PLTW COMPUTER SCIENCE

Activity 1.1.2

Algorithms and Coding Fundamentals: Happy Accelerometer

goals

- Learn coding fundamentals
- Apply file naming conventions and version control
- Develop and test an app incrementally
- Develop an app independently for creative expression

description of app

Create an app that allows the user to balance a happy face on the center of the screen using the accelerometer built into the Android™ device as inputs.

Essential Questions

1. How do I describe algorithms to someone new to computer science?
2. What mathematical and logical concepts have I seen before in my other classes?
3. What makes a computer science concept fundamental or essential?

Essential Concepts

- Algorithms, Variables, Arguments, Procedures, Strings and Concatenation, Data Types, and Logic
- Arithmetic Operators, Relational Operators, and Logical Operators

- Conditionals and Event-driven Programming

Resources

Activity 1.1.2 Student Files

Designing a New Project

Design Overview: Happy Accelerometer

User Story

Create an app that challenges a user to keep a sprite (happy face) in the center of the screen using the accelerometer. Note: in the next activity, you will make this more challenging for the user by asking the user to balance the tablet with only their foot.

App Overview
The Happy Accelerometer app will help you learn how the program keeps track of where images are located or placed on a screen. It is important for you to know this as you begin to create and personalize your apps. With this app, you will create labels on the user interface to show the location of the sprite and the value reported by the accelerometer.

Review the design concepts you need to consider when developing an app. While this activity suggests the design components and design, in future activities you will need to take a user story and create your own design.

Initial Backlog Breakdown
The user needs to be able to:

- ☐ Move the tablet and see motion
- ☐ See the coordinates of the moving object
- ☐ See the numerical representation of the tablet motion
-

1. Navigate to MIT App Inventor and log in.
2. Create a new project. Save the project using the naming convention set forth by your teacher. Example: *LastNameFirstInitial112HappyAcc1*

> Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Setting Up the User Interface in the Design View

> Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

3. Set up the *Screen Properties* so the app will have certain default settings for all users. For example, in this app, the user may need to scroll down to see all the features, and you don't want the tablet to change between portrait and landscape layouts.
    - Select **Screen1** in the Components list.
    - Check the box under **Scrollable** to make a scroll bar appear on the screen.
    - Change the *Title* to "HappyAccelerometer".
    - Change the *ScreenOrientation* to **Portrait** to keep the tablet from changing views.
    - Change the *Background* to a color you choose.

Based on the user's requirements, the user interface will need to include the following components:

- *AccelerometerSensor* – A non-visible component that can detect shaking and measure acceleration in approximately three dimensions. When you tilt the tablet, it tells the sprite which way to move.
- *ImageSprite* component – An image (happy face) that will move on the *Canvas* based on the tilt of the device.
- *Canvas* – A canvas is a two-dimensional, touch-sensitive, rectangular panel that the user can draw on and the sprite can move on.
- *Label* components (2) – A label component is a piece of text displayed on the user interface. You are going to use two labels to report values for where the sprite is and where it should be going.

As you put components into the *Designer* view from the *Palette* drawers, MIT App Inventor automatically gives you access to the correct programming blocks in the *Blocks* view.

4. Rename the components of the user interface so you know what each component is for in the *Blocks* view.
5. Add the *accelerometer sensor*. Keep in mind that this is a non-visible component and only shows up below the screen.

6. Add the *ImageSprite* that will move around the *Canvas*.
    - From the *Drawing and Animation* component drawer, drag the *Canvas* component onto your user interface.
    - After dragging it into the *Canvas*, select the **ImageSprite** in the *Components* list and set the *Properties* to:
        - Download the *HappyFace.png* file.
        - Use the Upload File box to upload the *HappyFace.png* image from the activity's source files.
        - In the Picture box, change from none to *HappyFace.png*.
        - Set the Width to "50" pixels.
        - Set the Height to "50" pixels.
        - Rename the *ImageSprite* to "HappyImage".

7. Add a *Canvas* to provide space for the image to move.
    - From the *Drawing and Animation Palette*, drag the component onto your user interface.
    - After dragging it in, select it in the *Components* list and set the properties as follows:
        - Set the Width to **Fill parent**.
        - Set the Height to "80" percent.

8. Add *Labels* to read the location of the sprite and the value reported by the accelerometer.

    Note: Later in this activity, you will want to know where the *HappyImage* is and the value for the *accelerometer*. You will create *Labels* on *Screen1* to report this information to the user.

    In the *User Interface* component drawer, rename the *Labels*. Suggested names for these labels are:

    - *LocationLabel*
    - *AccelerometerLabel*

    Important: Remember, if you do not add the components in *Designer* view, you might be confused as to why you can't find the corresponding programming blocks when you switch to *Blocks* view.

    The activities you are doing use the names described above. If you don't change the names as described, you might have trouble finding the blocks later.

9. Map the width and height of the *Canvas*.

    Notice the boxes (*X*, *Y*, and *Z*) in the *HappyImage* properties. Set a value for each to put the image in the middle of the screen.

PLTW DEVELOPER'S JOURNAL

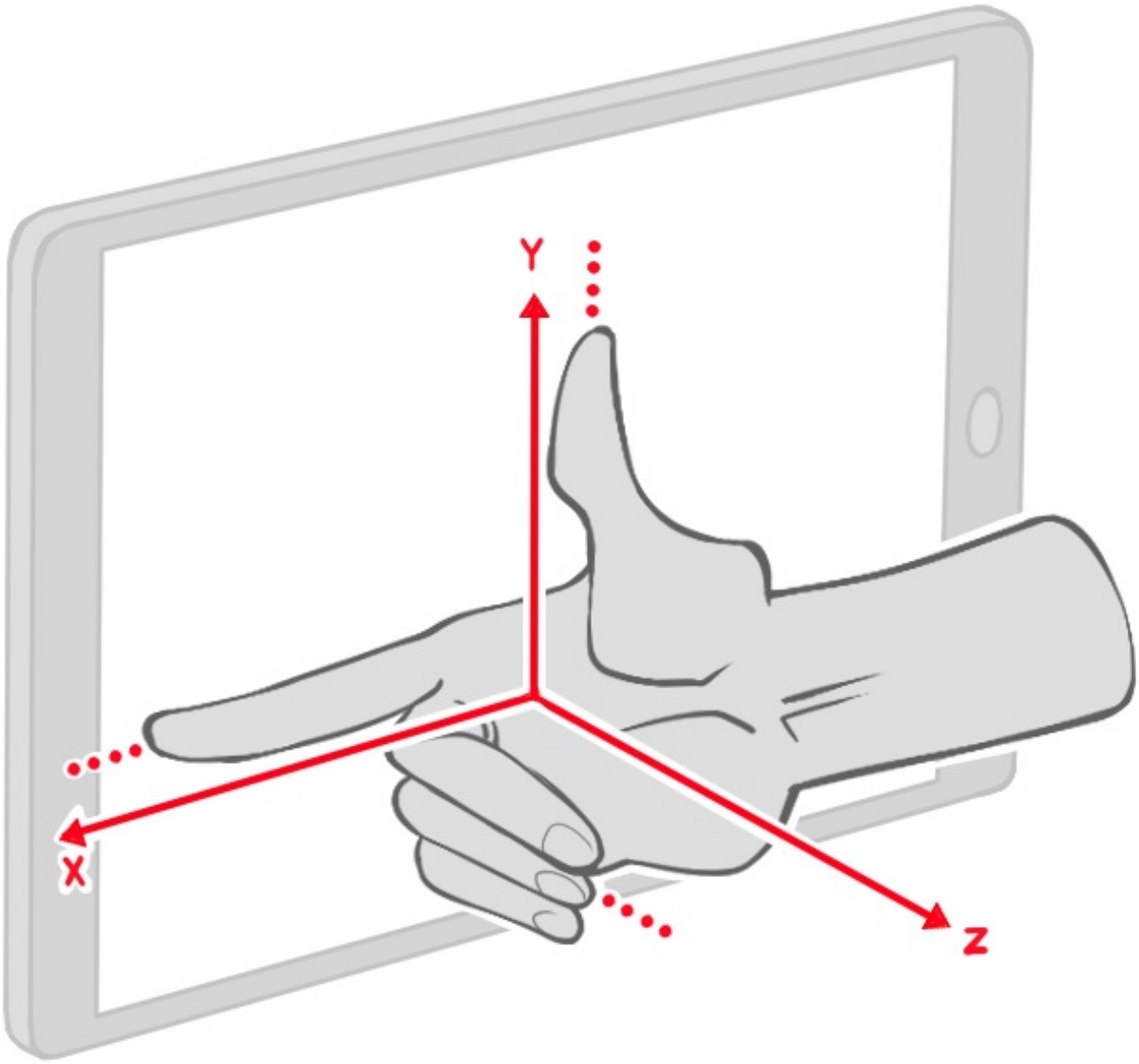
1. Change the values for X. Can you figure out the range for the X-axis where the sprite can be located?
2. Try the same thing with the Y-axis values to see if you can map what numbers

correspond to locations on the *Canvas*.
3. Changing Z has no effect. Why do you think that is?

3D Axis Coordinate System (X, Y, Z): Helpful System Visual

An easy way to keep track of the directions associated with the X-axis, Y-axis, and Z-axis is the “right-hand rule”.

1. Make an “L” with your index finger and thumb of your right hand.
2. Lay the back of your hand on the tablet with your thumb pointing up.
3. Now bend your remaining fingers to make a 90-degree angle with your thumb and index finger (your remaining fingers should be pointing back at you).



- Your thumb represents the Y-axis (up and down on the tablet).
- Your index finger represents the X-axis (side to side on the tablet).
- Your remaining fingers represent the Z-axis (going into and coming out of the screen). This is why changing the value made no noticeable difference.

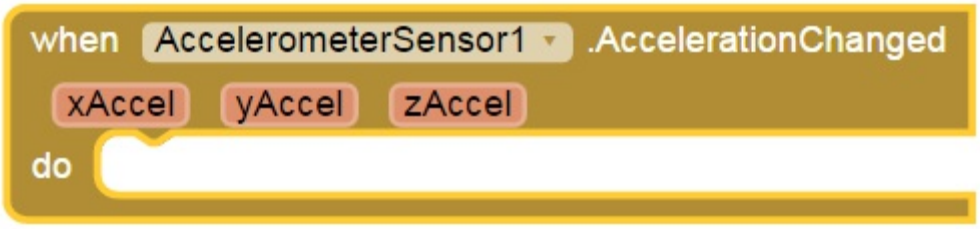
Variables and Arguments for the Accelerometer

The Blocks View Setup

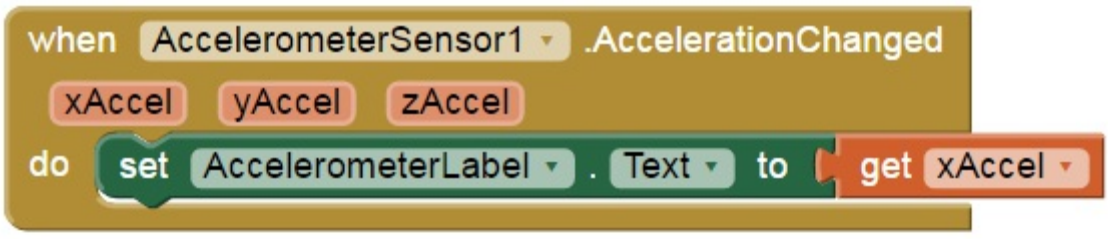
Many features and functions of technology are defined by numerical values. While you are tilting a device to the right, the device senses the movement and puts a numerical value to the tilt. To harness these features, it's important to understand the values the device has in relationship to the movements.

You are going to get to know the accelerometer a little better by learning the values of the tilt. Knowing the tilt values will help you use the accelerometer values in apps to make events happen based on how the user is holding the app.

10. Switch to *Blocks* view.
11. Click the *AccelerometerSensor1* component in the *Blocks* drawer and drag the event handler for *AccelerationChanged* into the *viewer*.



12. Click your *AccelerometerLabel* and drag the set *AccelerometerLabel.Text* block into the event handler.
13. Looking at the event handler, you should see three orange blocks. These orange blocks represent **variables**. Hover your pointer over each variable block.
14. Select the *get* block for the *xAccel* variable and drag it out to add to your *set* block.



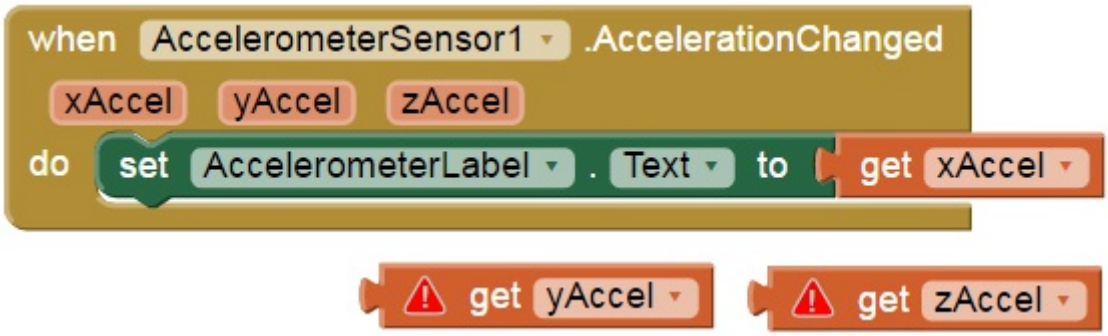
15. Test your app by holding the tablet parallel with the floor and tilting it side to side (along the x-axis).

PLTW DEVELOPER'S JOURNAL

1. What do you see in your *AccelerometerLabel*?

2. Do the values reported make sense?

Understanding Variables and Arguments



Your accelerometer is showing you data you can use. You will use this data in an event handler to make the sprite move in a certain direction.

These variables (orange boxes) are like containers that can hold values that are changing. The actual numbers or values in these orange blocks are **arguments**. The arguments get passed along to be used by other parts of a program.

In this app, these arguments get the information about how the accelerometer has changed. The event handler knows what the current value is from the argument value and gives appropriate outputs, which makes the happy face move in a certain direction.

Using event handlers to make decisions based on the arguments passed to the event handler is a fundamental idea in coding.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Data Types

So what does the accelerometer really tell the program? What do those inputs look like? Data used in a program often has a numerical value, but there are different kinds of numbers. Sometimes using words makes more sense than using numbers.

As you develop apps, think about: What kinds of data will you be working with in your programs and how can you translate one type of data to another that makes more sense for what you are doing?

Many types of data are used in programming. You will learn about many of them in this course. Here are four essential data types that provide the information you need to understand how the accelerometer is used in the program:

Data Type	Description	Example
integer	A whole number that does not have a decimal or any digits after the decimal.	(2)

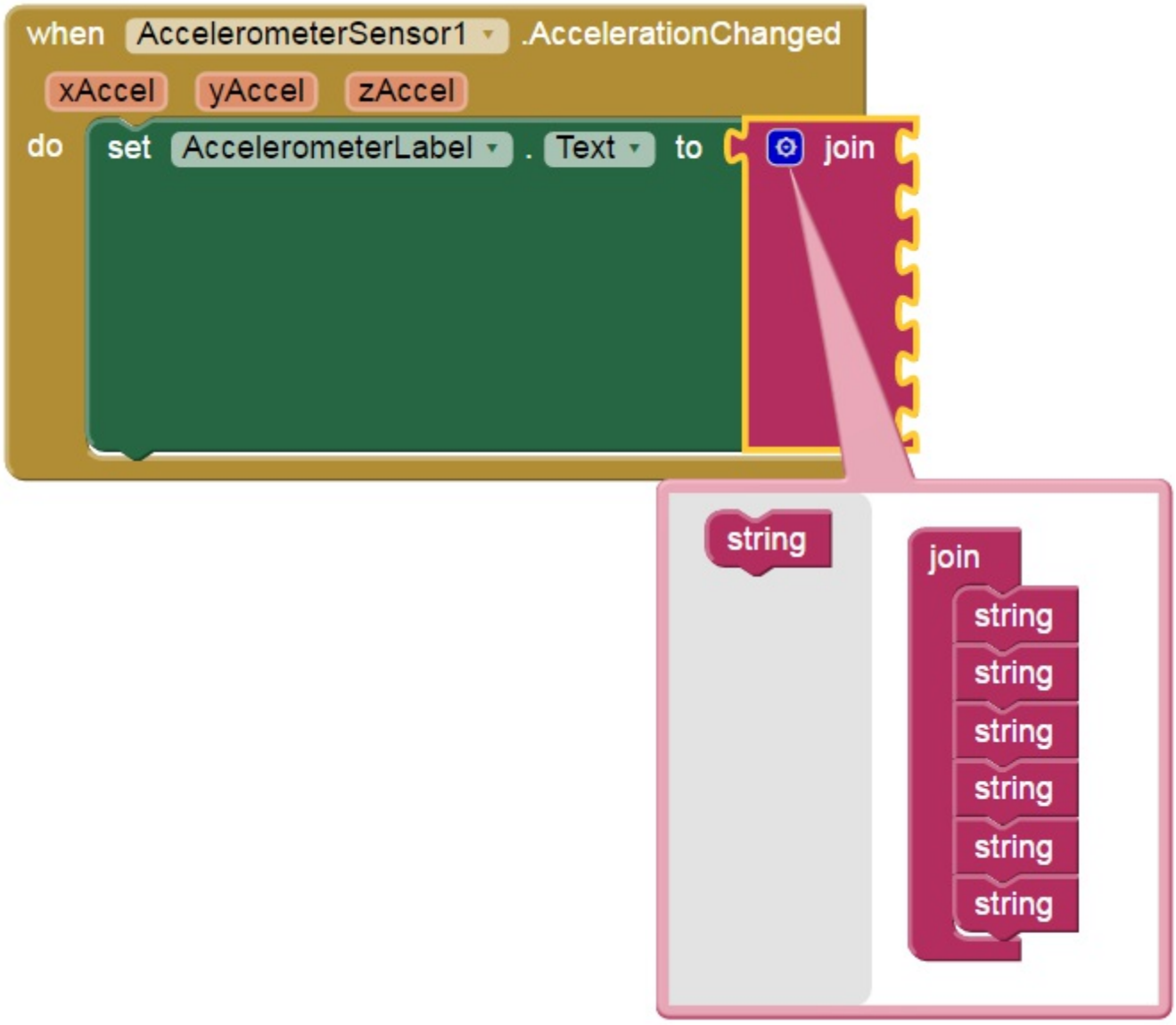
float	A type of number that provides very precise information by including all the numbers after the decimal.	(6.8345)
strings	Text or characters displayed by a program. In MIT App Inventor *Blocks* view, the text string block lets you manage how text, also known as **strings** are seen by the user of an app.	“hello”
Boolean	A data type that only has two values, usually denoted as true or false.	(True)

Concatenation of Data

The accelerometer provides long float values. To help the user make sense of those numbers, you can combine the numbers with strings (text) that describe which axis the number represents all in one label.

Concatenation is joining together, without changing, separate items into one—in this case, within a single label. Concatenation is often associated with joining text strings together, so in App Inventor you will find the *join* block in the *Text Drawer*.

16. Add a *join* block from the *Text* drawer in *Blocks* view.
17. Use the blue (**Mutator**) button to add more inputs.
    - In the *mutator* box, click the individual *string* block (on the left of the box) and drag the *string* block into the *join* block to create a new input.
    - Create six inputs: three for the text that will identify the axis and three for the float values that describe the tilt of the device.



18. Add three text strings from the Text blocks. Make sure you put in a space (replace the _ with a blank space) before the (X, Y, and Z characters).

    - " _X-axis:_"
    - " _Y-axis:_"
    - " _Z-axis:_"

    Important: Notice the space before the ending ".

    Whatever is between the " " will appear exactly that way. To make it easier to read, you can add spaces within your strings.

19. Add back the *get* blocks to return the variables for:

    - *xAccel*
    - *yAccel*
    - *zAccel*

Important: Be sure to place the *get* blocks after the appropriate string labels.

20. Test your app.
    - ☐ As you tilt the tablet in different directions, the numbers change to show the numerical value of the accelerometer based on the tilt of the X-axis, the Y-axis, and the Z-axis, all at the same time.
    - ☐ The numbers show as expected.
    - ☐ Each number can be identified with an axis.

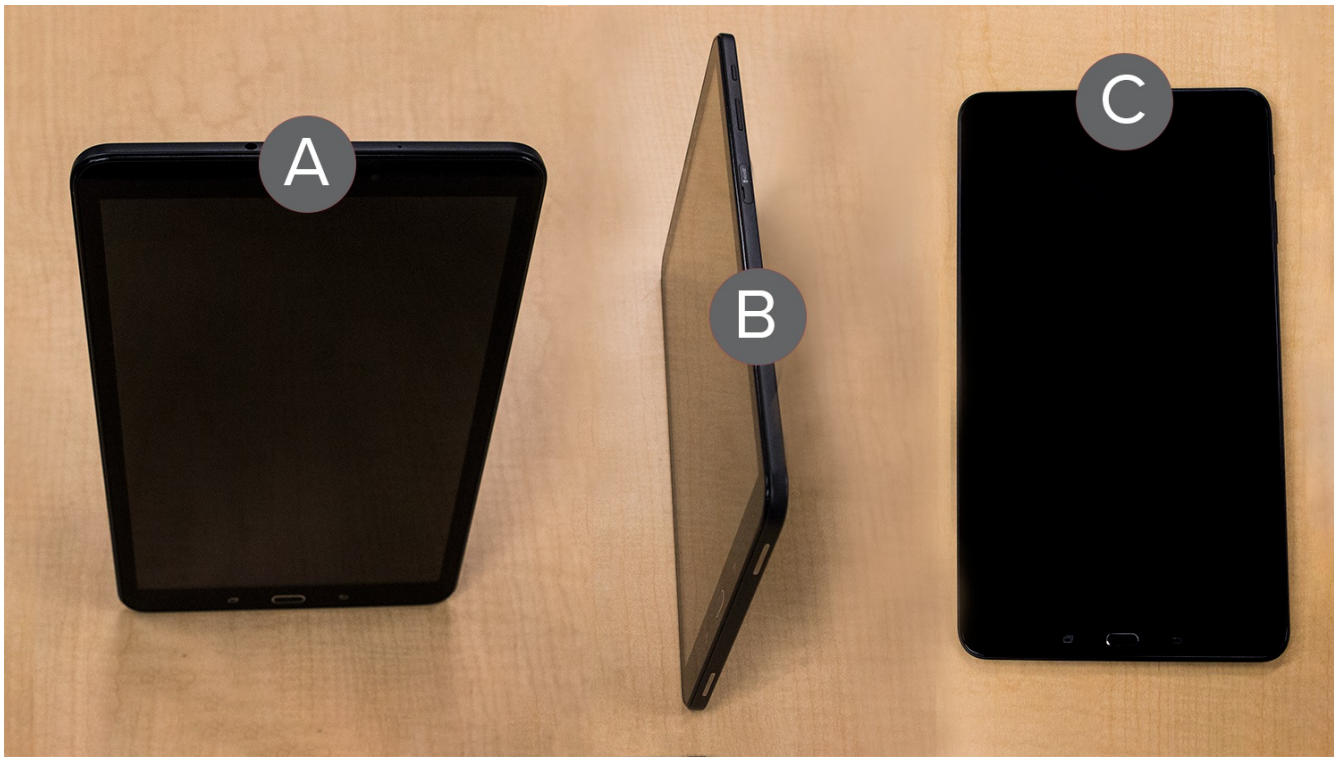
> Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

21. In the *Math* drawer, there is a built-in block to make a float number become an integer. Find that block and use it to modify your program. The block is a common math term.

    Hint: Think about the difference between what you sometimes get on the calculator and what you turn in as your answer in class. What does your math teacher ask you to do with float numbers to get integers?

22. After adding the block to get integers instead of float type numbers as the output, move the tablet around to play with the values.

PLTW DEVELOPER'S JOURNAL Draw and identify the rounded values of X, Y, Z for each of the tablet positions in the picture.



23. You have completed the first part of the app and need to save your progress before further development.
    - Under **Projects**, select **Save project as**.
    - Name your project, adding a version number as described by your teacher. For example, your teacher may tell you to add “v2” to the end of your current app name.

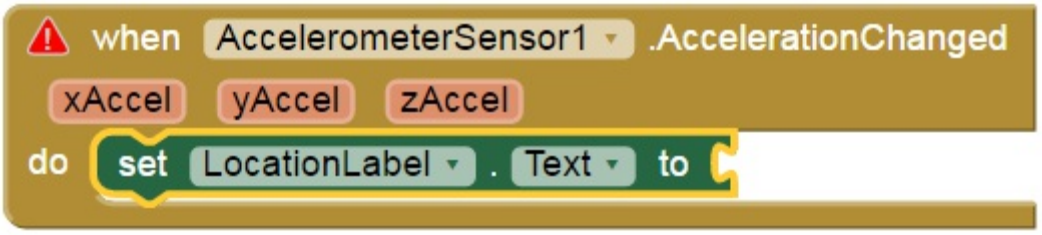
    Important: Saving and changing the name can help you track what each app does if you need to go back and revisit a different part that was working before iteration.

Tracking the Happy Face

In the second part of this app development process, you will create code to track where your *HappyImage* is on the screen. By knowing these values, you can add complexity to your balance game based on where the *HappyImage* is located. The goal is for the *HappyImage* to move in the direction the device is tilted.

- Use the *AccelerometerSensor1* to get the initial location of the *HappyImage* sprite when the app starts.
- You need the location of the *HappyImage* every time it moves.
- The X-axis and Y-axis will tell the *HappyImage* which way to move.

24. Click the **LocationLabel** and drag the set *LocationLabel.Text* block inside the *AccelerometerSensor.AccelerationChanged* event handler.



25. Just as you did with the accelerometer values, use concatenation to create a message that reports the values for the X and Y positions in a sentence. For example, the message might read, "The image location is 55: x-axis and 98: y-axis." The coordinates of the *HappyImage* are called:

    - *HappyImage* X
    - *HappyImage* Y

    Use concatenation within the *LocationLabel.Text* to join the integer values of the X and Y, along with labels to identify each axis.

26. Play with adding different numbers of spaces in different locations inside the " " blocks. Make the *LocationLabel* coordinates appear neatly.

27. Test your app on the tablet.

PLTW DEVELOPER'S JOURNAL

1. What do you notice on the screen as you add and remove spaces in different places?
2. Is there a way to add spaces before the Y label without having another string of just spaces?
3. Did the *happyImage* sprite move where you thought it would?

Having played with App Inventor for a while, you might have figured out the device's coordinate system. If not, here is your chance to learn the numbers you can use to make the app do different things based on the location of the *happyImage*.

PLTW DEVELOPER'S JOURNAL

Can you describe where the origin (0,0) is?

Draw the coordinate grid and label the (X,Y) coordinates on the screen. The graph paper in your journal can help you make the image neat. Include specific coordinate labels.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Procedures: Making the happyImage Sprite Move

Now that you can see the *happyImage* and understand the accelerometer a little better, you will make the *happyImage* move with the accelerometer!

To do this, you are going to add a procedure to the *when AcceleromeeteSensor1.AccelermometerChanged* event handler. The procedure will place the *HappyImage* at the last given (X,Y) coordinate given by our accelerometer.

In computer science, the action of using a procedure built outside the current code block is referred to as **calling a procedure** because you are instructing the program to a specific procedure to execute at a specific place in the code, even though the procedure is not completely outlined there.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

28. Click the *HappyImage* sprite and pull the *.MoveTo* call block into the *AccelerometerSensor1.AccelermometerChanged* event handler.

29. Click to open the *AccelerometerSensor1* drawer:

    - Drag in the *get AccelerometerSensor1.XAccel* value.
    - Drag in the *get AccelerometerSensor1.YAccel* value.
    - The *HappyImage* moves along the X-axis by changing the X value of the *HappyImage* based on the *xAccel accelerometer* value.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

The procedure moves the *happyImage* to the initial X and Y positions and then moves the image based on the accelerometer's changes.

The problem is that the values the program gets from the accelerometer are only integers from –10 to 10. Where is the only place in the canvas your happy face will move?

You need to change the (X,Y) value to larger numbers and numbers that reflect which way the tablet is tilted. To do this you will use mathematical operators.

Operators

There are a few types of mathematical operators: **arithmetic operators**, relational operators, and **logical operators**.

Arithmetic Operator	**The same ones you use in math class such as + (addition), – (subtraction), x (multiplication), and ÷ (division).**
Relational Operator	< (less than), > (greater than), = (equals)
Logical Operator	Conditional statements like (True, False).

Note: Later on, you will learn more about Boolean type data and logical operators such as AND, OR, and NOT.

Refer to your downloadable resources for this material. Interactive content may not be	Refer to your downloadable resources for this material. Interactive content may not be

available in the PDF edition of this course.	available in the PDF edition of this course.

Reminder: The accelerometer returns integers from –10 to 10. So sometimes the program adds the value, sometimes it subtracts.

```
5 + 5 = 10
5 + (-5) is the same as 5 - 5 = 0
```

30. In the *Math* blocks drawer, find the operator that will return the sum of two numbers. Add this operator so that the number returned to the *Move.To* procedure for both X and Y is the last location returned, plus the new value of the *AccelerometerSensor*. __ *Accel*.

What should happen when the block is filled in correctly?

As you move the device, the *happyImage* should move across the screen.

31. Does your app function the way you want it to? Perhaps the movement does not quite work the way you thought it should in one axis.
    - Consider the motion values of the accelerometer in comparison to the tilt.
    - What math operator can you use to change the value of the incorrectly operating axis?

    Reminder: You want to get the *HappyImage* to move with the tilt of the device, instead of opposite the tilt, without changing the current *HappyImage* location or the accelerometer's number values.

32. Add an operator that will invert the numerical value of the axis value that is not working.

    - Put in the number that performs the math operation to invert the axis motion (change a positive number to a negative number).

33. Test your code, and continue adjusting the multiplier to get the effect you want.

> Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

34. Show your completed app to your teacher.

35. If your teacher directs, do an iteration save and try modifying your app. For example, you could adjust the speed by adding some additional arithmetic operators. Hint: you can speed up with multiplication or slow down with division.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Conclusion

1. What is an algorithm?
2. What are some arithmetic operators, relational operators, and logical operators you learned about in this activity?

3. What are some data types were you introduced to in this activity?

4. How did you interpret and respond to the **essential questions**? Capture your thoughts for future conversations.