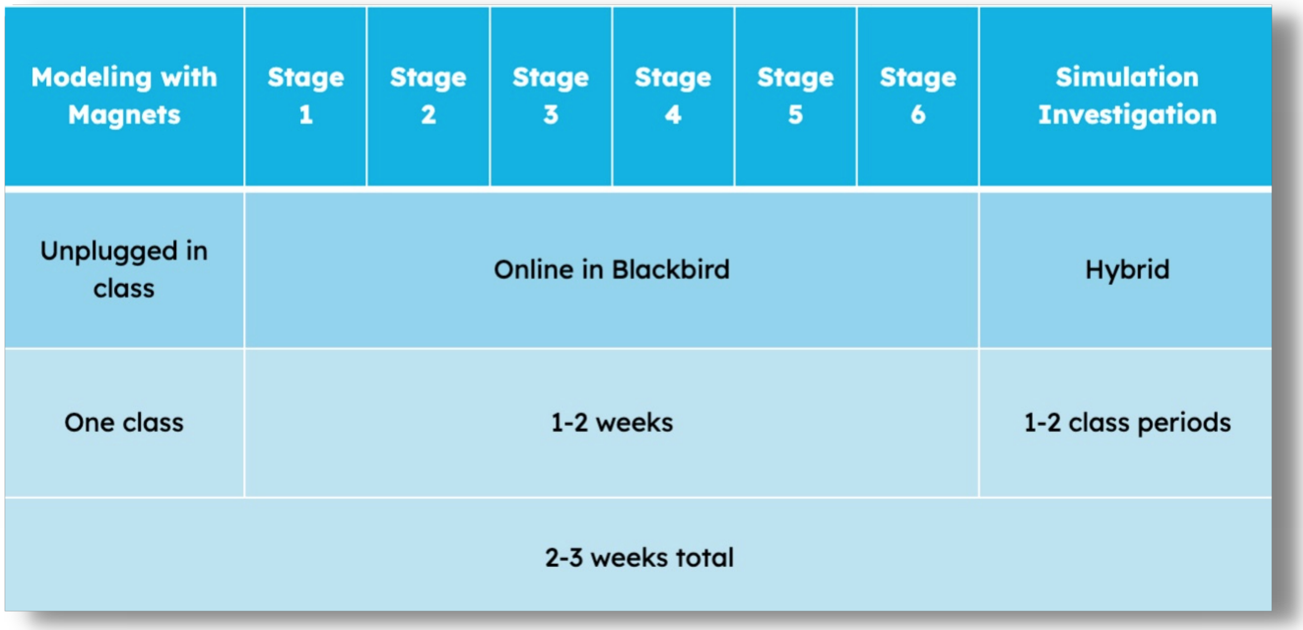
blackbird

MAGNET ROCKET SIMULATION TEACHER GUIDE

PACING GUIDE

'This unit is designed to take 2-3 weeks of class time to complete, with or without Blackbird lessons as homework. The unplugged ‘Modeling with Magnets’ and ‘Simulation Investigation’ activity outlines can be found in the Blackbird Teacher resources inside the app. Each of these unplugged/hybrid classroom activities have more detailed time estimates. The ‘Stage Guides’ section below has more detailed notes about the time to dedicate to each stage as well.

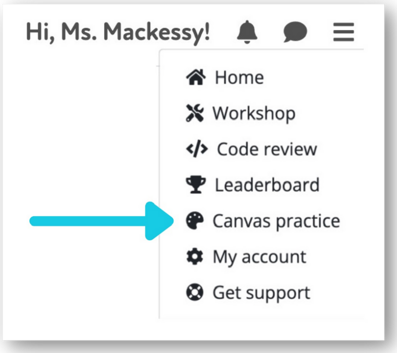


Modeling with Magnets	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Simulation Investigation
Unplugged in class	Online in Blackbird						Hybrid
One class	1-2 weeks						1-2 class periods
2-3 weeks total							

USING STAGE GUIDES

'An overview of major concepts introduced in each stage of lessons in Blackbird. Use the 'Guiding Question' at the start of a stage and 'Probing Questions' once students have finished. Probing questions can facilitate conversations about the simulation students are programming, provide opportunities for questions, important concept review, and connection back to the science of magnetism. Pick your favorite 2 or 3 to use in class.

- ### Stage 1 - Image Objects
  - **Guiding Question:** How can we learn to use Blackbird to start creating our magnetism model?
  - **What students have created at the end of the stage:** A program that places images of the two magnets on the canvas.
  - **Important programming concepts introduced:**
    - Image objects
    - The canvas and coordinates
    - Image library
    - Debugging
  - **Important science concepts:**
    - The like-poles of magnets repel each other
  - **Probing Questions**
    - The magnets in the simulation are facing each other, why is that important?
      - Magnets repel when like-poles are facing each other.
    - The magnet rocket simulation uses canvas.height and canvas.width several times. What do those represent? What happens when you divide one of them in half?
      - They represent the height and width of the canvas, respectively, shown in pixels. They are both evaluated by the program as numbers. When you divide one of them in half, you get the number of half of the pixels in the width or height of the canvas. This is a useful way to find the center of the canvas. You can introduce the Canvas Practice App here to show these properties.



        - What does the Debug button do? Why do you think that might be an important tool as we build this simulation?
            - Steps through a program one line at a time. It is a very useful way to find where errors are occurring in a program.

        - How are the magnet images we're using in the computer model different from the magnets we used to build a physical model?
            - Students will mention the colors and labels. Review what those are intended to represent. They might mention the images are only two-dimensional. Ask why that might matter in the model.

    - **Extension Ideas**
        - Practice lesson 1.3P
        - The Extra challenge at the end of 1.1, "Add to this program so it places 5 Images on the canvas to create a whole zoo of animals!"
        - Workshop project idea: Have students upload some of their favorite images from around the web and use them in a creative way in a program.

    - **Support Ideas**
        - Help students identify the pattern that objects and properties have in the code: [object name][a period][the property name]. For example, rocket.x. 'rocket' is the object name, the period (or dot), and x is the property name. If students are having trouble remembering the names of the properties you can Run the code, and click the {+} symbol next to the name of the object. All of the object property names will be displayed.
        - Remind students that more information lives in the Deep Dive button on each task.
        - Remind students that blue links give helpful examples if they move their mouse over the link.

    - **Notes for Teachers:** At the end of this stage students will have created a simple program that places a couple of magnet images on the canvas. The HTML canvas has its own coordinate system that students won't be familiar with. It's worth taking a moment to show them the canvas using the 'Canvas practice' app. You can place several points on the canvas and ask students to identify the pattern they see in the coordinate pairs. Make sure students know where the 'origin' of the canvas is, and recognize that increasing x and y values of objects positions them to the right and down, respectively. It's not expected that students will completely understand the canvas at this point in the lesson sequence, but it's a good idea to call attention to it at this point. Students will get lots of practice positioning and moving objects in the lessons to come.

- **Stage 2 - Variables**
    - **Guiding Question:** How can we add more information to our magnetism model?

    - **What students have created at the end of the stage:** Images of the two magnets on the canvas and memory storage spaces (variables) for data to be used in calculations

    - **Important programming concepts introduced:**
        - Declaring and assigning variable
        - data types (numbers and 'undefined')
        - Incrementing

    - **Important science concepts:**
        - Forces like gravity, air resistance, and the force with which an object is moved ('power') all affect the movement of objects.
        - Defining variables in a system

    - **Probing Questions**
        - What is the difference between declaring a variable and assigning a variable?
            - Declaring - using **var** to make a new variable.
            - Assigning - using an equals sign (=) to set a value to be remembered by the variable.
            - You can assign a variable a new value as many times as you want. You only need to declare, or create, a variable once.

        - What is an undefined variable? Which variables are currently undefined in the magnet simulation?
            - A variable that isn't remembering any information.
            - **Dist** and **power**

        - What is incrementing?
            - Increasing the value of a numeric variable.

        - Draw attention to the variables named **power**, **gravity**, **airResist**, **speed**. Ask students to consider what data they might remember in the program.
            - No right or wrong answers at this point. These variables become more relevant in later lessons, and are key for creating a realistic simulation.

        - How will adding variables to our program help us create a better model?
            - Variables are memory storage spaces in computer programs. Adding more of them to the program means more information can be stored and used in the model. See if students recognize that if they wanted to create a more accurate model in the future, they'd need to add more variables - more information - to the program. This understanding might not come until they start using the information stored in these variables later in the program.

- **Extension Ideas**
  - Practice lesson 2.3P
  - The Extra challenge at the end of 2.1, "Create a program that declares two variables that hold numbers. Use the two variables in a math equation later in the program."
  - See if students can use a program and variables to solve some of the homework problems they have for math class.

- **Support Ideas**
  - Review the last task in lesson 2.2, a Debug task, with students. When the lesson stops on a line, ask students to identify the 1) declaration keyword (var) 2) the name of the variable 3) the assignment operator (=) and 4) the assigned value. Practice by going through all of the new variables created in the program.
  - Ask students to declare a variable and give it their first name. Have them assign it their height in inches (or cm). Ie. var mike = 76;

- **Notes for Teachers:** At the end of this stage students have added some variables, memory storage spaces, to their program. These are simply holding information currently, but will later be used in calculations that will determine how the magnet rocket will move on the canvas. Now is a good time to use the Debug mode in the finished lesson (2.2) to go over each of the variables and tie them back to science concepts. You can ask students to think of ways they've seen, or experienced, air resistance, for example.

Stage 3 - Animation

- **Guiding Question:** How can we add movement to our model?

- **What students have created at the end of the stage:** Moving images of the two magnets with variables for data to be used in calculations later. The rocket does not move realistically.

- **Important programming concepts introduced:**
  - The animate function
  - repeating instructions (iterating)
  - Decrementing

- **Important science concepts:**
  - Movement is a function of distance over time
  - Velocity

- **Probing Questions**
  - What happens to code **inside** the animate function? **Outside** the function?
    - Inside - It is run over and over 30 times a second.
    - Outside - it is only run once.

    - What does Blackbird mean by ‘one frame of the animate function’?
        - One loop, or iteration, of the code inside the function. The animate function runs at about 30 ‘frames’ per second.
    - What happens to the car at the end of lesson 3.1?
        - Run and pause the program created in 3.1 several times to illustrate that the car continues moving to the right of the canvas. ‘render invisible’ directive just stops Blackbird from causing an error because of it. The car is still there - just not visible.
    - What is decrementing?
        - Decreasing the value of a variable.
    - What does the background function do?
        - Covers up everything on the canvas with a color, white by default.
    - How is the addition of the animate function, or some other type of loop, a useful tool in scientific computer modeling?
        - Loops, like the animate function, allow programs to run the same set of instructions over and over. These are helpful tools in modeling because they allow scientists to see the effect of a phenomena after many years, or many generations, for example. In our model we’re using loops to move a shape on the canvas to create an animation that resembles the movement we saw with the physical magnet launching model.
- **Extension Ideas:**
    - Practice lesson 3.3P
    - The extra challenge from lesson 3.1, “Use what you’ve learned about incrementing to increase the width and height of the car image as it rolls across the canvas.”
    - Ask students to create an animation that moves an image up or down the canvas by a value held in a variable, like they will in stage 4. For example:
        - var speed = 5;
        - car.x = car.x + speed;
- **Support Ideas**
    - Be sure students recognize how the animate function works. Use the Debug mode to show how the code inside the function loops over and over. Have students predict what will happen when each line of code is run.
    - Demonstrate how incrementing works. Place a ‘breakpoint’ on line 23 of lesson 3.1 then press Run. The program will stop on that line and you can move your mouse over the equal sign (=) to see the result of the expression.

- **Notes for Teachers:** The addition of the animate function adds an additional dimension to the computer program. Before this stage the program ran from start to finish in an instant - the image objects and variables were created (declared) and displayed on the canvas, and the program was done. Now the program never stops. The instructions inside the animate function run over and over, 30 iterations per second, until the user stops the program. Be sure that students recognize the part of the program that is making it run continuously. This is also a good place to show students how the animate function works using the Debug button with a finished version of lesson 3.2.

Stage 4 - Ballistics

- **Guiding Question:** How can we make the movement of the magnet in our model more realistic?

- **What students have created at the end of the stage:** A more realistic depiction of two magnets repelling each other

- **Important programming concepts and vocabulary introduced:**
  - Mathematical expressions using variables

- **Important science concepts**
  - Projectile motion experiences friction in the form of air resistance
  - The effect of gravity on projectiles
  - Overcoming forces through an opposing action (vector diagrams)

- **Probing Questions**
  - What are the variables gravity, airResist and power meant to represent in this simulation? What impact do each have on the flight of the magnet rocket?
    - **gravity** - meant to simulate the force of gravity, eventually pulling the object down the canvas.
    - **airResist** - meant to simulate the friction caused by air during flight, slowing the object down
    - **power** - meant to simulate the power of magnets or some other propulsive force (gas engine). Acts against gravity.
  - What does the variable speed control in the rocket simulation? What happens when the value of speed is positive? What about when it's negative?
    - The variable speed is used in the simulation to control how the rocket moves on the canvas. When speed is positive the rocket moves down the canvas (adds pixels to its y-coordinate). When speed is negative the rocket moves up the canvas (removes pixels from its y-coordinate).
  - Why does multiplying **speed** by **airResist** slow the rocket down?
    - The value of **airResist** is **0.97**, so any value multiplied by this will be smaller than the original value.

        - Why does the magnet bounce up and down at the end of Magnet Rocket IV?
            - When the 'rocket' gets close to the launcher, the repulsive force (power) is strong enough to move the rocket up the canvas. As it gets further away from the launcher, the force decreases to the point where gravity is stronger and the rocket falls down. As it falls, the distance between the two magnets gets smaller, and again, pushes the rocket up the canvas.
            - You can think of each of these cycles as a tiny magnet rocket launch!
        - How does our computer model compare to a physical model of magnetic repulsion?
            - This is an opportunity to use an unplugged demo of two magnets, in a tube, repelling one another.

    - **Extension Ideas:**
        - Practice lesson 4.3P
        - The extra challenge at the end of 4.1, "Edit this program so the rocket moves up the canvas *diagonally* so it moves up and to the right."
        - Ask students to create a program in the Workshop that speeds a car along the bottom of the canvas.

    - **Support Ideas:**
        - This stage adds several equations to the program, then uses the result to move an object on the canvas. You might focus on the first step - the equations - by having students create really simple programs that use variables in a math equation. Here is an example, "A program that adds a and b."
          ```
          var a = 2;
          var b = 2;
          var c;
          c = a + b;
          ```

    - **Notes for Teachers:** At the end of this stage students have an animation that looks a lot like the repulsion of two magnets in a system. Take time to ensure students understand why that is - mathematics. It isn't necessarily important for students to understand the inverse square law equation used to calculate power, but it is important for them to be able to recognize it is an important part of making the model more realistic. Math allows us to model all sorts of natural phenomena - velocity, population growth, fluid dynamics, you name it. Ask students about equations they know about that model something in the real world (area, rate, etc).

      If you want to dive deeper into why the magnet rocket moves the way it does, you can use the Debug button to take a close look at the value of speed at different points in the animation. You can use the Pause button to stop the program while the magnet is moving up, then look at the value of speed, then Run the program and Pause while the magnet is moving down and look again. You can let the magnet slow down then Pause and examine the value of speed again.

- ## Stage 5 - Conditional Statements
    - **Guiding Question:** How can we program our model so it can make a decision about what it should do?

    - **What students have created at the end of the stage:** A realistic depiction of two magnets repelling each other that is able to be manipulated by the user.

    - **Important programming concepts and vocabulary introduced:**
        - Conditional statements (if and else)
        - mouse properties

    - **Important science concepts**
        - How distance between magnets impacts the PE and KE of a system.

    - **Probing Questions**
        - Where in your everyday life have you used a conditional statement?
            - Students might mention times when they've been asked or told to do something "or else!". They might describe a time they needed to do something, and if they didn't, they'd have to do a different thing. They might also talk about comparison operators (greater than, less than) from math class.
        - What does a conditional statement do in a computer program? Describe it to your elbow partner in your own words.
            - Conditional statements allow programmers to add logic to computer programs. They check if a condition is met, and if so, execute a set of instructions.
        - Why do we need to use a conditional statement in the magnet rocket model?
            - A conditional statement allows the program to check if the mouse button is clicked and move the position of the rocket, so we can launch it from different heights above the launcher.
            - A second conditional statement stops the rocket from sinking into the bottom of the canvas.
        - Why couldn't the magnet be dragged down to touch the launcher? How was the issue fixed?
            - The program was constantly calculating the new speed of the rocket magnet, even when it was being positioned for a launch. We used an else statement to 'turn off' the electromagnet launcher (stop the calculations that impact speed of the rocket) so we could position the rocket closer.
    - **Extension Ideas:**
        - Practice lesson 5.3P
        - Extra challenge from 5.1, "Add to this program by adding another if statement that checks to see if the 'r' key is pressed down. If it is, change the background color to red."
        - Challenge students to create a conditional statement that displays one image when the result of an equation is an even number, or a different image if the number is odd.

- **Support Ideas:**
  - Conditional statements are essentially instructions given to a computer in the form of ‘if this… then that’. You might help students understand the flow of these statements by creating some simple examples without code but using those terms. For example, “If it is raining outside, then I’ll wear my raincoat.” You can add an else statement to illustrate how those work as well, “If it is raining outside, then I’ll wear my raincoat, otherwise I’ll wear my sweater.” Ask students to come up with their own example. Then you can have them think about how they would write the code of their statement (it doesn’t need to work!):

    ```
    If (raining == true) {
      wear(raincoat);
     } else {
      wear(sweater);
     }
    ```

- **Notes for Teachers:** At the end of this stage students have added logic statements to their program. These statements (if/else statements) are checked each time the program passes over them. If the condition is met (the expression is true) then the code inside the statement runs. If the condition is not met, the program passes over the entire if statement, unless there is an else statement. In that case the code inside the else is run when the condition is not met. Statements like these are the basis of all logic in computer programs. As they grow in complexity, more complex systems can be modeled. Artificial intelligence is an extension of these types of logic statements. It is a good idea to stop and show students the behavior of conditional statements. You can stop on the line with the condition, line 11 in lesson 5.1 for example, and ask students to predict what will happen when the Step button is pressed.

Stage 6 - Displaying Information

- **Guiding Question:** How can we display important information about the magnet launch for the user?

- **What students have created at the end of the stage:** A simulation of two magnets in a system. Information about the movement of the ‘rocket’ magnet is displayed for further investigation about the system as well as to consider limitations of the model.

- **Important programming concepts introduced:**
  - The write function
  - the round function
  - string data type
  - absolute value function

- **Important science concepts**
  - Collecting data allows scientists to develop answers to questions about a system

- **Probing Questions:**
  - How did you calculate the area of the image in the first lesson of this stage? What were the units of that measurement?
    - Students calculate the area of the image by multiplying the width of the image by its height. The units of measurement are pixels.
  - Why were there so many spaces in the strings used in the write function of these lessons?
    - The write function prints out exactly what is included in the strings it is provided - spaces count just like any other character to a computer! The spaces put a little space between the number being displayed and the word added next to it.
  - Why does the program need to use the coordinates of the centers of the magnets to calculate the power?
    - The centers are useful in this program so the power equation on line 61 is never dividing by 0 - which would cause an error. It means the simulation isn't as accurate as it could be, but it saves us from having to add even more conditional statements to the program to prevent errors.
  - How does the round function work?
    - It takes a number (or an expression that evaluates to a number) and rounds it to the nearest whole number. It follows typical rounding rules (above 0.5 rounds up, 0.49 or less it rounds down).
  - Why aren't there any units displayed next to the repulsive force of the magnets?
    - Short answer: the units used to measure magnetic repulsion are complicated!
    - Longer answer: Since the environment we're building our simulation in is two-dimensional, and uses pixels as its most basic unit, the power calculated in this simulation uses arbitrary values for gravity, air resistance, and coils. The resulting power between the magnets follows general rules of magnetism (the inverse square law) but the result of the calculation is only meaningful in the context of this simulation.
  - What does the absolute value function do? Why do we need it in our simulation?
    - It takes a number (or an expression that evaluates to a number) and gives the absolute value (non-negative) of that number. The speed of the rocket flying up is negative, so we need to use the absolute value of the speed to see if the current speed is the greatest speed reached so far.

- **Extension Ideas:**
  - Practice lesson 6.3P
  - Extra challenge from lesson 6.1, "Explore the Docs and find out how to change the Image object to a Rectangle object. Add extra conditional statements to this program so the rectangle changes colors as the area grows. Add a line of text that displays the current color of the rectangle on the canvas."
  - There are other types of conditional statements students can explore in the Docs, like if/else if/else hierarchies and switch statements. Students might like to try adding these to their Workshop programs.

- **Support Ideas:**
  - Stage 6 is all about 'concatenation', or adding text and numbers together to make a sequence of words and numbers. In JavaScript this is done with the + operator. If students experience errors about 'this expression isn't connected properly' it is probably because of a missing + operator in their code.
  - You can practice concatenation 'unplugged' on the whiteboard or on paper. Ask students to describe the weather, or some other phenomena that includes variable data, in a complete sentence.
    'It is rainy today.'

    Then have them break that sentence into the parts they would need to include in the write function in Blackbird so it could be displayed on the canvas:

    'It is ', rainy, 'today.'

    Since rainy weather can change (a variable), that data might be stored in a variable called weather. Then have students jot down the completed line of code:

    ```
    write('It is ' + weather + ' today.');
    ```

- **Notes for Teachers:** This stage concludes with a completed, working version of the model. The goal is for all students to have this program to use in the 'Simulation Investigation' where they collect some data generated by the program, and then think about ways to improve the model so it better represents the relationship between two magnets. The data that can be displayed is only limited by the information given to, or produced by, the program. Encourage students to think about their physical magnet launching model and brainstorm other information that could be added to this model so it gives more information about the system.