STEM DISCOVER

STEM EDA

EXPLORE • DISCOVER • APPLY

STUDENT EDITION

CODING

STEM EDA

EXPLORE • DISCOVER • APPLY

Welcome to STEM EDA!

STEM Explore, Discover, Apply (STEM EDA) is designed as a three course progression through STEM (science, technology, engineering, and mathematics) topics. Students begin by exploring STEM concepts (STEM Explore, 6th grade), then transition to discovering fundamental concepts (STEM Discover, 7th grade), followed by the application of the concepts (STEM Apply, 8th grade). Throughout each course in the sequence, the engineering design process guides the students through the design and implementation of the projects and concepts.

Grade Progression

6th STEM Explore | 7th STEM Discover | 8th STEM Apply

What is STEM EDA?

STEM EDA engages middle school students through a series of hands-on projects that help improve their problem-solving and critical-thinking skills. All projects seamlessly integrate the engineering design process which allows students to creatively explore STEM through design.

This multi-grade level curricula utilizes liberal arts disciplines to provide meaning and depth to the content. Through STEM EDA, students develop invaluable skills focusing on leadership, team-building, creativity, and communication.

STEM EDA's modular nature provides ultimate flexibility to schools. Teachers can implement the curricula as a standalone elective course, insert specific modules into an existing class, or provide the modules as an after school program.

Goals of Course

- Foster excitement for STEM
- Develop a level of exploration in middle school students through STEM projects
- Provide a context for the engineering design process through "classic" STEM projects
- Drive towards fundamental concepts (at the grade appropriate level) through STEM experiences

> *"There is an energy here that is very contagious! My students are motivated and excited to come to school and work on this module. Students who were unmotivated and uninvolved are now key players in their small groups and have found an interest in academics they didn't think they had."*
>
> *-Middle School Teacher*

A Few Things You Will Notice

The stop sign indicates the end of a section and is a good/suggested stopping place. This symbol is visible in both the student edition and the teacher manual.

Cyber Pop Outs connect the STEM topic to the cyber world.

GREEN BOX

Definitions and notes will be pointed out to the students within this area.

CALCULATION BOX

Space for students to work problems.

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number, 2013-PD-127-000001, Modification #2. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

CREATED BY

NICERC™

AN ACADEMIC DIVISION OF THE CYBER INNOVATION CENTER

EDP | ENGINEERING DESIGN PROCESS

USING DESIGN TO DRIVE CURRICULAR EXPLORATION

The process contains overarching themes as well as defined steps! Use the process as a quick reference throughout the module.

- ***Iteration*** – Revisiting steps provides the opportunity to improve upon designs.
- ***Communication*** – Within a design team, communication is essential to reach an agreement on a solution.
- ***Teamwork*** – Group cooperation provides diverse perspectives and help in accomplishing goals.
- ***Creativity*** – STEM and liberal arts disciplines are integrated to encourage unique solutions.
- ***Imagination*** – Opportunity to apply creative thoughts during development offers unlimited options.

STEM DISCOVER™

Coding

CREATED BY

NICERC™

AN ACADEMIC DIVISION OF THE CYBER INNOVATION CENTER

Materials List

Per Team

Materials	Department
Computer with access to www.scratch.mit.edu	Classroom
Pencils	Classroom
Colored pencils	Classroom
Extra paper	Classroom

STEP 1: *IDENTIFYING THE PROBLEM*

You are a game developer at a video game company. Your company has decided to make a new game that will be available on all of the major video game consoles and computer operating systems. To build hype for the next multi-platform video game, your company has asked you to make a free demo that will be available online. Additionally, the characters and plot of the game have yet to be chosen, and the company wants your input. The company has asked you to build the demo using Scratch, a graphics-based programming language.

In your own words, identify the problem. _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

STEP 2: *RESEARCHING THE PROBLEM*

In your role as a game developer, you will create a video game using the Scratch programming language. What are some areas you should research?

_____

_____

_____

_____

Other topics that will be helpful to research are:

- What is a programming language?
- What are the different types of programming languages?
- What is Scratch?
- How do you use Scratch?
- What makes a video game successful?

In the sections that follow, you will research some of the topics mentioned above and learn how to create several games with Scratch. Let's start with the broadest of those topics, programming languages.

Programming Languages

In your own words, what is programming? _____

_____

_____

What do you think a programming language is? _____

_____

_____

A ***programming language*** is a language used to create instructions that are interpreted by a computer. Based on this definition, what do you think a program is? _____

_____

_____

As the definition of programming language suggests, a program is the list of instructions that the computer would follow.

Have you ever used a programming language to program before? If so, what was the language(s)? _____

_____

_____

Now that you know what a programming language is, let's do some research on the different types of programming languages and learn where they are typically used.

Types of Languages

There are many types of programming languages available for use. In fact, one specific language may fall into several different categories of programming languages. On the following pages, you will research the different types of programming languages listed below. For each type, list three identifying factors that describe it. Also, list three to five examples of a programming language in each category and give a brief description of that language.

Types of Languages to Research:

- Object-Oriented Programming Languages
- Educational Programming Languages
- Compiled Languages
- Graphics-Based Languages
- Curly-Bracket Languages

Object-Oriented Programming Languages

Identifying Factors:

1. _____
2. _____
3. _____

Examples of Object-Oriented Languages with Descriptions:

1. _____
2. _____
3. _____
4. _____
5. _____

Educational Programming Languages

Identifying Factors:

1. _____
2. _____
3. _____

Examples of Educational Programming Languages with Descriptions:

1. _____
2. _____
3. _____
4. _____
5. _____

Compiled Languages

Identifying Factors:

1. _____
2. _____
3. _____

Examples of Compiled Languages with Descriptions:

1. _____
2. _____
3. _____
4. _____
5. _____

Graphics-Based Languages

Identifying Factors:

1. _____
2. _____
3. _____

Examples of Graphics-Based Languages with Descriptions:

1. _____
2. _____
3. _____
4. _____
5. _____

Did you know that in their most simple terms, a computer is just a collection of "on/off" switches? Creating a specific sequence of to turn certain transistors on and off at certain times allows a computer to do different things. On and off are represented by either a 1 or a 0, binary code. However, it is extremely difficult to program in only 1's and 0's. Computers do not understand spoken or written words. Coding is a specific programming language used to tell a computer what you want it to do step-by-step. This code is then translated into binary code that the computer "understands".

a. What is binary code?

b. Are there different types of coding language? If so, can you name some examples?

c. Why do programmers not just write everything in 1's and 0's?

Curly-Bracket Languages

Identifying Factors:

1. _____
2. _____
3. _____

Examples of Curly-Bracket Languages with Descriptions:

1. _____
2. _____
3. _____
4. _____
5. _____

Scratch

You read about various types of programming languages, and now let's learn about Scratch, the language you will be using to create video games. Scratch is both an educational programming language and a graphics-based language because it has a "drag and drop the blocks" interface that is used to teach programming. Scratch is also considered an object-oriented programming language due to the nature of how programs are written. When programming in Scratch, you program objects called sprites using graphics-based instructions. The sprites typically represent the characters or objects that are interacting within the game. To make the sprites do what you want, you instruct the computer to control the sprites based on different actions performed by a user or based on actions happening within the program itself. The following sections will show you how to use many features within Scratch while simultaneously teaching you several fundamentals of programming such as logic and variables.

Before you begin creating anything in Scratch, you will need a framework to follow while programming. You will use the Programming Design Process (PDP).

The Programming Design Process (PDP) is a problem solving process that helps you organize your thoughts and guide your actions when writing a program. You can apply this process to both large and small problems. You will use the programming design process as a guide as you learn to program the sprites in this module. Let's review the meaning of each step of the PDP to gain a better understanding of how this process can be used as a guide.

Identify the Goal

In a similar fashion to the Engineering Design Process (EDP), the first step of the Programming Design Process is to identify the goal. Why do you think it is important to identify the goal of the program you are writing? _____

_____

_____

_____

Identifying the goal gives you a clear statement that you can refer back to if you lose track of what it is you are trying to accomplish. Another advantage of identifying the problem is that you have a statement that ensures everybody working on your team is trying to accomplish the same goal. Why do you think this is beneficial? _____

_____

_____

Design a Solution

Designing a solution is typically done on paper in the form of pseudocode. What do you think pseudocode is? _____

_____

_____

Based on the two parts of the word, *pseudo* and *code*, you might have guessed that ***pseudocode*** is not actual code but instead is a description of how the code of a program functions. Since it is not a formalized program, it is intended to be read by humans rather than computers. Pseudocode can take many forms such as a flow chart, a descriptive paragraph, a set of images, or a combination of these and others!

While forming programs in this module, you will use a form of pseudocode that combines flow charts and descriptions of what the code does during each step.

3 Implement the Solution

In Step 3 of the PDP, you input your code into the computer to match the pseudocode you created in Step 2, Design a Solution. In other words, you are turning your pseudocode into code. As you will see with Scratch, this is a matter of dragging and dropping various icons and changing some numbers so that each sprite behaves in the manner you want.

4 Run and Evaluate the Program

Step 4: Run and Evaluate the Program is the step where you test the section of code you just input into the computer. As you run the program, you will evaluate whether or not the program performed as expected. If the program performs as expected, then you move forward to Step 5. However, if it did not accomplish the goal, then you return to Step 2: Design a Solution.

If you need to return to Step 2, the first thing you should do is analyze your pseudocode and ask yourself whether or not it accurately describes how the goal will be met. If it does not, do this, you will need to make changes. Then, you move on to Step 3, make matching changes to your code, and return to Step 4 to run and evaluate the program once more.

If you returned to Step 2 and your pseudocode matches how you expect the program to perform, you then move on to Step 3 and check for errors in the code. This is also an excellent time to make minor tweaks to see how they may affect the program. You might discover that a piece of code you input does not actually do what you expect it to do.

5 Customize the Program

Once the program accomplishes the goal, you can move on to Step 5: Customize the Program. As you begin this step, ask yourself the following questions.

- "How can I improve this program?"
- "Can the program do the same thing with fewer lines of code or fewer programming blocks?"
- "Would the program function better if I added a certain feature?"

Since you are creating video games, these are other questions you might ask yourself.

- "Should this be the only character that does this action?"
- "Can I make the sprite face a different direction while doing this?"
- "Can I add animation to the character while it performs this action?"

Once you decide how to customize the program, you might make a small adjustment to the program or you might start the Programming Design Process again with a new goal in mind!

Bananas Meow

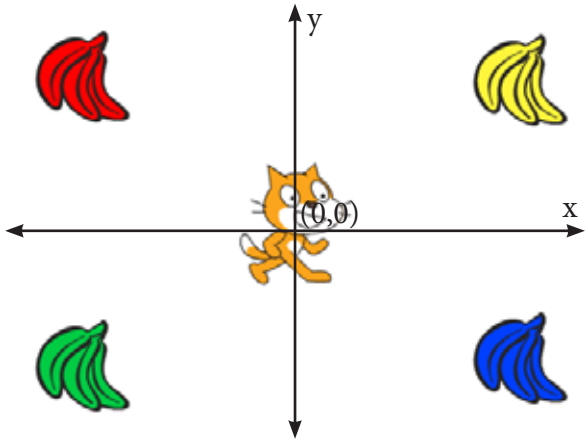
Now that you have the Programming Design Process (PDP) as a framework to use to program, let's begin using Scratch by creating a simple game called Bananas Meow.

The game can be found at https://scratch.mit.edu/projects/54969802/. Let's see how Bananas Meow functions by playing the game!

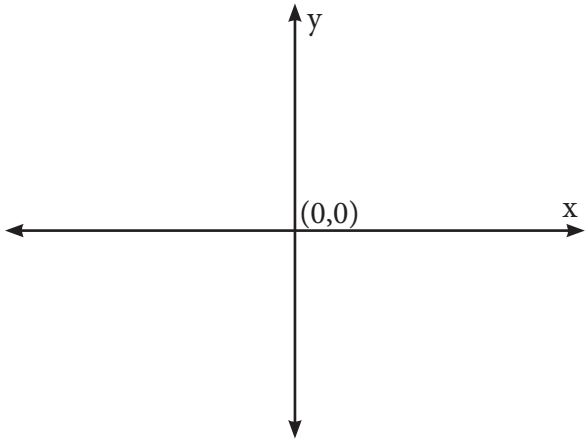
Now that you have played Bananas Meow, let's learn how to create it. You will start by learning how it functions through two fundamentals of mathematics that are used within the game: the rectangular grid and logic.

The Rectangular Grid

Everything in Scratch is built on a rectangular grid. The grid has a vertical y-axis and a horizontal x-axis. The intersection of the two axes is the origin. The origin is represented by the ordered pair (0,0).

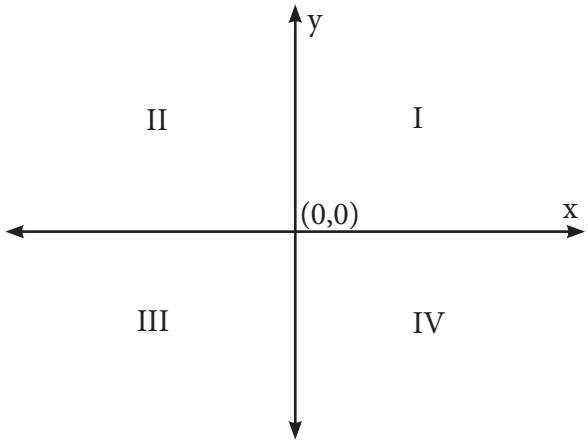


If the Bananas Meow game is removed from the background, you can more easily see the grid.



Each sprite on the grid is set to start at a specific spot. At which point do you think the cat in Bananas Meow is set to start? _____

The grid can also be partitioned into four different zones called quadrants. The quadrants are typically labeled and named with roman numerals I, II, III, and IV as shown below.



Referring to the image on the previous page, in which quadrant are the red bananas located?

_____

In which quadrant is the blue banana located? _____

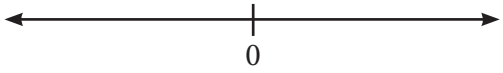
In which quadrant is the yellow banana located? _____

In which quadrant is the green banana located? _____

Earlier, you determined that the cat was located at (0,0) because it is at the origin. Just like the cat, every object can be represented by a specific position on the grid. If the position is **on** the x axis, then the x value will be zero. If the position is **on** the y axis, then the y value is zero. As it applies to the cat being at (0,0), it is located on both the x and y axis.

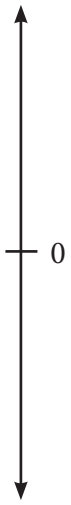
The bananas, however, are not located on the axes, but as you determined earlier, they are in quadrants I, II, III, and IV. To determine the location of these items, let's break down the grid to two separate number lines: a horizontal one and a vertical one.

On the horizontal axis below, put a plus (+) sign above the end you think is usually denoted as positive and a minus (-) on the end you think is usually denoted as negative.



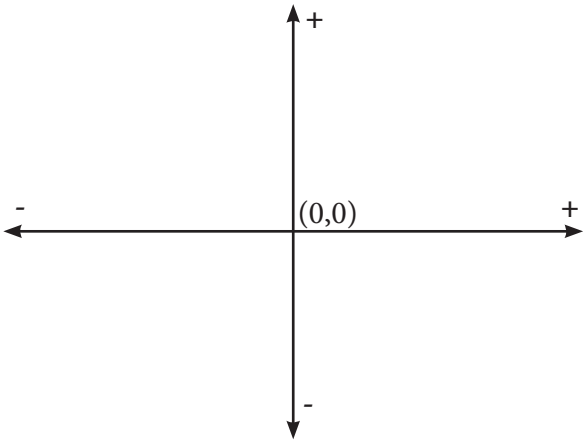
On the vertical number line below, put a plus (+) sign above the end that you think is usually denoted as positive and a minus (-) on the end that you think is usually denoted as negative.

Now if you combine the horizontal and vertical number lines at 0, you have the rectangular



coordinate grid with positive and negative directions labeled as shown below.

Using the information above, write in the quadrant(s) names in the blanks to the left of the



description of the quadrant(s).

Quadrants	Description
	x values are always positive
	y values are always positive
	x values are always negative
	y values are always negative
	x and *y* values are positive
	x and *y* values are negative
	x values are positive but *y* values are negative
	x values are negative but *y* values are positive

Based on the previous information about the quadrants where *x* and *y* values are positive and negative, write the color of the banana that could be represented by the *x* and *y* values below.

Banana Color	Values
	x = 182 y = -110
	x = 182 y = 119
	x = -179 y = 122
	x = -179 y = -116

Now that you understand where *x* and *y* values are positive and negative on a rectangular grid, let's explore the mathematical concepts that explain how sprites move across the game. Let's use the cat sprite as an example and start with an *x* value of 0. If you wanted the cat sprite to move to the right by 10 units, how would that *x* value change?

_____

_____

What if you wanted the cat sprite to move 10 units to the left? How do you think the *x* value would change? _____

_____

_____

If changing the *x* value causes the cat sprite to move horizontally (to the left or to the right), then what do you think you would change to make the cat sprite move vertically? _____

_____

How do you think the *y* value would change if you wanted the cat sprite to move 10 units upward? _____

_____

_____

How do you think the *y* value would change if you wanted the cat sprite to move 10 units downward? _____

_____

_____

What do you think you would change if you wanted to make the cat sprite move diagonally? _____

_____

Let's assume a sprite begins at the point (0,10). If you moved it ten units to the left and 10 units down, where would it end up? Explain your answer. _____

_____

_____

_____

Assume a sprite begins at (-7, 2). If you moved it 3 units up and then 8 units down, where would it end up? Explain your answer. _____

_____

_____

_____

Assume a sprite finished moving at (99, -20). If the path it took to that location started at (15, 72), describe a possible path it could have taken. _____

_____

_____

_____

_____

Assume a sprite finished moving at (2, -22). If the path it took to get to that location started at (31, 1), describe a possible path it could have taken. _____

_____

_____

_____

_____

Logic

Another integral mathematics fundamental that is used in programming is logic. Logic has many basic principles, but the one we will use is called "If ____, then ____."

How do you think this principle functions? _____

_____

_____

In programming, an "If _____, then _____" statement (or just "if, then") is a conditional statement where the action after the word "then" will happen when the condition after the word "if" is true. As an example, let's look at the statement below. Imagine that you do not know how this program works, but you are told that the statement below is true.

If the cat is touching the blue bananas, then make the blue bananas disappear.

What do you think would happen if the cat touches the blue bananas? _____

_____

_____

What do you think would happen if the cat does not touch the blue bananas? _____

_____

_____

What do you think would happen if the cat touches the green bananas? _____

_____

_____

What do you think would happen if the cat does not touch the green bananas? _____

_____

_____

What do you think would happen if the cat touches the yellow bananas? _____

_____

_____

What do you think would happen if the cat does not touch the yellow bananas? _____

_____

_____

Let's look at another example by including another statement.

If the cat is touching the blue bananas, then make the blue bananas disappear.

If the cat is touching the green bananas, then make the green bananas disappear.

What do you think would happen if the cat touches the blue bananas? _____

_____

_____

What do you think would happen if the cat does not touch the blue bananas? _____

_____

_____

What do you think would happen if the cat touches the green bananas? _____

_____

_____

What do you think would happen if the cat does not touch the green bananas? _____

_____

_____

What do you think would happen if the cat touches the yellow bananas? _____

_____

_____

What do you think would happen if the cat does not touch the yellow bananas? _____

_____

_____

What do you think would happen if the cat touches the red bananas? _____

_____

_____

What do you think would happen if the cat does not touch the red bananas? _____

_____

_____

Having played Bananas Meow, you know that each bunch of bananas disappears when the cat touches it. Assuming the program for Bananas Meow has four "if, then" statements that make this happen (one for each bunch of bananas), write out all four "if, then" statements using pseudocode (i.e., write each statement as a complete sentence.)

For the red bananas: _____

_____

_____

For the blue bananas: _____

_____

_____

For the green bananas: _____

_____

_____

For the yellow bananas: _____

_____

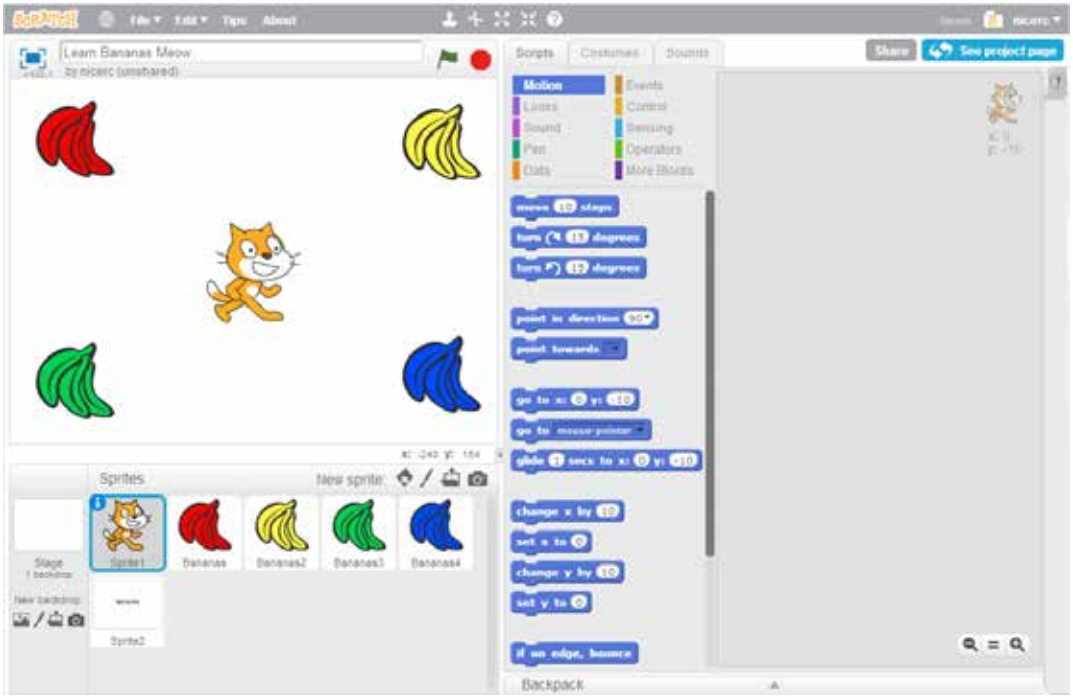
_____

Creating Bananas Meow

Now that you have learned the mathematics behind Bananas Meow, let's create it!

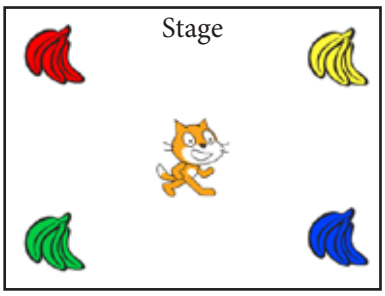
Go to https://scratch.mit.edu/projects/58323144/ and click "See Inside" to open the Learn Bananas Meow project with no code.

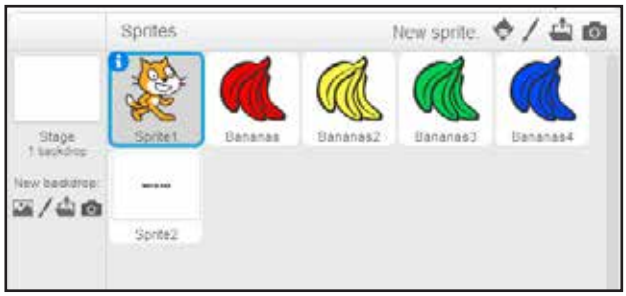
When you open the project, you should see something similar to the image below.



Let's look at the different parts of the screen.

Stage: This is where the game is played. It is also where you can preview your code. Typically, you will click the green flag to start the game and the red octagon to stop the game.





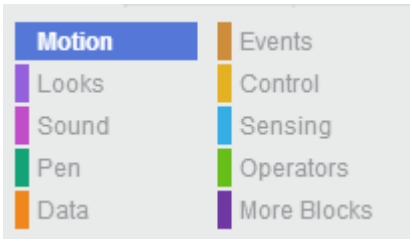
Sprites: The sprites are the programmable objects. These can be characters, scenery, text, or any graphical element you choose. For Bananas Meow, you can see that there are six different sprites and a white backdrop. Describe the six different sprites. _____

_____

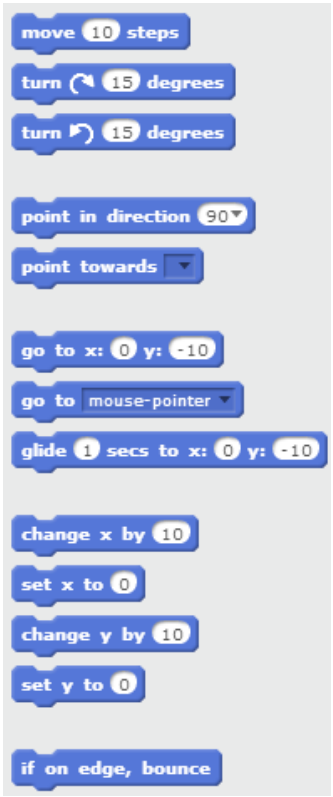
_____

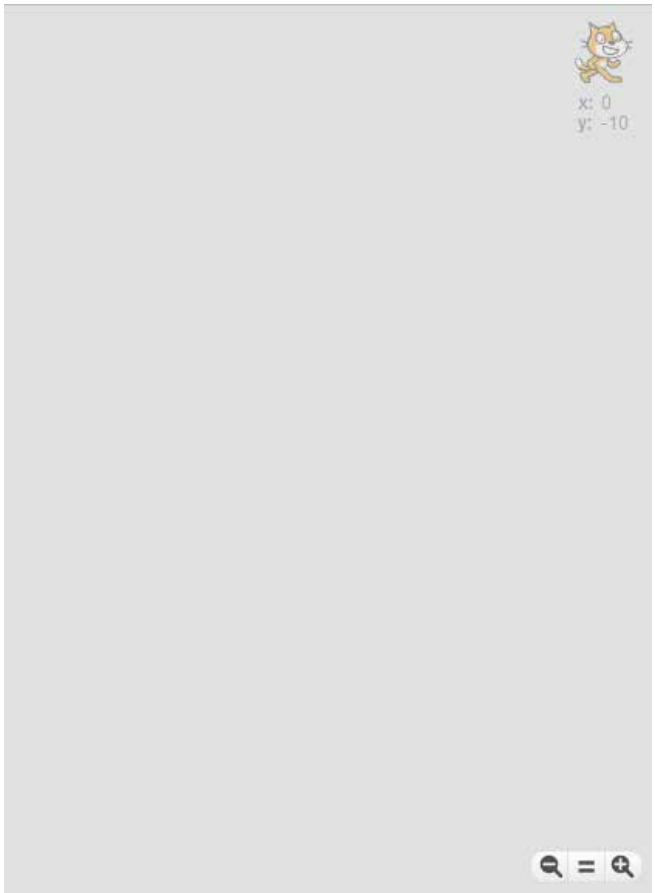
Scripts: This is where you program the sprite you have selected. This section is split into three parts: the block types, the blocks, and the scripts canvas.

- The block types - The block types are Motion, Looks, Sound, Pen, Data, Events, Control, Sensing, Operators, and More Blocks. You will learn more about these types as you create the scripts for each sprite. For now, select the Motion category.



- The blocks - Each category shown above has different blocks associated with it. Shown below on the left are the blocks for the Motion category. In order to program a sprite, drag these blocks to the script canvas. Notice that the blocks are like puzzle pieces and have notches and tabs on them. These notches show you how the different blocks fit together, just like puzzle pieces!





- The script canvas (shown above on the right) - This is where you drag the blocks for a specific sprite to create scripts or script stacks. To delete a block of script, drag it back to the section with the blocks. Also, notice the x and y position of the sprite you are working on is indicated in the top right corner.

Let's program the cat to move when you use the arrow buttons. You will use the programming design process to outline exactly what it is you want the cat to do.

PDP Challenge 1

Challenge: Make Sprite1 (the cat) move when the right arrow key is pressed.

Step 1: Identify the Goal

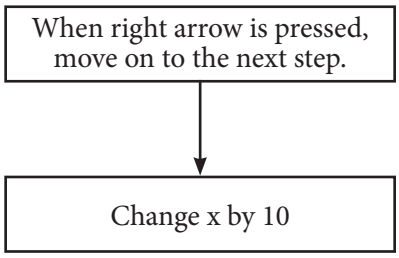
In your own words, identify the goal of the program. _____

_____

_____

Step 2: Design a Solution

The way the script stack will function is outlined below on the left. Use the block category section of Scratch to search for the blocks you think will help you accomplish the given stack flow. Sketch the blocks that you think should be used to the right of each step. Use crayons or colored pencils to color the blocks appropriately.



While exploring the different blocks, you probably found a block under Events that is labeled "when *space* key is pressed" with the word *space* as part of a drop down menu. If you experiment with this block, you will notice that by clicking on arrow next to the word *space,* you can replace it with the words *right arrow* to make it the block you want to use in the next step.

You also might have seen the block labeled "change x by 10" under the Motion section. Notice that the number 10 is surrounded by a white oval. This means that you can change that number to a different value. However, for this program you want to keep the number as 10.

Step 3: Implement the Solution

Now that you have identified the blocks to use, let's implement them by dragging them to the script canvas on the right side of the screen. Make sure you have the cat sprite selected since that is the sprite you are programming. Also, do not forget to change the word *space* to *right arrow* since it is the right arrow key that you want the program to respond to.

After you place your code blocks into position, compare your answer to the picture at the top of the following page.

Step 4: Run and Evaluate the Program

To run and evaluate this stack, press the right arrow key. Did the cat move toward the right side of the screen? If so, how many units did it move? Which axis did it move along? Did the y value change? Evaluate this stack by recording your observations and answering these questions in the space below.

If the program did not complete the goal, return to Step 2 and then Step 3 to ensure you built the stack properly. Then re-run the program and evaluate it. Once the program performs as expected, move to Step 5.

Step 5: Customize the Program

Now that Sprite1 (the cat) moves to the right, you probably want it to move to the left, up, and down. Customize the program by adding three additional stacks of code that accomplish the following goals.

- When the left arrow key is pressed, have the cat move to the left. (Hint: If "change x by *positive* 10" causes the cat to move to the *right,* what value do you think would make the cat move to the *left*?)
- When the up arrow key is pressed, have the cat move up. (Hint: If the "change x by ___ " block moves the cat horizontally, what block do you think will move the cat vertically?)
- When the down arrow key is pressed, have the cat move down.

Now that you can control the cat, let's add one more piece of functionality to the cat. When Bananas Meow started, the cat was hidden for one second, then it appeared at the location (0,0). Let's use the PDP to add this functionality.

PDP Challenge 2

Challenge: When the green flag is clicked, hide Sprite1, have Sprite1 go to (0,0), wait one second, show Sprite1.

Step 1: Identify the Goal

In your own words, identify the goal of this program. _____

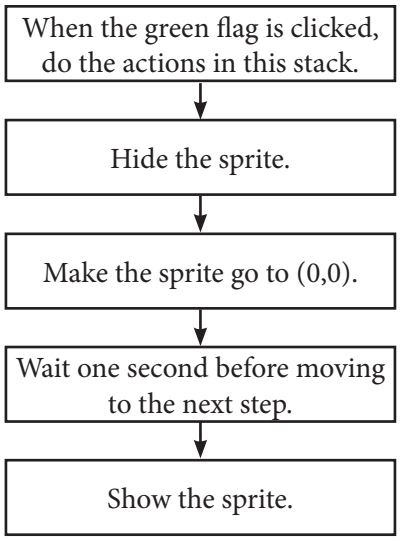
_____

_____

_____

Step 2: Design a Solution

Using the pseudocode written below, search for and then sketch the blocks that correspond to each step. Sketch the blocks to the right of the pseudocode.



While exploring the blocks, you probably found the "when green flag clicked" block under Events, the "hide" and "show" blocks under Looks, the "go to x: 0 y: 0" block under Motion, and the "wait 1 secs" block in the Control category.

If you explored these blocks, you probably found you can change the x and y positions in the "go to x: _____ y:_____" block and you can change the number of seconds in the "wait __ secs" block. However, for Bananas Meow, you will want these values to be 0, 0 and 1 respectively.

Step 3: Implement the Solution

Now that you have identified the blocks to use, place them on the script canvas and connect them in the proper order so that you accomplish your goal.

Step 4: Run and Evaluate the Program

To run and evaluate the program, click the green flag at the top of the Stage. Did the program accomplish the goal? If you cannot tell if it moved from it's original location, use the arrow keys to move the sprite around the Stage, and then click the green flag. Describe how the program functions and write whether or not it accomplished the goal in the space provided. If your program did not accomplish the goal, return to Steps 2 and 3 to ensure that you used the correct blocks and connected them properly.

Step 5: Customize the Program

For now, you will not make any customizations. Once you have completed creating Bananas Meow, you will be given an opportunity to customize the game as a whole. However, you may use the remaining space on this page to brainstorm ideas of how you might want to customize the cat sprite.

Let's add functionality to the bananas. Recall that when you first played Bananas Meow, the cat would collect the bananas when the cat touched them. Visually, the bananas would disappear, or hide, if they sensed that Sprite1 (the cat) was touching them. Use PDP Challenges 3 and 4 as a guide to program the bananas in Bananas Meow.

PDP Challenge 3

Challenge: When the green flag is clicked, make the bananas hide. Then after one second passes, have the bananas appear.

Step 1: Identify the Goal

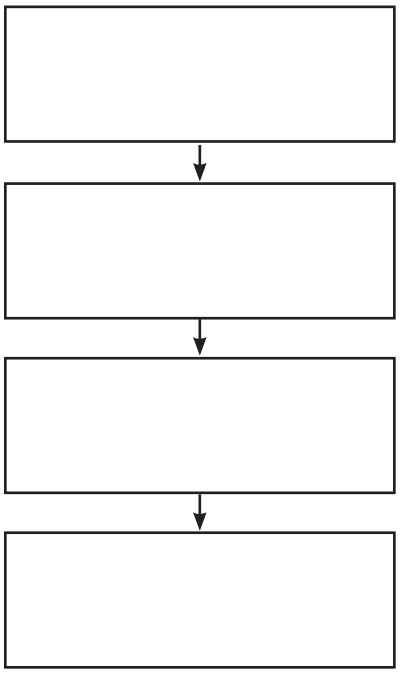
In your own words, write the goal of your program. _____

_____

_____

Step 2: Design a Solution

Since each bunch of bananas will behave exactly the same, let's focus our attention on the red bananas. Once the red bananas accomplish the goal, you can copy the code from that sprite to the blue, green, and yellow banana sprites. Fill the flow chart below with pseudocode that represents each block needed to accomplish the goal of the program. Sketch the blocks as they look in Scratch to the right of the corresponding section of the flow chart. (Hint: You have previously used all blocks that will be needed.)



Step 3: Implement a Solution

Now that you have a solution for the red bananas, drag and drop the blocks onto the script canvas. Be sure to select the red bananas sprite in the Sprites section before placing blocks in the script canvas. See image below.

Step 4: Run and Evaluate the Program

Click the green flag button to test the stack. Record your observations below and state whether or not the code accomplished the goal. If it did not accomplish the goal, return to Step 2 and Step 3 to ensure your program is correct.

_____

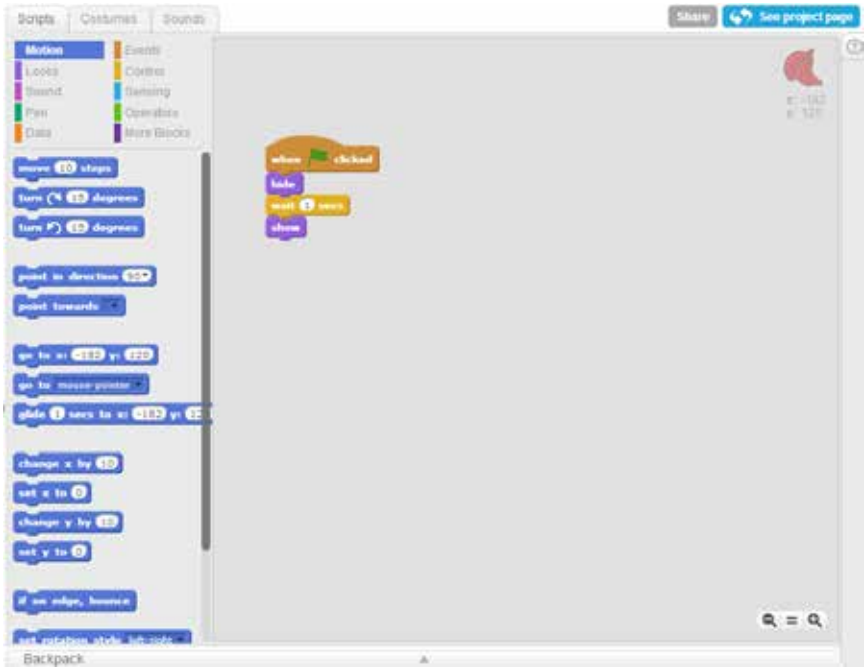
_____

_____

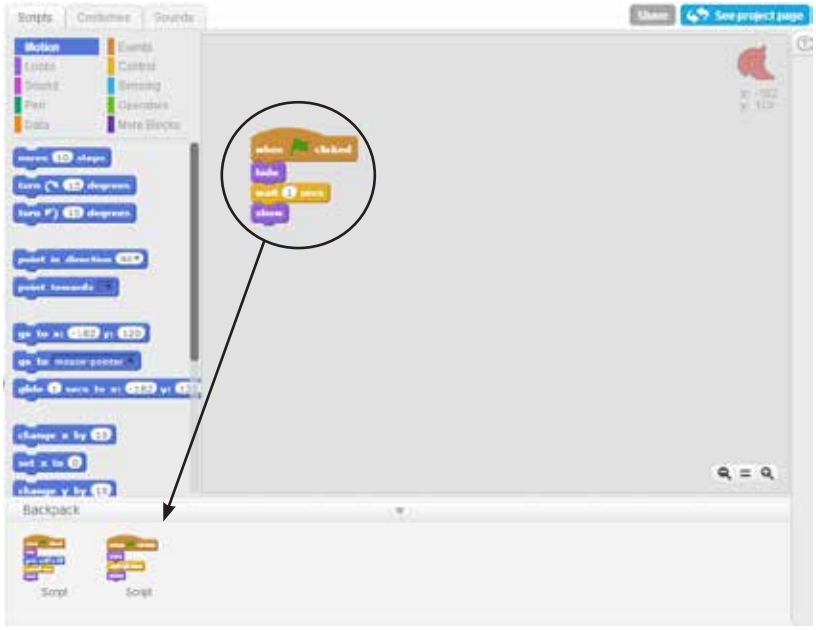
_____

_____

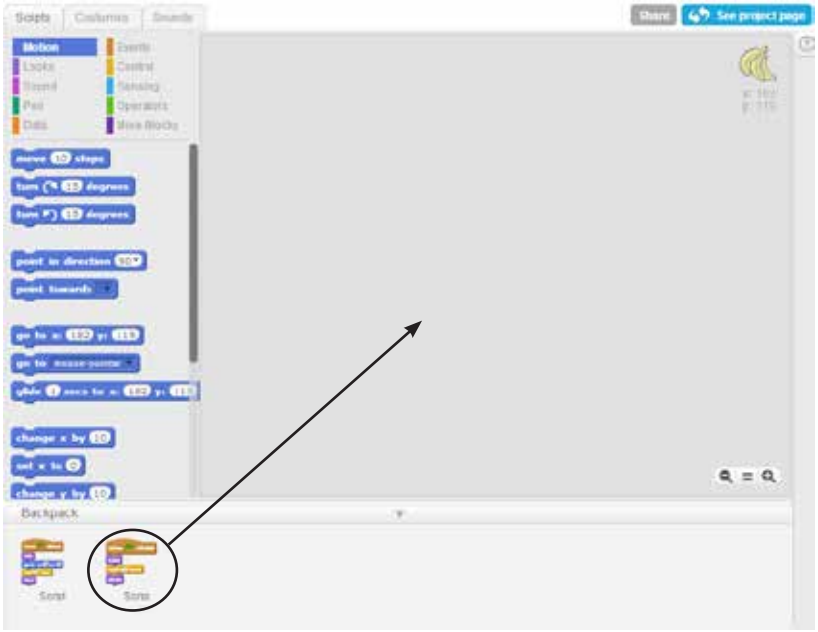
If the red bananas accomplished the goal, you should apply the code to the yellow, green, and blue bananas. A quick way to do that is use the "Backpack" feature. To make the backpack appear, click the arrow at the bottom of the page.



After clicking the arrow, the backpack should open. Click and drag the stack that you want to copy into the backpack.



Then, select the new sprite. For example, choose the yellow bananas. Drag the stack of code from the backpack onto the sprite canvas.



Do this for the yellow, blue, and green bananas. Once the code has been applied to each of the banana sprites, test the program once more. When the green flag is clicked, every sprite should disappear for one second and then reappear.

Step 5: Customize the Program

For now, you will only brainstorm ways to customize the banana sprites. When you finish creating Bananas Meow, you will be given an opportunity to customize the game with these ideas. Write your ideas for customizing the banana sprites below.

_____

_____

_____

_____

_____

PDP Challenge 4

Challenge: Make the bananas repeatedly check if they are touching Sprite1 (the cat). If they are touching the cat, then make the bananas disappear or hide.

Step 1: Identify the Goal

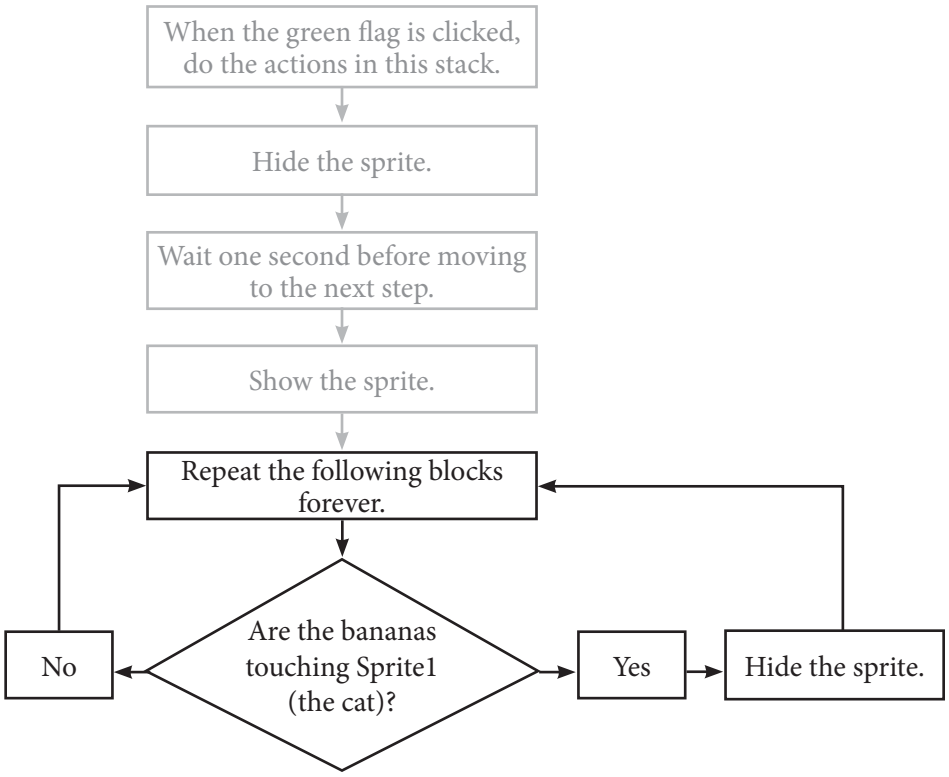
In your own words, write the goal of the program. _____

_____

_____

Step 2: Design a Solution

For this challenge, you will add to the functionality of the previous challenge. Let's begin with the pseudocode. The flow chart below shows the pseudocode from the last challenge in gray and this challenge in black.



Using the flowchart from the previous page, describe how you think the bananas will function. _____

_____

_____

_____

_____

While you were analyzing the flow chart, you probably noticed there is a loop in the program. The first step of the code being added states that the program will need to repeatedly ask if the cat and the bananas are touching each other. In order to ask this over and over (without ever stopping), the program will use a forever loop.

A forever loop repeats the blocks inside of it forever. Two other loops are the "repeat __" loop and the "repeat until" loop.

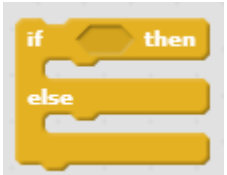
The "repeat ___" block above on the left would repeat 10 times. The "repeat until" block, repeats the code inside of it until a specific event happens. Although any of these could be used in the program, the forever loop will work and eliminate having to input any values.

Once the forever loop begins, the program needs to check to see if something is true. Avoiding specifics, if that something is true, then the program will need to perform an action. Look through the blocks to see if you can find a block that can accomplish this. What blocks do you think can perform this "if something, then something" structure? _____

_____

_____

Under the Control category, you probably noticed the "if, then" and the "if, then, else" blocks.



The "if, then" block performs a task if a certain condition is true. The "if, then, else" block performs one task if a certain condition is true and performs another task if that condition is false. While either can be used for Bananas Meow, the "if, then" block is more appropriate since the pseudocode does not say to do anything additional if the cat and the bananas are not touching.

How do you think the forever loop and the if, then loop will be placed in relation to each other? _____

_____

Since you want the bananas to sense if they are touching the cat, what block do you think should be used for the condition in the "if, then" block? Explore the different blocks to find one you think is appropriate. _____

_____

Under the Sensing category, there is a block titled "touching ___?" You can use this block to select Sprite1 (the cat) as the object the bananas are trying to determine if they are touching. How do you think the "touching ___?" block will be placed in relation to the "if, then" statement? _____

_____

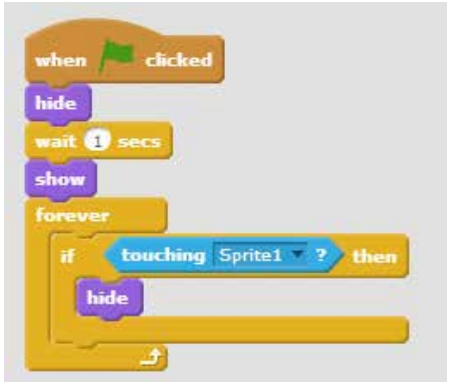
The "touching ___?" block fits into the hexagon shape of the "if, then" statement as shown below.

Lastly, what do you think should go inside the "if, then" block? _____

_____

Step 3: Implement the Solution

Now that you have all of the pieces you will need, input the code into Scratch. For assistance, reference the image below.



Step 4: Run and Evaluate the Program

Run the program by clicking the green flag. Use the arrow keys to move the cat around to see if the bananas disappear when the cat touches them. Did the program achieve the goal? Describe how the program operated. If it did not accomplish the goal, return to Steps 2 and 3 to find and correct any errors. _____

_____

_____

_____

_____

_____

Step 5: Customize the Program

You have already brainstormed ways to customize the banana sprites. Do some more brainstorming now that you know additional information about how to program in Scratch. See if you can think of any new ideas.

After PDP Challenge 5, you will be given an opportunity to customize the game with these ideas. Write your ideas for customizing the banana sprites below.

_____

_____

_____

_____

_____

PDP Challenge 5

Challenge: Add the title page to Bananas Meow. The title page should show for one second immediately after the green flag is clicked and then disappear.

Step 1: Identify the Goal

In your own words, write the goal of the program. _____

_____

_____

_____

_____

Step 2: Design a Solution

For this challenge, you will be using Sprite2. If you select it in the Sprites area, you might not be able to see that it is just a white background with the text "Bananas Meow." It may not appear until you apply the code.

Use the space below to write pseudocode that you believe will accomplish the goal. Be specific with the block names that will be used in each step.

Step 3: Implement the Solution

Use the computer to input your blocks of code.

Step 4: Run and Evaluate the Program

This was the final PDP Challenge for Bananas Meow. The game should now function as it did when you first played it. Does it play the same? Describe how it functions. _____

_____

_____

_____

_____

_____

Is this how the game is supposed to function? If not, return to Steps 2 and 3 to look for errors in your pseudocode and your actual code. If needed, use the original Bananas Meow file to compare your code. The codes should be fairly similar, if not exactly the same.

If it functions properly, then congratulations! You just finished making your first game in Scratch!

Step 5: Customize the Program

Now that you have completely finished making Bananas Meow, it is time to collect the brainstorming ideas you have been jotting down about improving sprite functions. Use the space below to organize the improvements you want to make to the performance of each sprite along with any additions you want to make. Use the questions below as a guide.

- How would you change the bananas?
- How would you change the cat?
- Are there new sprites you want to add?
- Should the game have a screen telling you when you have won or if you have lost?

_____

_____

_____

_____

_____

_____

_____

_____

_____

PDP Challenge 6

Challenge: Take the time to make at least one improvement to the game. Use the remainder of this page to organize your goal using the programming design process.

Step 1: Identify the Goal

Goal Statement: _____

_____

_____

Step 2: Design a Solution

Use the space provided to write your pseudocode.

Step 3: Implement the Solution

Input the code into Scratch with the computer.

Step 4: Run and Evaluate the Program

Run the program and describe how it functions. If the program does not accomplish your goal, revisit Steps 2 and 3 to make the needed changes to the pseudocode and the code on the computer. _____

_____

_____

_____

Step ***5: Customize the Program***

If time allows, make other customizations you put on your list in PDP Challenge 5.

Baseball Pong

Now let's try another game. Go to https://scratch.mit.edu/projects/54967570/ and play Baseball Pong a few times. Describe the game below. Include a description of sprites in the game, the game's functionality, how the game is played, how the speed and score operate, and how speed and score are related to each other.

Since you have had the chance to play Baseball Pong, let's learn about the mathematics fundamentals you will use when you recreate it.

Variables

What do you think a variable is? _____

A variable is a value that you can change. These are usually represented by letters. In mathematics, the most commonly used letter for a variable is *x*. In programming however, variables often take on full words as identifiers.

Looking back at Baseball Pong, what two values do you think are variables? _____

You might have noticed that the score and the speed values change as you are playing. This is because score and speed are variables that have been defined in the program. After variables are defined, a program can change them when certain events happen and compare them to make additional events occur.

How and when do the score and speed variables change when you are playing Baseball Pong? _____

You probably noticed that the score changes every time the baseball touches the paddle and the speed changes as soon as the score becomes an even number. The concept that allows for this is operators.

Operators

You have been using operators and doing operations ever since you learned to count. The most common operations you have done are addition, subtraction, multiplication, and division. The operators in Scratch for these operations are pictured below.

Addition Operator
Subtraction Operator
Multiplication Operator
Division Operator

However, those are not the only operations available. A very popular one in programming is the modulo operation. This operation finds the remainder of one number divided by another. To understand how to it works, let's look at how to calculate values with a modulus.

Modulo Operator

Example: Find 14 mod 10.

Divide 14 by 10 but write it with a remainder.

$$14 \div 10 = 1 \text{ r}4.$$

Since the remainder is 4, the solution is 4. Thus,

$$14 \bmod 10 = 4.$$

Try the next few.

Problem 1: Find 26 mod 10.

Problem 2: Find 2 mod 5.

In this case, the number being operated on (the dividend) is less that the modulus (the divisor). If that is the case, then the solution is simply the number being operated on. Thus, the solution is 2. If you do the math, it will confirm this.

$$2 \div 5 = 0 \text{ r}2$$

$$2 \bmod 5 = 2$$

Problem 3: Find 5 mod 12.

Problem 4: Find 15 mod 12.

Problem 5: Find 12 mod 12.

Any time the dividend is a multiple of the divisor, then the remainder is zero.

$$12 \div 12 = 1 \text{ r}0.$$

Since the remainder is 0, the solution is 0. Thus,

$$12 \bmod 12 = 0.$$

Problem 6: Find 4 mod 2.

Problem 7: Find 13 mod 7.

A benefit of using the modulo operator is to tell when something happens every fifth, eighth, tenth, or __th time! The table below illustrates this with mod 2. Fill in the remainder of the table and describe the result or pattern in the space provided.

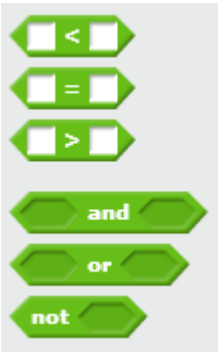
x	x mod 2
1	1 mod 2 = 1
2	2 mod 2 = 0
3	
4	
5	
6	
7	
8	
9	
10	

Describe the pattern: _____

_____

_____

In essence, every "even-th" time, the remainder is zero. If you combine this with an "if, then" block, you can make something happen every time a number is even (i.e., when x mod 2 = 0)! This brings up the next set of operators, the ***logic operators***.



Less Than Operator

Equal To Operator

Greater Than Operator

And Operator

Or Operator

Not Operator

The ***logic operators*** in Scratch (pictured above) are *less than*, *equal to*, *greater than*, *and*, *or*, and *not*. In your own words, describe what you think the benefit of using each of these is in the context of an "if, then" statement.

Less than - _____

_____

_____

Equal to - _____

_____

_____

Greater than - _____

_____

_____

And - _____

_____

_____

Or - _____

_____

_____

Not - _____

_____

_____

Less than, greater than, and *equal to* can all be used to compare two values. These values can be variables or constants. When using "if, then" blocks, you can make something happen if a value is larger, smaller, or equal to another value. Describe how you think the "if, then" blocks below operate.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Let's say that the score is 28. Which of the code blocks on the previous page would be executed? _____

_____

You can combine this concept with mod, the abbreviation for the modular operation, to make other actions happen. Describe how the block below would function.

_____

_____

_____

The *and* and *or* operators can be combined with "if, then" blocks to make things happen based on the truth of multiple values. When using the *and* operator, the items inside an "if, then" block will only execute if BOTH conditions are true. When using the *or* operator, the items inside an "if, then" block will execute if EITHER condition is true. This is shown in the images below.

In the case above, the bark sound will only play if BOTH the score is 0 and the speed is 10.

In the case below, if the score is 0 then the pop sound will play (i.e., it does not really matter what the speed is if the score is 0). Also in this case, if the speed is 10, the pop sound would play (regardless of what the score is). If BOTH the score is 0 and the speed is 10, then the pop sound will still play since one of the two conditions is true.

Given the scores and speed below, fill in the table with bark, pop, or neither based on the two blocks above.

Score	Speed	Sound (Bark, Pop, Bark and Pop, or Neither)
0	1	
3	10	
0	10	
3	7	
10	0	
10	10	
0	0	

The last logical operator mentioned is the *not* operator. This operator negates values. Let's see what is meant by this with the following "if, then" blocks.

In the block above, the bark sound will be played only if the score is NOT 5. Thus, the sound will play when the score is 4, 6, 10, 958798345, or any number that is not 5. In the block below, the pop sound will play when the score is NOT greater than 5. Therefore, it will play when the score is less than or equal to 5.

How do you think the block below functions? _____

_____

_____

Based on the blocks above, fill in the table below with bark, pop, meow, or a combination of the options based on the value given for the score. If none of the sounds will play, write none.

Score	Sound (Bark, Pop, Meow, None)
3	
34	
1500	
-10	
4	
-0.5	
2^3	
5	

The final operator to learn about is the Random Number Generator (RNG). In Scratch, the RNG lets you choose bounds for the random number.

In the image above, the RNG would choose a value between 1 and 10. Where do you think the Baseball Pong game uses an RNG? _____

_____

If you guessed that it uses an RNG to determine where the ball will go after it hits the paddle, then you guessed right. Another place it uses an RNG is to determine the initial direction the ball will go when you start the game.

Creating Baseball Pong

Since you have learned all of the mathematics fundamentals needed to create Baseball Pong, you will begin with Challenge 1 below. Each challenge, as summarized below, will add a small amount of functionality to the game. Once you reach Challenge 8, you will be encouraged to make the game better or fix any aspects you might identify as issues.

- Challenge 1: Make a functioning paddle.
- Challenge 2: Make the ball appear and point upward.
- Challenge 3: Make the ball move and bounce off the edges.
- Challenge 4: Make the ball bounce off of the paddle.
- Challenge 5: Make the game stop if the ball touches the bottom of the Stage.
- Challenge 6: Keep score.
- Challenge 7: Make the difficulty increase as you play.
- Challenge 8: Make your own improvements.

On the following pages, each challenge is broken into smaller steps and outlined using the Programming Design Process (PDP). To begin, go to https://scratch.mit.edu/projects/59222066/ and click "See Inside." This should put you inside the Scratch file with three sprites: Paddle, Baseball, and Sprite1. The script canvas for each sprite should be blank. Begin Challenge 1 on the following page.

PDP Challenge 1

Challenge 1: Make a functioning paddle.

- When the green flag is clicked, have the paddle wait until the mouse touches it to do anything else.
- Broadcast a message, such as "startgame."
- Have the program repeatedly set the *x* position of the paddle to the *x* position of the mouse.

Step 1: Identify the Goal

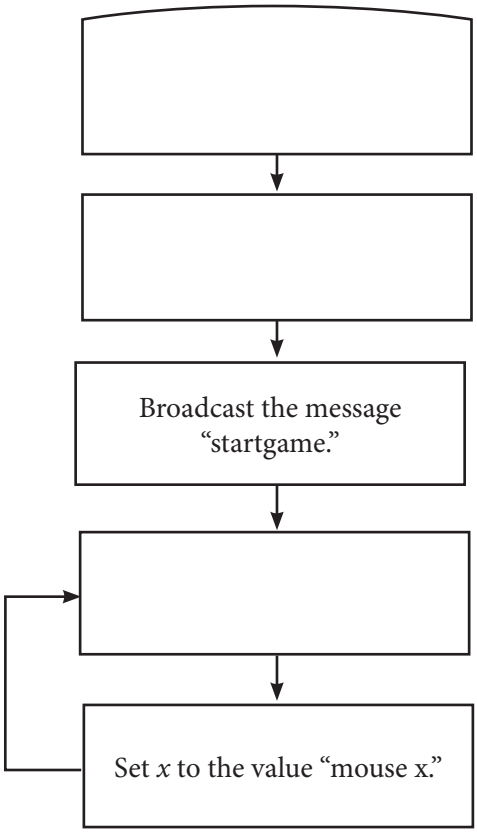
In your own words, identify the goal of the program. _____

_____

_____

Step 2: Design a Solution

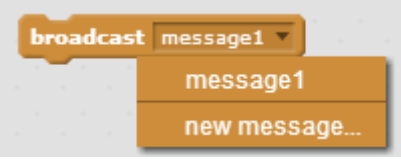
Fill in the flow chart so that each step is covered. To the right of each step, describe or sketch the code blocks that would correspond to that step. Note that some of the steps outlined below may require more than one block.



Before moving to the next step, let's look at the third frame in the flow chart. This frame shows that you will need to tell the sprite to broadcast a message. You probably found the "broadcast message" block under the Events category. If not, click on Events to find it.

This block is used as a way for one sprite to communicate to another sprite. Notice that there is a "when I receive message" block above the "broadcast message" block in the Events section. Later on, you will use the "When I receive message" block to receive the code for the baseball to execute. In summary, the "broadcast message" block is used so the paddle can tell the baseball to start executing code.

To change the message that is being broadcast, click the drop down arrow next to "message1" and select "new message...." You will then be prompted to type a new message name. Do this and type, "startgame" and click ok.



Step 3: Implement the Solution

Now that you have identified the blocks to use, implement them by dragging them to the script canvas on the right side of the screen. Make sure you have the paddle sprite selected since that is the sprite you are programming.

Step 4: Run and Evaluate the Program

To run and evaluate the program, click the green flag at the top of the Stage. Did the program accomplish the goal? Describe how the program functions and write whether or not it accomplished the goal. If your program did not accomplish the goal, return to Steps 2 and 3 to ensure that you used the proper blocks, selected the correct items from drop-down menus, and connected the blocks properly.

Step 5: Customize the Program

For now, you will skip making any customizations. You will be given an opportunity to customize the game as a whole in PDP Challenge 8. Use the remaining space on this page to brainstorm ideas of how you might want to customize the paddle sprite.

PDP Challenge 2

Challenge 2: Make the ball appear and point upward.

- Have the ball receive the message from the paddle.
- Go to the center of the Stage and appear.
- Choose a random upward direction in which the ball will move.

Step 1: Identify the Goal

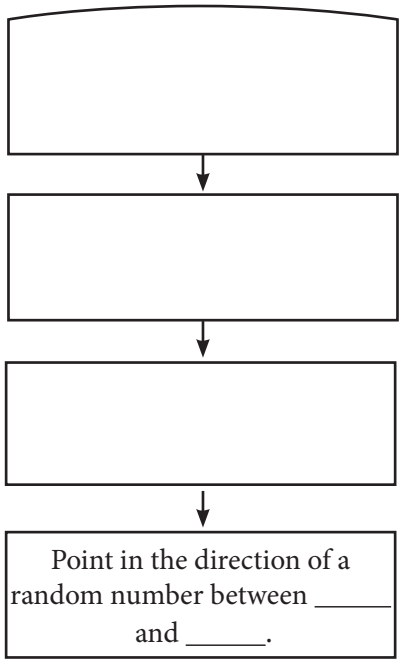
In your own words, identify the goal of the program. _____

_____

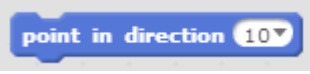
_____

Step 2: Design a Solution

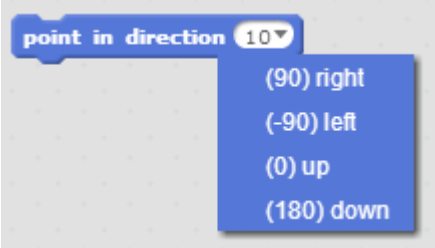
Fill in the flow chart so that each step of your goal is covered. To the right of each step, describe or sketch the code blocks that would correspond to that step. Note that some of the steps outlined below may require more than one block (i.e., a motion block that is filled with an operator block).



Before you implement a solution, let's take a look at the "point in direction" block. This block, along with the "move ____ steps" block prevents you from needing to tell a sprite to go in the *x* direction or the *y* direction when you want a sprite to move. First, the "point in direction" block gives the sprite a direction to travel in, then the "move ___ steps" will move the sprite in the direction it is pointed.



The "point in direction" block has four predetermined options for directions using the drop-down menu (pictured) or you can type in a number from -180 to 180.



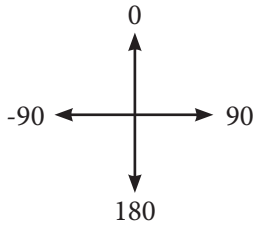
Another way the number can be determined is by placing a value block (the blocks with an oval shape) inside of it. Since you want the value to be a random number, which block will you will want to place inside the "point in direction" block?

_____

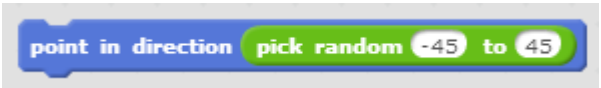
Once you place the "pick random" block inside the "point in direction" block, you will need to determine bounds for the random number that is chosen. To determine these bounds, you need to know how Scratch determines the direction of objects. The images above and the below this paragraph show how this is done. If up is 0, what are two numbers that could be used to ensure the random value produces an upward value?

1. _____

2. _____



The first value you use can be anything between -90 and 0 and the second value can be anything between 0 and 90. The original Baseball Pong used -45 and 45. The resulting block is shown below.



Step 3: Implement the Solution

Now that you have identified the blocks to use, implement them by dragging them to the script canvas on the right side of the screen. Make sure you have the baseball sprite selected since that is the sprite you are programming.

Step 4: Run and Evaluate the Program

To run and evaluate the program, click the green flag at the top of the Stage. Did the program accomplish the goal? Describe what you observed and write whether or not it accomplished the goal. If your program did not accomplish the goal, return to Steps 2 and 3 to ensure that you used the proper blocks.

Step 5: Customize the Program

For now, you will skip making any customizations. You will be given an opportunity to customize the game as a whole in PDP Challenge 8. Use the remaining space on this page to brainstorm ideas of how you might want to customize the baseball sprite.

PDP Challenge 3

Challenge 3: Make the ball move and bounce off edges.

- Repeatedly have the ball move 10 steps.
- If the ball touches an edge, have it bounce.

Step 1: Identify the Goal

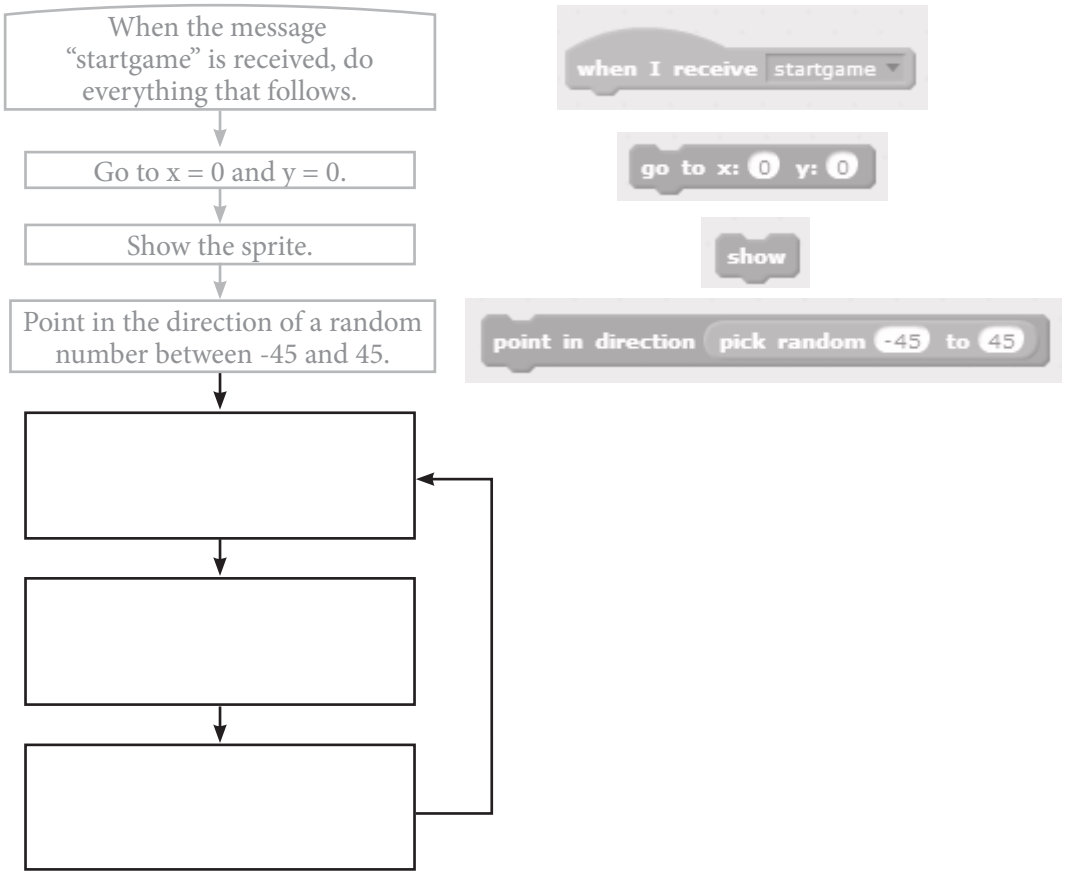
In your own words, identify the goal of the program. _____

_____

_____

Step 2: Design a Solution

Fill in the flow chart so that each step of your goal is covered. Note that you will be adding to the previous PDP challenge. The flow chart below includes the previous challenge's flow chart. To the right of each step, describe or sketch the code blocks that correspond to that step. Note that some of the steps outlined below may require more than one block.



Step 3: Implement the Solution

Now that you have identified the blocks to use, let's implement them by dragging them to the script canvas on the right side of the screen. Make sure you have the baseball sprite selected since that is the sprite you are programming. Also, be sure to attach your solution to the stack of code that is already there.

Step 4: Run and Evaluate the Program

Run and evaluate the program by clicking the green flag at the top of the Stage. Did the program accomplish the goal? Describe what you observed and write whether or not it accomplished the goal. If your program did not accomplish the goal, return to Steps 2 and 3 to ensure that you used the proper blocks.

Step 5: Customize the Program

For now, you will skip making any customizations. You will be given an opportunity to customize the game as a whole in PDP Challenge 8. Use the remaining space on this page to brainstorm ideas of how you might want to customize the baseball sprite.

PDP Challenge 4

Challenge 4: Make the ball bounce off the paddle.

- Repeatedly (but before it moves) have the ball check if it is touching the paddle. If it is touching, then pick another random upward direction.

Step 1: Identify the Goal

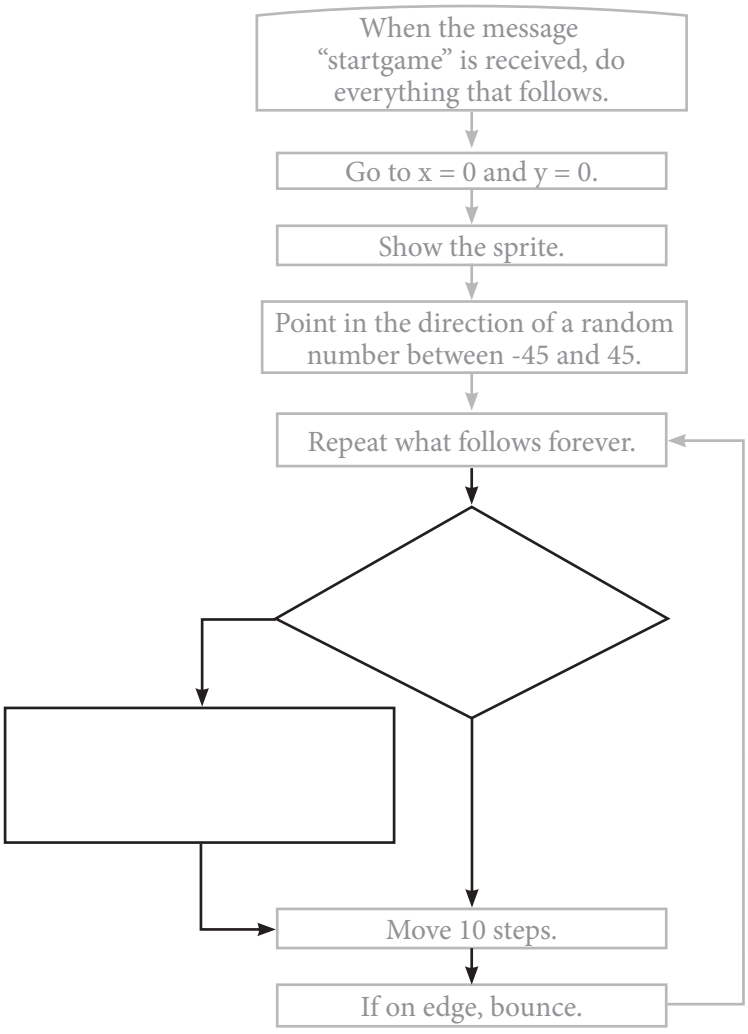
In your own words, identify the goal of the program. _____

_____

_____

Step 2: Design a Solution

Fill in the flow chart so that each step of your goal is covered. Note that you will be adding to the previous PDP challenge. The flow chart below includes the previous challenge's flow chart. To the right of each step, describe or sketch the new code blocks that correspond to that step. Note that some of the steps outlined below may require more than one block (i.e., a motion block that is filled with an operator block).



Step 3: Implement the Solution

Now that you have identified the blocks to use, let's implement them by dragging them to the script canvas.

Step 4: Run and Evaluate the Program

Run and evaluate the program by clicking the green flag at the top of the Stage. Did the program accomplish the goal? Describe what you observed and write whether or not it accomplished the goal. If your program did not accomplish the goal, return to Steps 2 and 3 to ensure that you used the proper blocks.

Step 5: Customize the Program

For now, you will skip making any customizations. You will be given an opportunity to customize the game as a whole in PDP Challenge 8. Use the remaining space on this page to brainstorm ideas of how you might want to customize the baseball sprite.

PDP Challenge 5

Challenge 5: Make the game stop if the ball touches the bottom of the Stage.

- Repeatedly check to see if the ball is touching Sprite1.
- If it is, make the ball hide and stop the entire program.

Step 1: Identify the Goal

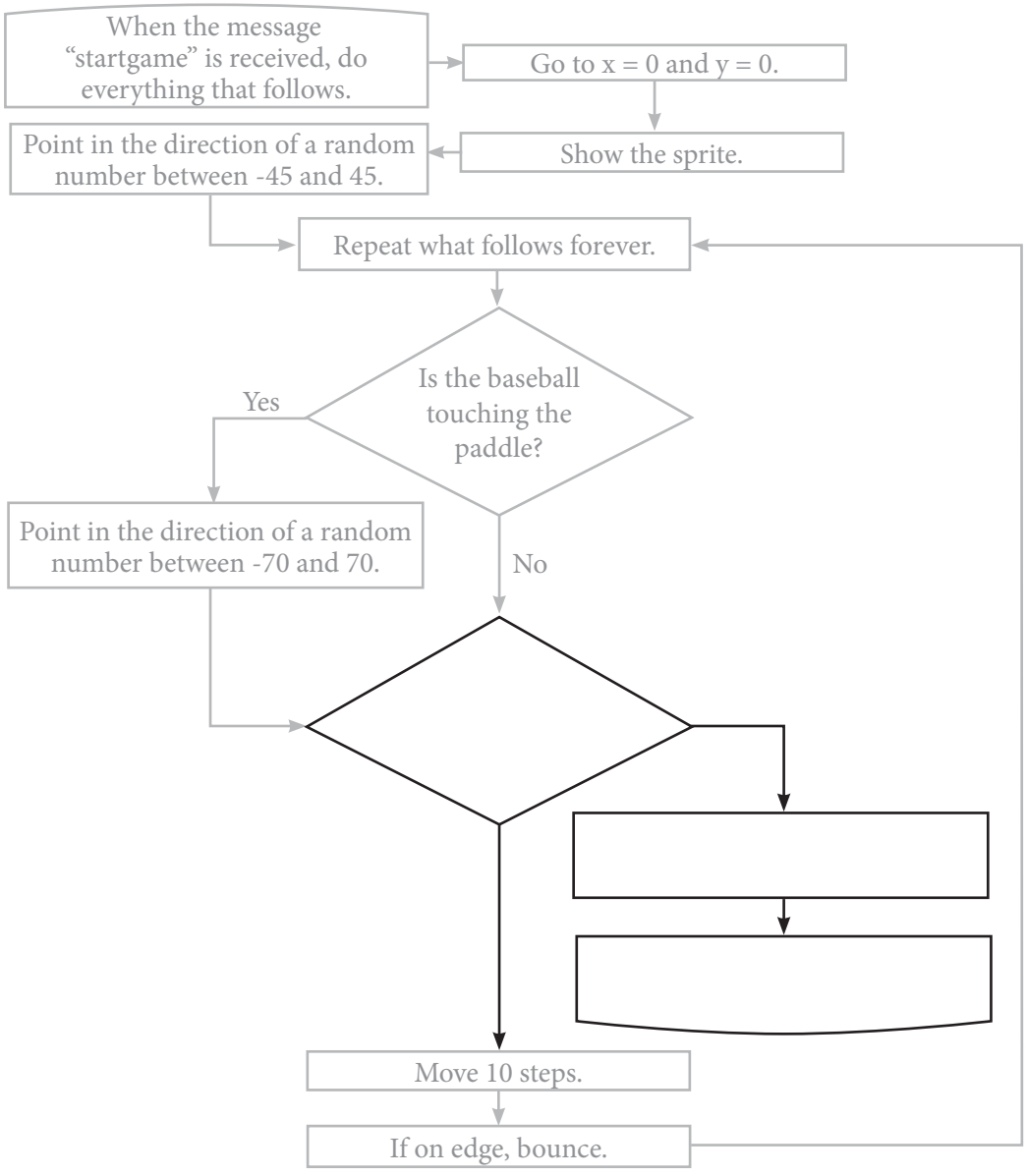
In your own words, identify the goal of the program. _____

_____

_____

Step 2: Design a Solution

Fill in the flow chart so that each step of your goal is covered.



Step 3: Implement the Solution

Now that you have identified the blocks to use, let's implement them by dragging them to the script canvas.

Step 4: Run and Evaluate the Program

Run and evaluate the program by clicking the green flag at the top of the Stage. Did the program accomplish the goal? Describe what you observed and write whether or not it accomplished the goal. If your program did not accomplish the goal, return to Steps 2 and 3 to ensure that you used the proper blocks.

_____

_____

_____

_____

_____

Step 5: Customize the Program

For now, you will skip making any customizations. You will be given an opportunity to customize the game as a whole in PDP Challenge 8. Use the remaining space on this page to brainstorm ideas of how you might want to customize the baseball sprite.

_____

_____

_____

PDP Challenge 6

Challenge 6: Keep score. Have the score increase if the ball touches the paddle.

Step 1: Identify the Goal

In your own words, identify the goal of the program. _____

_____

_____

Step 2: Design a Solution

In order to accomplish this goal, you will need to use a variable. To create a variable, go to the Data scripts and select "Make a Variable."

Scratch will ask you what you want to name the variable and if you want the variable available for all sprites or just this sprite. Since this variable will reflect the score of the game, name it "score" and make it available for all sprites. After creating it, the Data section should resemble the image below.

Since you created a variable and some new options appeared in the Data scripts, which of these do you think will be used to increase the score by one? Where do you think this block should be placed in the stack? _____

_____

_____

One more thing that needs to happen within the program is to insert a block of code that sets the score to zero when the game starts. Which block do you think will do that? Where do you think this block should be place? _____

_____

_____

Although the "set score to 0" block could be on any sprite, you will put it with the baseball sprite. On the following page, add the frames needed that reflect the blocks you will add to Baseball Pong's code. Then, connect the blocks with arrows in the flow chart on the following page. After you have completed the pseudocode, move to Step 3.

Step 3: Implement the Solution

Now that you have identified the blocks to use, let's implement them by dragging them to the script canvas.

Step 4: Run and Evaluate the Program

Run and evaluate the program by clicking the green flag at the top of the Stage. Did the program accomplish the goal? Describe what you observed and write whether or not it accomplished the goal. If your program did not accomplish the goal, return to Steps 2 and 3 to ensure that you used the proper blocks.

_____

_____

_____

_____

_____

Step 5: Customize the Program

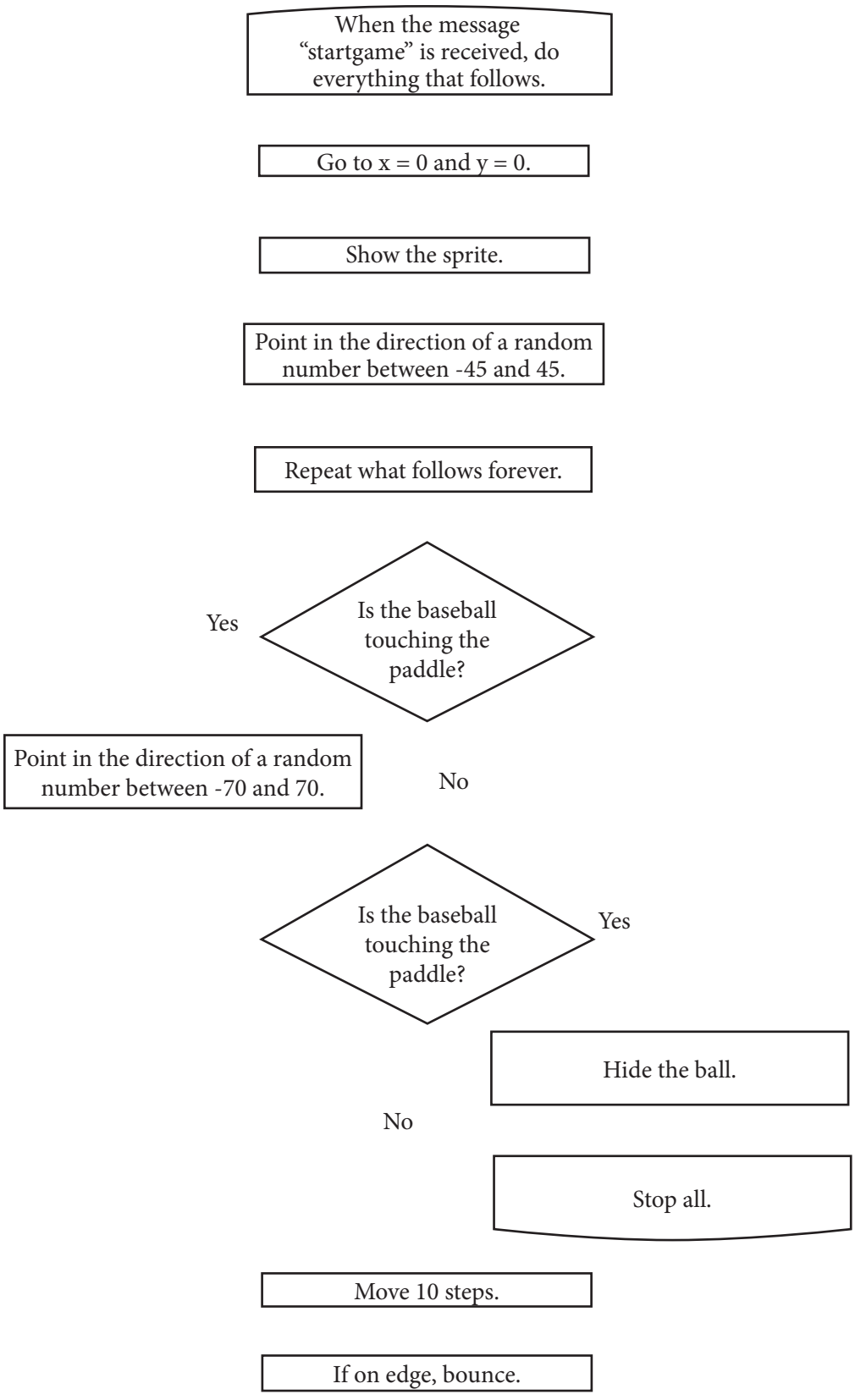
List any customizations or improvements you may want to make in Challenge 8.

_____

_____

_____

_____



PDP Challenge 7

Challenge 7: Make the game more difficult with speed. If the score becomes an even number, then change the speed of the ball by one.

Step 1: Identify the Goal

In your own words, identify the goal of the program. _____

_____

_____

Step 2: Design a Solution

In order to design a solution you will need to create another variable, this time for the speed of the ball. What do you think you should call this variable? _____

Since this variable will need to only change if the score becomes an even number, what other blocks might be involved? In other words, how does Scratch determine if a number is even? _____

_____

_____

Another thing you should consider is the location you want to place the "if, then" block that determines what to do if the score is an even number. Keep in mind there is a block that already makes the ball move at a constant speed. Will this block change? If so, how?

_____

_____

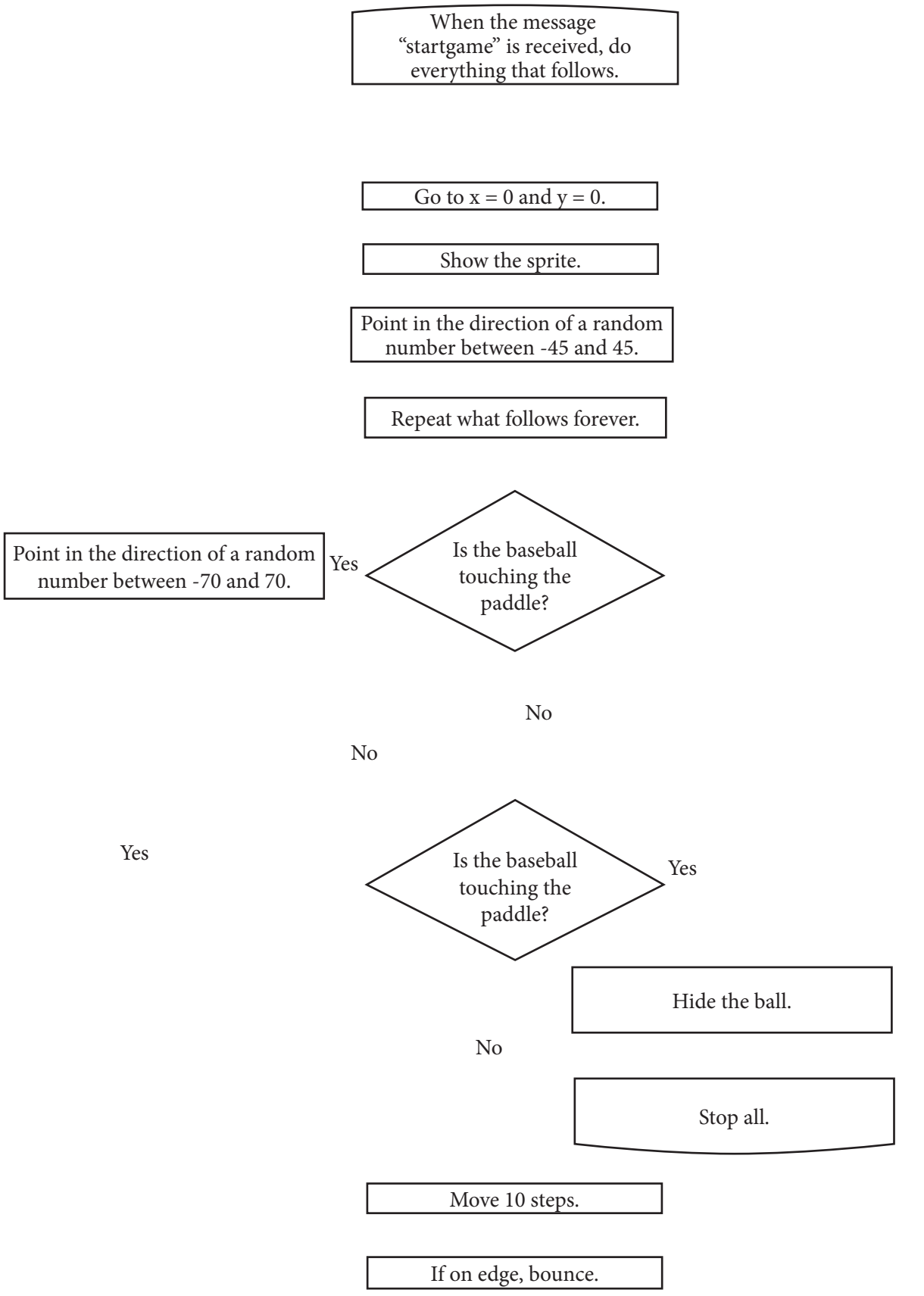
_____

One final consideration is to set the speed to an initial value. What should that value be? Where will this initializing code block be placed? _____

_____

_____

On the following page, add the blocks of code that you believe will accomplish the goal. Then, add the arrows to your flow chart so that it reflects the flow of the program. Note that the steps that were added in the previous programming design process challenge are not shown and will need to be added as well.



Step 3: Implement the Solution

Now that you have identified the blocks to use, let's implement them by dragging them to the script canvas.

Step 4: Run and Evaluate the Program

Run and evaluate the program by clicking the green flag at the top of the Stage. Did the program accomplish the goal? Describe what you observed and write whether or not it accomplished the goal. If your program did not accomplish the goal, return to Steps 2 and 3 to ensure that you used the proper blocks.

Step 5: Customize the Program

List any customizations or improvements you may want to make in Challenge 8.

Take the time to make at least one improvement to Baseball Pong. Use the remainder of this page to consider a goal and complete it with the programming design process. Look back to you notes in Step 5 of PDP Challenges 1 to 7 for ideas.

PDP Challenge 8

Step 1: Identify the Goal

Goal Statement: _____

_____

_____

Step 2: Design a Solution

Use the space provided to write your pseudocode. If more space is needed, use the following page.

Step 3: Implement the Solution

Input the code into Scratch with the computer.

Step 4: Run and Evaluate the Program

Run the program and describe how it functions. If it does not accomplish your goal, revisit Steps 2 and 3 to make the needed changes to the pseudocode and the code on the computer. _____

_____

_____

_____

Step ***5: Customize the Program***

If time allows, make any other customizations you put on your list in the PDP Challenges.

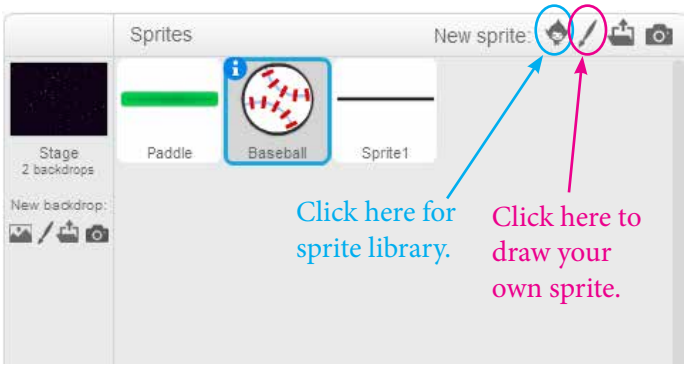
Use this page to help organize your thoughts for PDP Challenge 8.

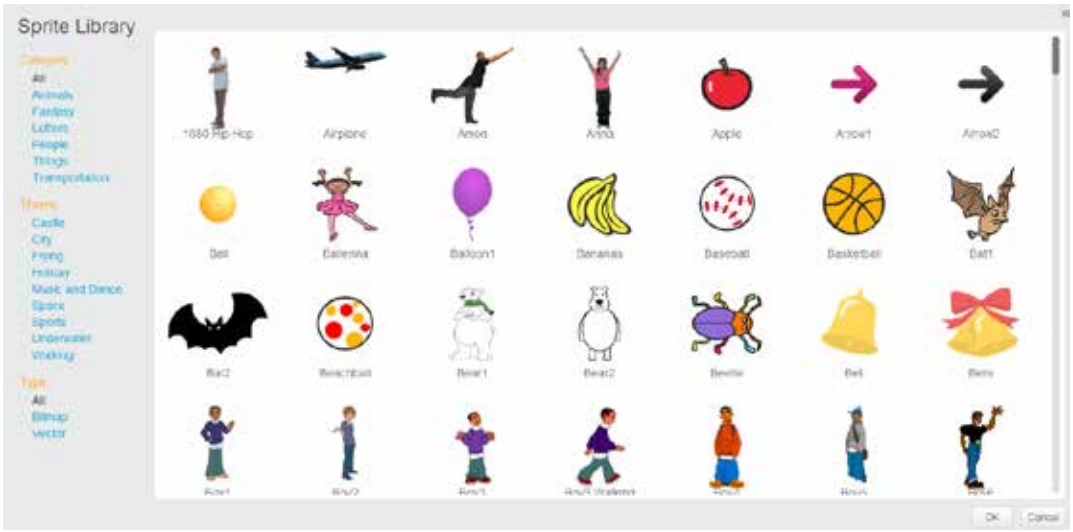
STEP 3: *BRAINSTORMING SOLUTIONS*

You are a game designer creating a new game demo for your company. You have a good idea of the functions of Scratch and the types of games you can create in it, so now it is time to start thinking of different games you might want to create. Use the following pages to come up with three different general descriptions of possible games to create. Things you will want to consider are characters, objects your characters will interact with, how to win, how to lose, how to keep score, how to change the difficulty of the game, and the code and sprites needed to make each of these things happen.

Use the following as a guide to browse the Scratch sprite library to create a new sprite.

The Sprite library has a large selection of sprites for you to base your characters on. To draw your own sprite, select the paint bush icon highlighted in pink below.





Game Idea 1

Name of Game: _____

Characters: _____

_____

_____

_____

Other Sprites: _____

_____

_____

_____

Location/Backdrop/Plot: _____

_____

_____

_____

Game Functionality (How and why do the sprites move or interact with each other?):

_____

_____

_____

_____

How to Win: _____

_____

_____

_____

How to Lose: _____

_____

_____

_____

Sketch of the Game

Draw a sketch of the Scratch Stage that shows the backdrop and the sprites. Include information on how different sprites interact with each other.

Code Blocks

List the code blocks required for the major actions that the sprites will be performing. Use additional paper if needed.

Game Idea 2

Name of Game: _____

Characters: _____

_____

_____

_____

Other Sprites: _____

_____

_____

_____

Location/Backdrop/Plot: _____

_____

_____

_____

Game Functionality (How and why do the sprites move or interact with each other?):

_____

_____

_____

_____

How to Win: _____

_____

_____

_____

How to Lose: _____

_____

_____

_____

Sketch of the Game

Draw a sketch of the Scratch Stage that shows the backdrop and the sprites. Include information on how different sprites interact with each other.

Code Blocks

List the code blocks required for the major actions that the sprites will be performing. Use additional paper if needed.

Game Idea 3

Name of Game: _____

Characters: _____

_____

_____

_____

Other Sprites: _____

_____

_____

_____

Location/Backdrop/Plot: _____

_____

_____

_____

Game Functionality (How and why do the sprites move or interact with each other?):

_____

_____

_____

_____

How to Win: _____

_____

_____

_____

How to Lose: _____

_____

_____

_____

Sketch of the Game

Draw a sketch of the Scratch Stage that shows the backdrop and the sprites. Include information on how different sprites interact with each other.

Code Blocks

List the code blocks required for the major actions that the sprites will be performing. Use additional paper if needed.

STEP 4: *CHOOSING A SOLUTION*

Once you have created three design ideas for your new game, you need to decide which game idea to use. In order to do this, you should rate your design ideas on a set of criteria.

Three criteria for rating your ideas are provided below. Notice, two additional spaces for criteria have been left open for you to determine.

Criteria	Description
Fun Factor	Games need to be fun and engaging to captivate their audience. Which game do you think will be the most engaging for an audience? Some factors to consider are having a relatable main character, an eye-catching look, and bonus features.
Game-play Difficulty	You do not want your game to be too difficult, but you do want it to be a challenge. The right balance will keep the players engaged for as long as possible.
Ease of Construction	Does the game use a lot of complex coding? Is the coding simple? Will you need to learn new blocks that you do not already know?

Design Analysis

Use the three pre-determined criteria along with the two criteria you added to assess your designs. You will evaluate all three game design ideas by ranking the ideas as they relate to the given criteria. For instance, you will look at the category of **Fun Factor** first. You will rank your game ideas on a three point scale where one is the design you feel is the most fun and three is the least fun.

For example, you might rank Game Idea 1 with a two for being the second most fun; Game Idea 2 with a rank of one for having the best fun factor; and Game Idea 3 with a rank of three for not being much fun to play. On the table below, you would write a 2 by Game Idea 1; a 1 by Game Idea 2; and a 3 by Game Idea 3.

You will continue this until all criteria are assessed. Then you will sum the values for each design idea. The Game Idea with the lowest total should be the design you choose for the final design of your game. Since it has the lowest sum, it must have had the most first place rankings. Keep in mind, however, that through this analysis, you may find good things about each game design idea, which might lead you to combine the best components of ideas for a new game design.

Criteria	Game Idea 1	Game Idea 2	Game Idea 3
Fun Factor			
Game-play Difficulty			
Ease of Construction			
Rank Totals:			

Circle the design idea that is the top ranking idea overall (smallest summation value).

Is this the design you thought would rank highest? Why or why not? _____

_____

_____

_____

Will you choose the top ranked idea for your final game design or will you combine ideas to form a new design? Explain. _____

_____

_____

Final Design

Name of Game: _____

Characters: _____

_____

_____

_____

Other Sprites: _____

_____

_____

_____

Location/Backdrop/Plot: _____

_____

_____

_____

Game Functionality (How and why do the sprites move or interact with each other?):

_____

_____

_____

How to Win: _____

_____

_____

_____

How to Lose: _____

_____

_____

_____

Sketch of the Game

Draw a sketch of the Scratch Stage that shows the backdrop and the sprites. Include information on how different sprites interact with each other.

Code Blocks

List the code blocks required for the major actions that the sprites will be performing.

STEP 5: CREATING & DEVELOPING A PROTOTYPE

Now that you have chosen the final design of your game, it is time to begin making it. Use Scratch to create your game. Keep in mind that you might want to change some aspects of the game once you start putting the code together.

Use the programming design process templates included in this section to design and build the code you will run for each sprite. Although three programming design templates are given in this step, you may need to complete the PDP many more times to build your final design.

Programming Design Process

Step 1: Identify the Goal

_____

_____

_____

_____

_____

_____

_____

_____

Step 2: Design a Solution

Use the following page to create the pseudocode that will accomplish the goal identified above.

Pseudocode

Write your pseudocode here.

Step 3: Implement the Solution

Input the code into Scratch with the computer.

Step 4: Run and Evaluate the Program

Run the program and describe how it functions.

Observations: _____

_____

_____

_____

_____

_____

_____

If it did not accomplish your goal, identify changes you think will need to be made. Then, revisit Steps 2 and 3 to make the needed changes to the pseudocode and the code on the computer.

Changes to be made that will help you accomplish this goal: _____

_____

_____

_____

_____

_____

_____

_____

Step 5: Customize the Program

Now that you have accomplished the goal of this PDP iteration, list any new goals that will require you to begin a new PDP iteration. _____

_____

_____

_____

_____

_____

Programming Design Process

Step 1: Identify the Goal

_____

_____

_____

_____

Step 2: Design a Solution

Use the remainder of this page to create the pseudocode that will accomplish the goal identified above.

Step 3: Implement the Solution

Input the code into Scratch with the computer.

Step 4: Run and Evaluate the Program

Run the program and describe how it functions.

Observations: _____

_____

_____

_____

_____

_____

_____

If it did not accomplish your goal, identify changes you think will need to be made. Then, revisit Steps 2 and 3 to make the needed changes to the pseudocode and the code on the computer.

Changes to be made that will help you accomplish this goal: _____

_____

_____

_____

_____

_____

_____

_____

***Step* 5: Customize the Program**

Now that you have accomplished the goal of this PDP iteration, list any new goals that will require you to begin a new PDP iteration. _____

_____

_____

_____

_____

_____

Programming Design Process

Step 1: Identify the Goal

_____

_____

_____

_____

Step 2: Design a Solution

Use the remainder of this page to create the pseudocode that will accomplish the goal identified above.

Step 3: Implement the Solution

Input the code into Scratch with the computer.

Step 4: Run and Evaluate the Program

Run the program and describe how it functions.

Observations: _____

_____

_____

_____

_____

_____

_____

If it did not accomplish your goal, identify changes you think will need to be made. Then, revisit Steps 2 and 3 to make the needed changes to the pseudocode and the code on the computer.

Changes to be made that will help you accomplish this goal: _____

_____

_____

_____

_____

_____

_____

_____

Step 5: Customize the Program

Now that you have accomplished the goal of this PDP iteration, list any new goals that will require you to begin a new PDP iteration. _____

_____

_____

_____

_____

_____

CREATIVE ELEMENT *MOVIE PLOT*

A movie studio found your game demo and has enjoyed playing it so much that they want to create the next big blockbuster based on your video game! In order to do this, you will need to create a movie plot featuring a character inspired by your game. Some aspects you will want to consider are the setting, the protagonist, the supporting characters, the antagonist, the problem or conflict, and the resolution to that conflict.

To help you develop your movie plot, you need to create an outline.

Outlines are useful for many reasons, but the most important one is that it allows you, the author, to know everything about your story and its characters before you begin to write. This is actually similar to the brainstorming process you did for the game itself. This page, and those that follow, provide a structure for your outline.

Title: _____

Characters:

Main Characters (Tell who they are and give at least four facts about them.)	
Protagonist	**Antagonist**

Supporting Characters (Tell who they are and give at least two facts about them.)

Setting (time, location, background details): _____

Problem (why the protagonist takes action/cause): _____

Goal/Solution (what action is taken/effect): _____

_____

_____

_____

_____

_____

_____

_____

_____

Plot (beginning, middle, and end):

Beginning (conflict/start of the problem)

Middle (action/events)

List three to five events that occur that move the story along toward the solution to the problem.

1. _____

2. _____

3. _____

4. _____

5. _____

End (solution/resolution)
This is the climax of the story where it reaches its most intense point, and the conflict is resolved.

Use the space provided to write your movie plot. Refer to the information from your outline as a guide.

STEP 6: *TESTING & EVALUATING THE PROTOTYPE*

Now it is time to test and evaluate your game. Actually you have been testing your game the entire time you were building it. Instead of evaluating the game you made, you will evaluate a game created by some of your classmates.

Partner with another team and play each others games. As you play their game, identify your favorite things about it and think about ways the game could be improved. Try not to look at the code.

Also, choose two sprites and think about how those sprites might function. You will be asked to reverse engineer the code for those sprites. In order to reverse engineer a sprite, you will need to analyze how it functions and try to guess the code that causes it to behave that way.

Exchange games with another team and answer the following questions to evaluate their game.

What are your favorite parts of the game and why? _____

_____

_____

_____

_____

In what ways do you think this game can be improved? How would you make these improvements? _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Identify a sprite to reverse engineer. Write a brief description of the sprite on the line below. Then, use the remaining space to write the pseudocode that describes how you think the sprite functions. Be sure to include specific types of code blocks that you think are causing certain functions to be carried out. Once you are done, click "see inside" to see how close your pseudocode is to the actual code.

Description of the sprite: _____

Pseudocode:

Identify another sprite to reverse engineer. Write a brief description of the sprite on the line below. Then, use the remaining space to write the pseudocode that describes how you think the sprite functions. Be sure to include specific types of code blocks that you think are causing certain functions to be carried out. Once you are done, click "see inside" to see how close your pseudocode is to the actual code.

If there are no more sprites in the game, think of a way another sprite can be added and describe how it would function with pseudocode.

Description of the sprite: _____

Pseudocode:

STEP 7: *IMPROVING & REDESIGNING*

It is time to consider how you could improve your game. Since you evaluated another team's game and they evaluated your game, share your thoughts with each other. Think about how you would make the improvements they suggested. In the space below, describe how the improvements would be made, then write pseudocode to make one of the improvements.

Description of improvements: _____

_____

_____

_____

Sketch of the pseudocode to make one improvement:

Presentations

You will now present your game and movie plot to the class. Use the following information and questions as a guide to form your presentation.

Description of the Game:

- What is the game called?
- What are the sprites?
- How are the sprites controlled?
- What do the sprites do?
- What are the improvements you want to make?

Description of the Movie Plot:

- What is the name of the movie?
- Who is the protagonist?
- Who is the antagonist?
- Give a summary of the movie.

Fill in the guiding questions below for each team as they give their presentation.

Presentation 1:

What is the game called? _____

What are the sprites? _____

_____

_____

How do the sprites function? _____

_____

_____

What is the movie called? _____

Give a brief description of the movie plot. _____

_____

_____

_____

_____

Presentation 2:

What is the game called? _____

What are the sprites? _____

_____

_____

How do the sprites function? _____

_____

_____

What is the movie called? _____

Give a brief description of the movie plot. _____

_____

_____

_____

_____

Presentation 3:

What is the game called? _____

What are the sprites? _____

_____

_____

How do the sprites function? _____

_____

_____

What is the movie called? _____

Give a brief description of the movie plot. _____

_____

_____

_____

_____

Presentation 4:

What is the game called? _____

What are the sprites? _____

_____

_____

How do the sprites function? _____

_____

_____

What is the movie called? _____

Give a brief description of the movie plot. _____

_____

_____

_____

_____

Presentation 5:

What is the game called? _____

What are the sprites? _____

_____

_____

How do the sprites function? _____

_____

_____

What is the movie called? _____

Give a brief description of the movie plot. _____

_____

_____

_____

_____

Presentation 6:

What is the game called? _____

What are the sprites? _____

_____

_____

How do the sprites function? _____

_____

_____

What is the movie called? _____

Give a brief description of the movie plot. _____

_____

_____

_____

_____

Presentation 7:

What is the game called? _____

What are the sprites? _____

_____

_____

How do the sprites function? _____

_____

_____

What is the movie called? _____

Give a brief description of the movie plot. _____

_____

_____

_____

_____

Presentation 8:

What is the game called? _____

What are the sprites? _____

_____

_____

How do the sprites function? _____

_____

_____

What is the movie called? _____

Give a brief description of the movie plot. _____

_____

_____

_____

_____

Presentation 9:

What is the game called? _____

What are the sprites? _____

_____

_____

How do the sprites function? _____

_____

_____

What is the movie called? _____

Give a brief description of the movie plot. _____

_____

_____

_____

_____