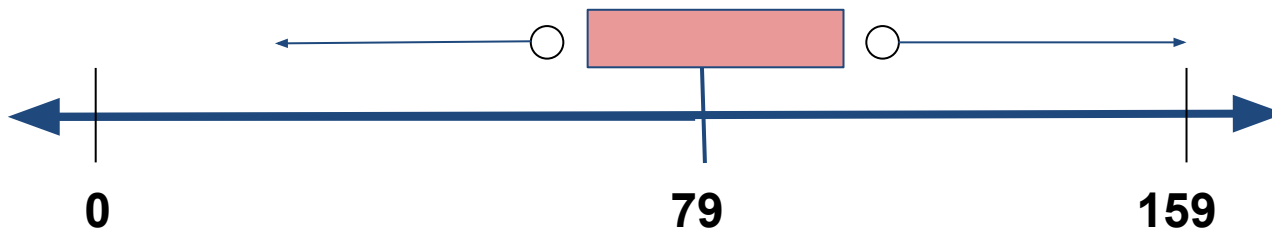
            if (get_object_center_column(0,0) == 79) // is the object centered in front of the camera?
            {
                motor(0,50); // drive forward
                motor(3,50);
            }
        }
        else // the object count was not greater than 0 (no object found)
        {
            ao(); // turn off motors
        }
    }

    ao(); // turn off motors
    camera_close(); // open the camera
    return 0;
}
```

Understanding Inequalities

Activity 9

1. Look at the number line below.
2. Write an inequality statement for the left (less than statement)
3. Write an inequality statement for the right (greater than statement)
4. How would you represent the pink box using inequalities?



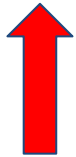
Proceed to the next slide

Understanding Inequalities: Activity 9- Solution

<70 left

>90 right

≥ 70 and ≤ 90 middle (pink box)

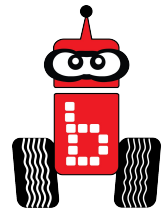


In programming we can not just write the word, “and”. Is there a Boolean statement that would help us write this?

Proceed to the next slide

Understanding Inequalities

Activity 9



> Greater than

$5 > 4$ is TRUE

< Less than

$4 < 5$ is TRUE

\geq Greater than or equal

$4 \geq 4$ is TRUE

\leq Less than or equal

$3 \leq 4$ is TRUE

$==$ Equal to

$5 == 5$ is TRUE

$!=$ Not equal to

$5 != 4$ is TRUE

$\&\&$ is AND

Which one is new?

Understanding Inequalities

Activity 9

&& is AND

Read and discuss the following with a partner.

It is used to link two conditions, in both **if** and **while** statements.

Such as:

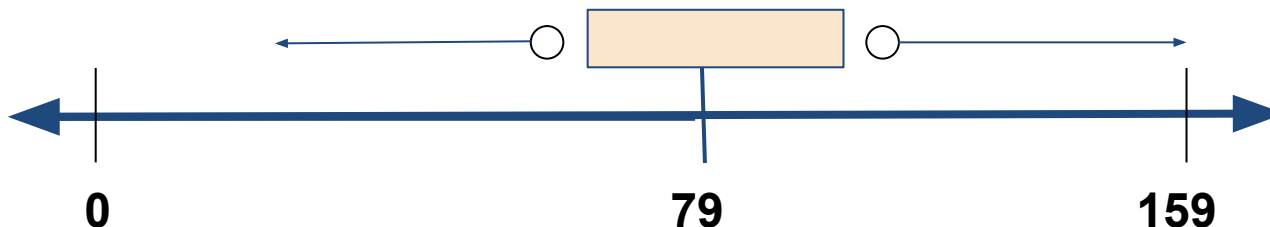
≥ 70 and ≤ 90 middle (pink box)

Driving Efficiently and Directly to the Green Object

Activity 10

Goal: Use the camera to drive to the green object in a more efficient and direct way to the line.

1. Use [code planning paper](#) to plan the steps in your code. [Color Tracking Review](#).
2. Your program must:
 - a. Find the green object
 - b. Must go to the green object and stop
 - c. Your code uses 3 **if** statements
 - d. One of the **if** statements boolean values will focus on the center of the image and drive directly to the object



Driving Efficiently and Directly to the Green Object

Activity 10 Solution

```
#include <kipr/botball.h>

int main()
{
    camera_open(); // open the camera

    while (a_button() == 0) // checks the status of the a_button; stops when the a_button is pressed
    {
        camera_update(); // updates the camera image

        if (get_object_count(0) > 0) // is the object count greater than 0? (does it see the object?)
        {
            if (get_object_center_column(0,0) < 69) // is the object on the left?
            {
                motor(0,-50); // turn left
                motor(3,50);
            }

            if (get_object_center_column(0,0) > 89) // is the object on the right?
            {
                motor(0,50); // turn right
                motor(3,-50);
            }

            if ((get_object_center_column(0,0) >= 69) && (get_object_center_column(0,0) <= 89)) // object centered?
            {
                motor(0,50); // drive forward
                motor(3,50);
            }
        }
        else // the object count was not greater than 0 (no object found)
        {
            ao(); // turn off motors
        }
    }

    ao(); // turn off motors
    camera_close(); // open the camera
    return 0;
}
```

Assessments and Rubrics



Suggestions: *Understanding* or *Group Collaboration* rubrics