Girls Who Code At Home

Algorithmic Artist Game
Unplugged Activity

Activity Overview

We love games! Not only are games a great way of connecting with other people, but they are also a chance to be playful and adaptive in the face of a challenge. In this unplugged activity*, you will create algorithms - or sets of instructions - by telling a player how to recreate a drawing you made. In a program, algorithms are how we communicate to the computer what we want it to do. Once you've played a few rounds, we'll show you how to mod the game (change it up) to make it your own! Before you start playing, we recommend checking out the featured Woman in Tech Spotlight, Danielle Forward. Danielle works as a Product Designer at Facebook and founded Natives Rising, an organization that raises awareness about Native American role models in tech and design through mentorship programs.

Materials

- Another player. You need at least two people, but this game works for a big group as well. This game doesn't require you to be physically next to the people you are playing with, so you can play it with friends or family on your next virtual hangout!
- Algorithmic Artist Game Planning Guide
- Paper
- Pen or pencil and/or markers

**This game is based on Block Talk by the Institute of Play.*

Women in Tech Spotlight: Danielle Forward

Image Source: Medium

First came art, then came technology. As a child, Danielle was fascinated with Japanese comics, animations, and pencil drawings. Using her skills, she eventually decided to pursue Graphic Design in college. She switched to Interaction Design when she realized the growing potential of technology to solve human problems.

As a current Product Designer at Facebook, Danielle works with Internet.org and other social platforms to spread civic and social awareness to millions of users. Danielle is also the founder of Natives Rising, an organization that raises awareness about Native American role models in tech and design through mentorship programs.

You can read more about Danielle's journey from her own perspective in the article, 10 Questions with Danielle Forward. We really like her answer to question #5! Check out this video about Natives Rising to learn more about the organization Danielle founded.

Reflect

Being a computer scientist is more than just being great at coding. Take some time to reflect on how Danielle and her work relates to the strengths that great computer scientists focus on building - bravery, resilience, creativity, and purpose.

 **RESILIENCE**	It took Danielle ten years to graduate with a BFA, and she talks a lot about her perseverance, patience, and planning. What are challenges you have faced in your life? What methods and tools did you use to work through those challenges?

Share your responses with a family member or friend. Encourage others to read more about Danielle to join in the discussion!

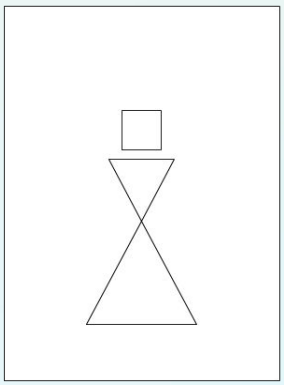
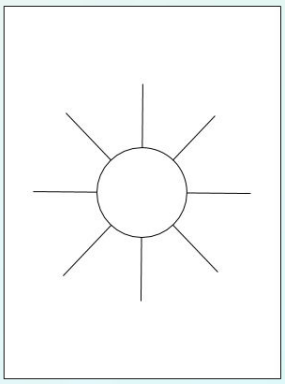
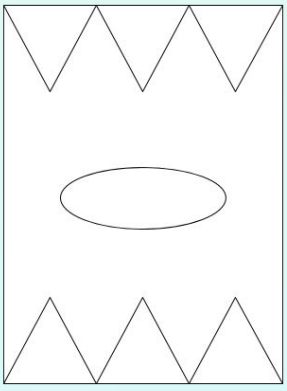
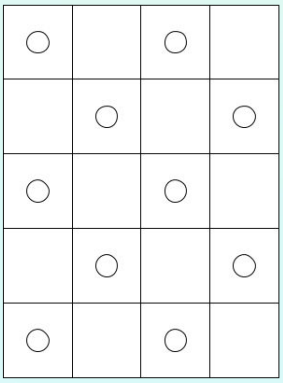
Step 1: Get Started (5 mins)

Gather your materials and players in a spot that is good for drawing - the kitchen counter, living room coffee table, desk, etc. If you're playing with a friend virtually, connect with them over a video chat app and make sure that they have the Algorithmic Artist Planning Guide.

Once you're set up, it's time to start building those drawing algorithms!

Step 2: Create Your Drawings (5-10 mins)

Before you can play, you need to make the drawings that you will use in the game. First, decide which level or levels you want to play: easy, medium, or hard (we recommend a few of each). Next, use the Algorithmic Artist Planning Guide (page 9) to help you generate your drawings. Depending on how long you want to play, each player can make 3-4 drawings.

Example Drawings			
Easy	Easy	Medium	Hard
			

Step 3: Learn the Rules (2 mins)

Decide who will be the first instructor. If you are in the same room, make sure the artist(s) cannot see the drawings. The instructor should have their drawings and the artist(s) should have blank paper and a pen, pencil, or marker.

Here are the rules:

1. The instructor chooses a drawing and tells the artist(s) how to draw it.
2. The artist is not allowed to see the drawing.
3. The instructor may use words that include basic shapes and locations to instruct the artist(s) how to draw the picture.
   Ex: *Draw a circle in the middle of the page, then draw two lines on opposite ends of the circle starting at the edge of the circle. Draw a total of 3 sets of opposite lines from the circle.*
4. The artist(s) are not allowed to speak or ask questions while the instructor is talking.
5. When all the artist(s) are finished, artists will reveal what they drew, then the instructor will reveal the original drawing.

Step 4: Play! (10-15 mins)

You can play a few rounds with one instructor or switch off every round. Decide as a group if you want to play with easy, medium, or hard cards or if you want to play with all levels and choose at random.

After playing a round, reflect together on the drawing(s).

- Did the artist draw it as you expected? Why or why not?
- Which set of instructions were most difficult to interpret?
- How could the instructor improve their instructions?

Step 5: Reflect - What is an Algorithm? (5 mins)

Let's stop for a second and talk about **algorithms***. Algorithms are a generalized and repeatable set of instructions that have a specific purpose and output, given a set of inputs. We often hear algorithms mentioned in the context of digital computing, but they are also important for **analog** (unplugged) processes!

**The following concept descriptions are adapted from the NYC DOE CS4ALL Blueprint.*

Step 5: Reflect - What is an Algorithm? (Continued)

All algorithms have **inputs** and **outputs**. Let's take a baking example: the ingredients are the inputs, the recipe is the algorithm, and the output is the completed baked good. Let's take a recipe for baking a cookie. In baking, you can't just throw all the ingredients together and expect the cookie to come out well - the instructions are ordered in a particular way. First we need to mix the sugar and eggs, then add the flour and baking powder, and finally add the toppings. We especially love to add chocolate chips in our cookies!

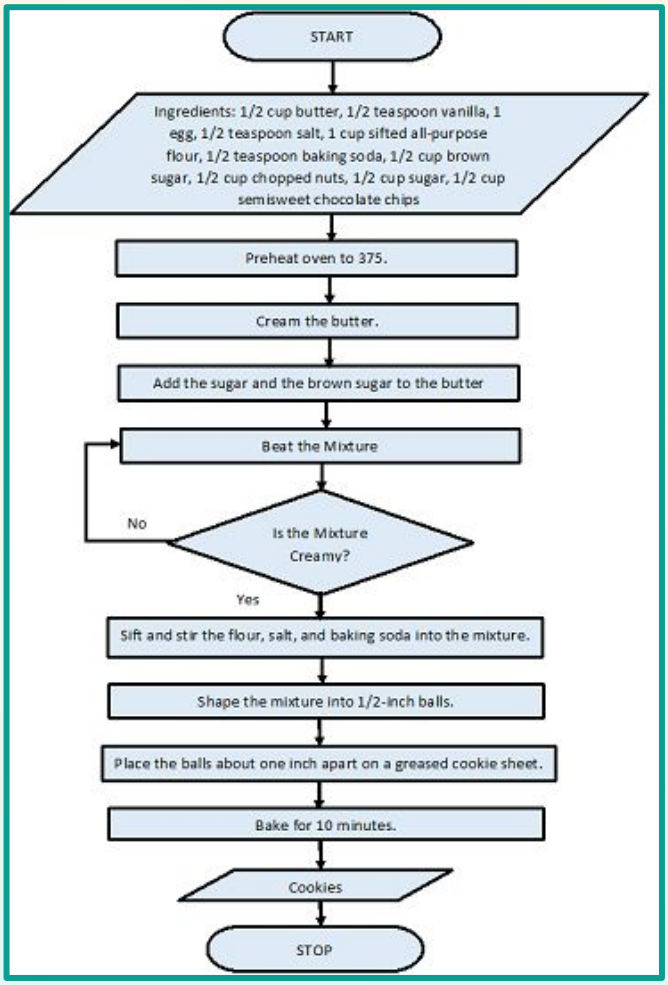


Image Source: Study.com

The order in which the steps of an **algorithm** (or the recipe) are completed is called the **control flow.** Programmers tell computers which order to do each step of an algorithm using logic like IF or conditional statements and loops. If statements require the computer to first check if something is true before running the code. For example, when preparing our cookie we might check if the sugar and egg is mixed, then add the flour.

Loops are used to repeat a step either for a certain number of times, until a condition is met, or sometimes forever. We use a loop in baking to *repeat* mixing until the mixture has no lumps.

When creating an algorithm, it's important that steps in the instructions are clear and concise. Anyone following your steps should be able to recreate the **output** or result of your algorithm without explanation. For example, a good recipe might tell the user that "after the eggs are beaten, add 1/2 cup of sugar to the bowl and mix thoroughly". If the step in the recipe said "add sugar" instead, this could cause the baker to be confused about whether they need to mix it or just add the sugar? This is called **algorithm design**.

Step 5: Reflect - What is an Algorithm? (Continued)

When designing algorithms, it's important to understand how the algorithm will be used and who might be impacted by it. Algorithms often have biases based on who created it, when it was created, why it was created, and how it was made. It might not seem so important in this exercise, but when it comes to things like facial recognition and job recruitment, these algorithmic biases can lead (and have led) to social injustice. Watch this video to learn more about algorithmic bias.

Whether giving instructions to create a drawing, writing a recipe, or building an app that allows you to try on different hairstyles, always remember the users on the other side of it. In the next section, you will change the game to make it your own!

Step 6: Mod Your Game (10 mins)

Once you have played a few rounds, you can try implementing a new constraint: The instructor can no longer use the following words in their description: **above, below, left, right.**

Play a couple of rounds, then reflect on the following questions:

- How did this constraint change the game?
- What strategies did you come up with as the instructor to get around using these words?
- What changed for you as the drawer?

Computer scientists call the addition of a feature to a pre-existing game a **mod** (short for modification). If you've ever designed a game, you probably discovered that it's tough to create a new game from scratch! The best way to learn how games work is to mod a game you know. In fact, you might discover that by changing one part, you will need to change others as well.

Next, work together to come up with another change to the game. If you have a big group, you can break up into smaller teams. Here are a few ideas to get your started:

- Add colors.
- Include a time constraint.
- Implement a points system.

Step 7: Playtest Your Mod (10 mins)

Now that you have your own version of the game, you want to test it! Try playing a couple of rounds with each other to see how your mod feels. Is it fun? Do you need to change something else to make it better? Reflect as a game design team and make any changes to refine it. Don't forget to give it a name!

Ask other friends and family members to playtest your game and see what feedback they have. If you want to make changes based on their feedback, go ahead and do that too!

Step 8: Share Your Girls Who Code at Home Project (5 mins)

We would love to see your algorithmic creations, especially if your added mods to your game! Don't forget to share your algorithmic instructions and drawings on social media.
Tag @girlswhocode #codefromhome

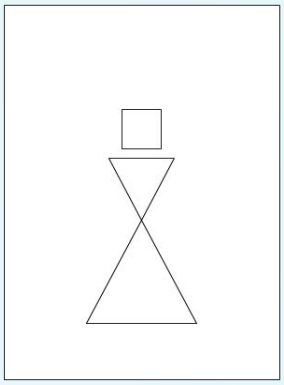
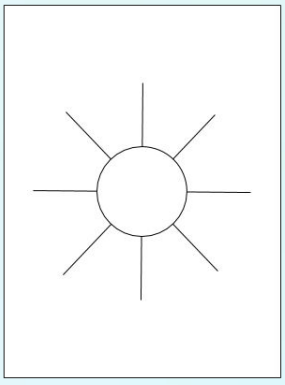
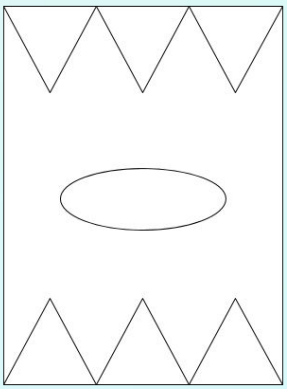
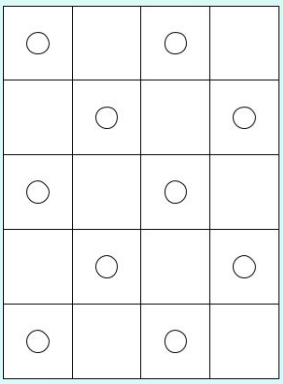
Algorithmic Artist Game Planning Guide

Instructions

Use this guide to create the drawings you will use in your game. We will use four variables to generate the content of the drawing: type of shape, the number of shapes, the size of a shape, and the color of a shape. How you configure them is up to you!

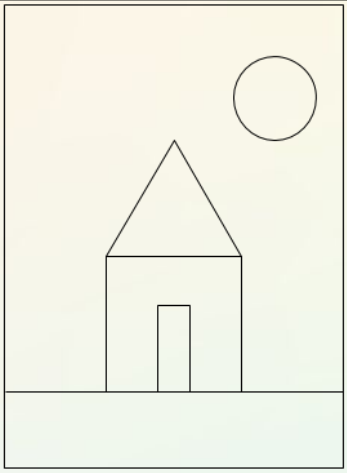
- **Shape.** What shape will you use? For example, circle, square, rectangle, triangle, oval, line, hexagon, etc.
- **Number of shapes.** How many of each shape will you include in your drawing? We recommend 2-4 shapes for an easy drawing, 5-6 shapes for a more challenging drawing, and 7-10 shapes for the hardest drawing.
- **Size of the shape.** How big or small will it be? Tiny, small, medium, big, huge? This will be relative to every player, so during gameplay you should think about other ways you can describe the size of a shape.
- **Color of the shape.** What color is it? Red, green, yellow, purple, teal?

You can use the tables below to help you plan each drawing. We've included two tables for each level with space to draw, but you can always add more to make the game shorter/longer, easier/harder, etc.

Example Drawings			
Easy	Easy	Medium	Hard
			

Example Drawing and Instructions

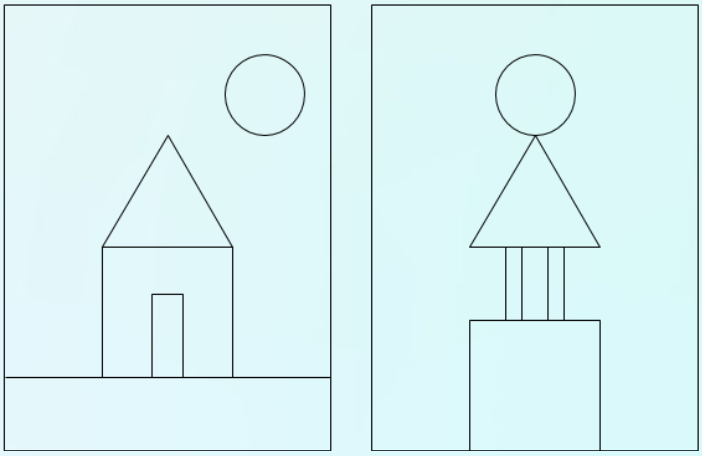
Level: Medium

Shape	Number of shapes	Drawing
Triangle	1	
Circle	1	
Square	1	
Rectangle	2	

Algorithm Instructions

1. Draw a rectangle at the bottom of the page
2. Draw a square on top of the rectangle you drew. The square should be aligned in the middle of the page
3. Draw a rectangle inside the square so that the bottom is touching the rectangle.
4. Draw an equilateral triangle on top of the square. Two of the corners of the triangle should touch two adjacent corners of the square.
5. Draw a circle on the top right of the page.

Based on the shapes we have listed, you could draw either of these images.



Level: Easy (2-4 shapes)

Drawing 1

Shape	Number of shapes	Drawing
Algorithm Instructions		

Level: Easy (2-4 shapes)

Drawing 2

Shape	Number of shapes	Drawing
Algorithm Instructions		

Level: Medium (5-6 shapes)

Drawing 1

Shape	Number of shapes	Drawing

Algorithm Instructions

Level: Medium (5-6 shapes)

Drawing 2

Shape	Number of shapes	Drawing

Algorithm Instructions

Level: Hard (7-10 shapes)

Drawing 1

Shape	Number of shapes	Drawing

Level: Hard (7-10 shapes)

Drawing 1

Algorithm Instructions

Level: Hard (7-10 shapes)

Drawing 2

Shape	Number of shapes	Drawing

Level: Hard (7-10 shapes)

Drawing 2

Algorithm Instructions