Asian MathSci League, Inc (AMSLI)
Partner: National Olympiad in Informatics Philippines (NOI.PH)
Email address: amsliphil@yahoo.com / ask@noi.ph
Contact Nos: +632-9254526 +63906-1186067

Student Copy
AIEP Send-off Scratch 2 Session 10

ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

When we hear "artificial intelligence" or "AI," what ideas, concepts, or things come to mind? Perhaps, we might be imagining robots, self-driving cars, or even Baymax, the personal healthcare companion from *Big Hero 6!*

But AI is not confined to the realm of fiction; facial recognition systems like those used in biometrics, virtual assistants like Amazon Alexa, smart vacuum cleaners like Roomba, and chess engines like Stockfish all leverage AI.

To formalize our discussion, we start by defining some terms. Artificial intelligence refers to the branch of computer science that is interested in the creation of intelligent agents. An intelligent agent is anything that perceives its environment and, based on the things it senses and the experiences it accumulates, acts autonomously to achieve its goal.

But what exactly is this goal? It can be as simple as generating a maze (our focus in the previous session) or as complex as creating a robot that can discover a cure for cancer. Broadly speaking, we can classify the goals of intelligent agents into four categories:

a) Think Humanly. Human thinking is a lot more complex than simply being able to solve puzzles; it encompasses creative tasks, such as writing essays. An example is GPT-2, which can produce text that is near-indistinguishable from human works.

b) Think Rationally. Thinking rationally means always choosing the best decision that will lead the agent to accomplish its goal. This is embodied in OpenAI's AlphaGo, which beat the world champion Lee Sedol in the game of Go in 2016.

c) Act Humanly. Agents that act humanly aim to imitate how we people interact with the world around us. For instance, the robot Sophia can maintain eye contact and even mirror some of people's facial expressions.

d) Act Rationally. Similar to thinking rationally, acting rationally means always doing the best action towards goal completion. An excellent example is EMMA, a system that transforms our ordinary cleaning appliances into self-driving vehicles.

Creating an Intelligent Agent

Traditional AI relies heavily on rules, `if-else` statements, and graph search algorithms, such as depth-first search and breadth-first search. In this regard, if we want to create an agent that can play tic-tac-toe perfectly, we can actually make one in Scratch using a bunch of `if-else` blocks that dictate the best counter to every possible move.

How about chess and other more complicated games? There are just too many positions to consider! Modern AI attempts to solve this dilemma by employing machine learning, a collection of techniques that allow a computer to learn and improve on its own.

Asian MathSci League, Inc (AMSLI)
Partner: National Olympiad in Informatics Philippines (NOI.PH)
Email address: amsliphil@yahoo.com / ask@noi.ph
Contact Nos: +632-9254526 +63906-1186067

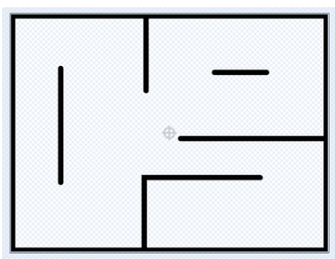
Traditional AI Automatic Maze Solution[1]

To wrap up our discussion of traditional AI, let us complement our maze generation project from our previous session by creating an agent that can automatically solve a given maze. In computer science, we refer to this task as pathfinding.

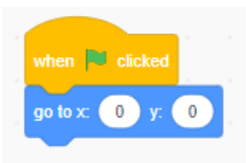
Here is a video of the final output.

A. Setting up the Preliminaries

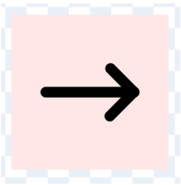
1. Retain the default sprite, but rename it to Cat and set its size to 30.
2. Add the Apple sprite from the library, setting its size to 60.
3. Create a new sprite named Maze, and draw a pattern similar to the one below; the intention of this pattern is to include looping corridors (top right) and dead-ends (bottom right). Set the pen size to 16 (for ample thickness) and the color to black.



4. Select the Maze sprite, and position it at the center of the stage.



5. Create a sprite named Path, and draw a square tile that is 30 pixels long. Recall from the previous handout that one box in the drawing area is 4 pixels long. Hence, the side of this tile should span a little less than 8 boxes. Draw a black arrow at its center to denote the direction that the Cat sprite will take.



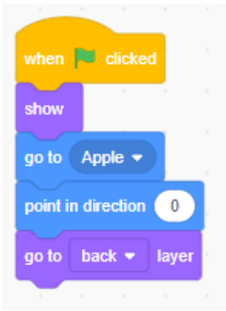
6. Name the first costume Up. It points to the right, but just name it Up. Duplicate this costume to create three more costumes corresponding to the remaining cardinal directions: Right, Down, and Left. All of them point to the right, but do not worry. We will delegate the task of rotating them to our code.

   Color each costume differently; this will be significant later on.

[1] Adapted from https://www.youtube.com/watch?v=22Dpi5e9uz8&t=886s

B. Solving the Maze Backwards

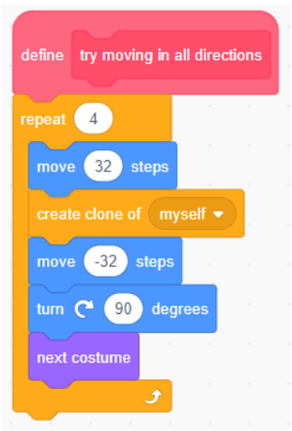
1. Intuitively, it is easier to solve a maze by starting at the goal (the apple). We begin by positioning the arrow tile (Path sprite) behind the apple (`go to back layer`).

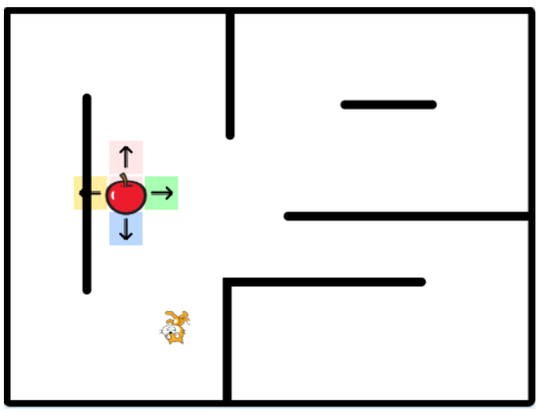


2. Create a My Block for the **Path sprite**, and name it `try moving in all directions`. Make sure that it runs without screen refresh:

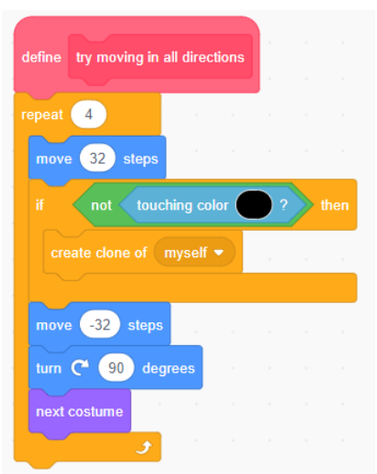
   - Clone the arrow tile to explore the four cardinal directions.
   - Since the side of an arrow tile is 30 pixels, the sprite should move 32 pixels before cloning itself; the extra 2 pixels provide a small space between tiles.
   - After it clones itself, it should return to its original position and orient itself to face the next direction. Remember to change its costume as well.

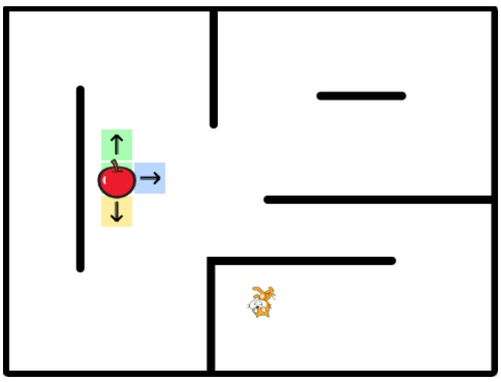
   Running the My Block shown below should display four arrow tiles facing away from the apple.



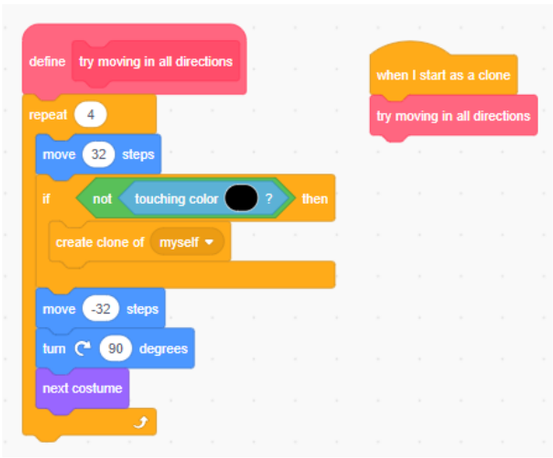


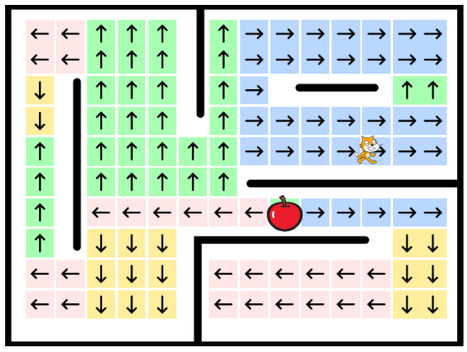
3. It works as expected — although one of the tiles hits a maze boundary. To resolve this, enclose the creation of a clone inside an `if` block that serves as a guard.



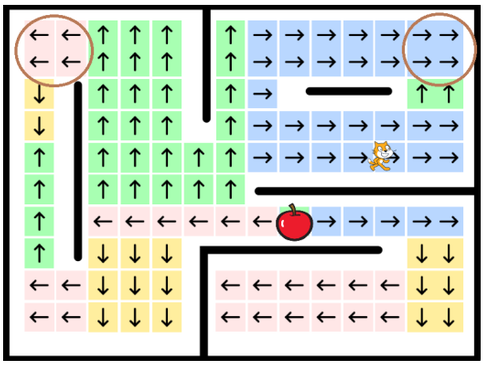


4. To fill the entire grid with arrow tiles, create a recursive procedure that intertwines creating a clone (`when I start as a clone`) and trying to move in all four cardinal directions. The base cases of this recursive procedure are as follows:
   - We stop creating clones when the sprite touches something black (the color of the maze boundaries). Since the arrows on the tiles are also colored black, this condition also prevents the sprites from stacking on top of each other.
   - Scratch itself restricts the number of clones that can be created to 300.

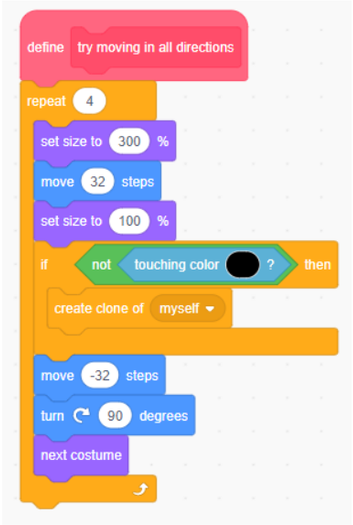


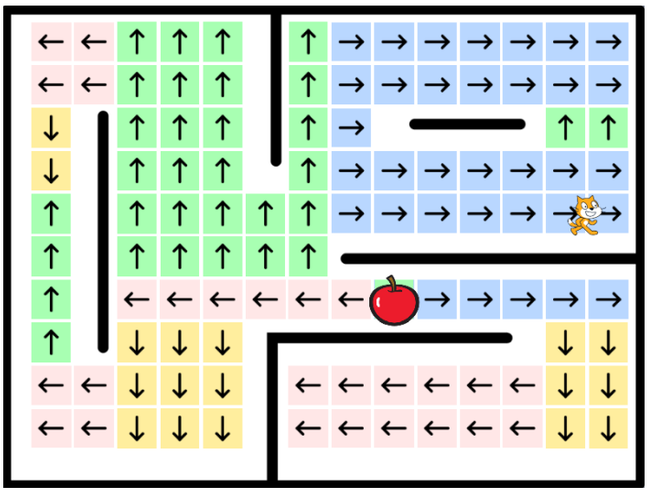


5. Nice! However, there appears to be a slight problem. The tiles at the borders seem to be fused for some reason.

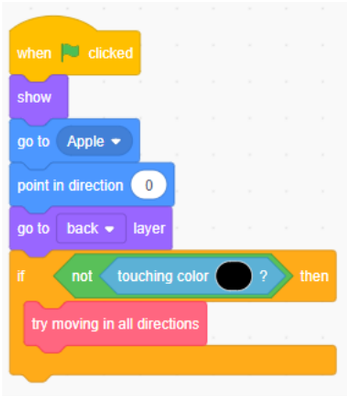


6. This issue is a result of how Scratch fences the sprites on the stage. A workaround is to set the size of the sprite to 300% but scale it back to 100% before cloning.





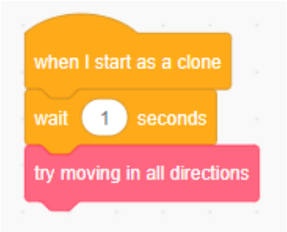
7. Call the My Block `try moving in all directions` when the flag is clicked.



💭 Checkpoint

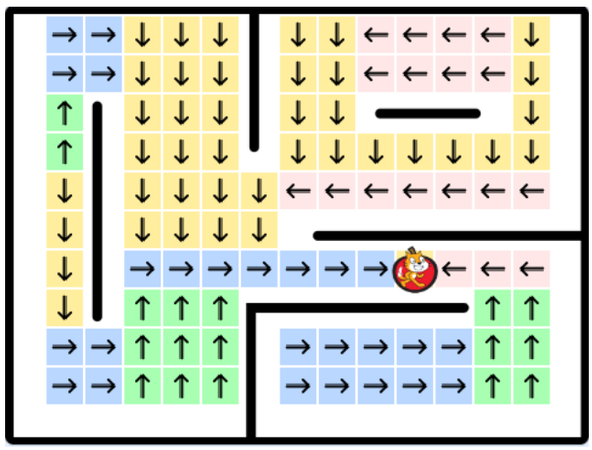
Does our maze solving procedure employ depth-first search or breadth-first search?

To help you visualize how the stage is filled with arrow tiles, try inserting a `wait` block, as seen in the figure on the right. You may also watch this video.



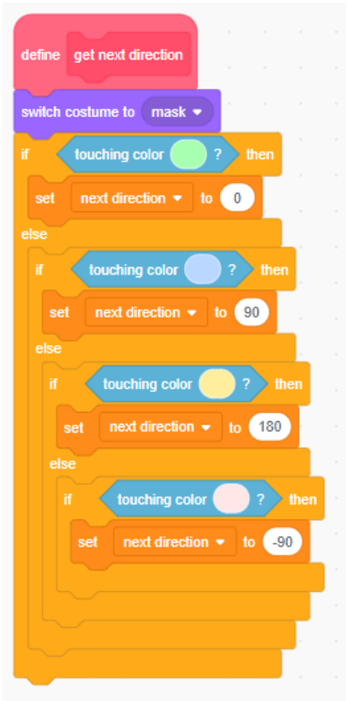
C. Creating the Cat AI

1. Horizontally flip the four costumes of the Path sprite. Running our previously created blocks should now generate tiles that point to the apple.

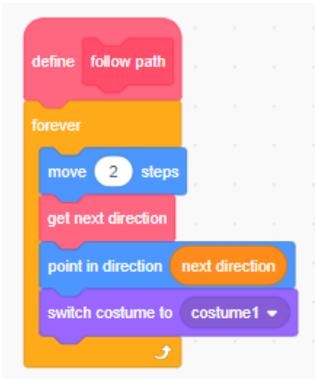


2. Create a costume named Mask for the Cat sprite, and draw a filled square tile that is 48 pixels long (spanning 12 boxes in the drawing area). Make sure to position it at the center of the drawing area.

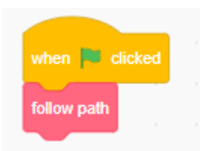
3. Create a My Block for the **Cat sprite**, and name it `get next direction`:
   - Switch its costume to the mask.
   - Create a new variable named `next direction` to denote the next direction that the cat will travel. Recall how we colored each of the costumes of the Path sprite differently; the value of `next direction` depends on which color the Cat sprite is touching.



4. Note that the only job of `get next direction` is to update the `next direction` variable. It does not try to change the actual direction in which the cat is facing. This responsibility belongs to a new My Block, which we will name `follow path`:
   - Run it with screen refresh.
   - Move the cat forward, and orient the cat to the next direction.
   - Switch back the costume to that of a cat.

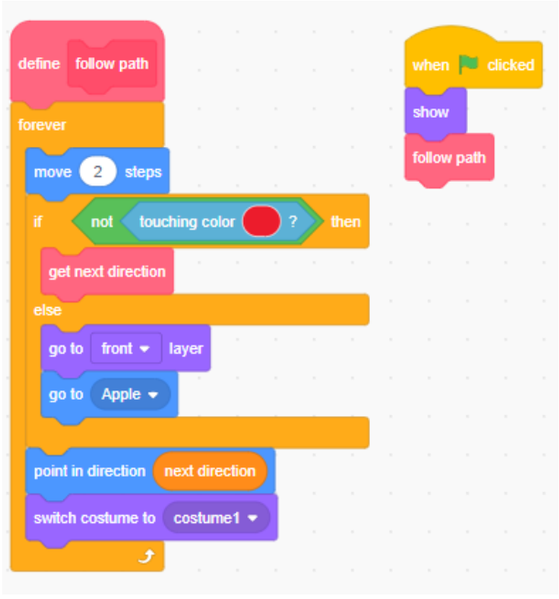


5. Call the My Block `follow path` when the flag is clicked.

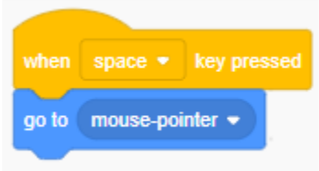


D. Final Touches

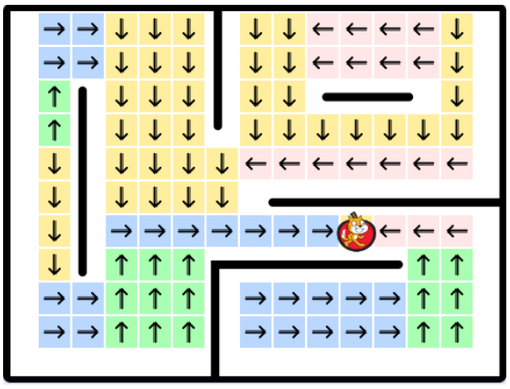
1. Running our code shows that we are near the finish line! However, when the cat reaches the apple, it becomes trapped and begins jittering. To resolve this, switch directions only when the cat has not yet reached its goal — but, once it touches the apple, it excitedly "jumps" on top of it (go to `front` layer).



2. To complete our project, position the cat to where the mouse pointer is every time the spacebar is clicked. This gives us a fun and nifty way to specify the starting position of our cat.



3. Enjoy playing with our automatic maze solver! Click the image below to play the video of our final output.

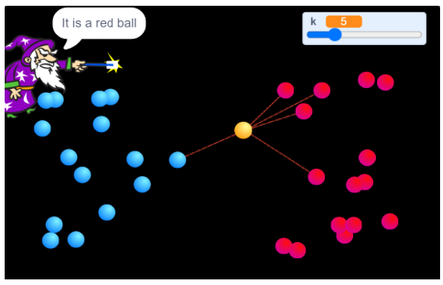


There are several ways to enhance this project. Aside from trying out more complex maze patterns, it may also be interesting to try to "smoothen" the movement of the cat (especially when it turns at junctions) or to find a way to hide the arrow tiles.

Asian MathSci League, Inc (AMSLI)
Partner: National Olympiad in Informatics Philippines (NOI.PH)
Email address: amsliphil@yahoo.com / ask@noi.ph
Contact Nos: +632-9254526 +63906-1186067

Machine Learning *k*-Nearest Neighbors

Suppose that you are a weather reporter assigned to the town of Akihabara. However, for some reason, it has not yet submitted its daily weather report!



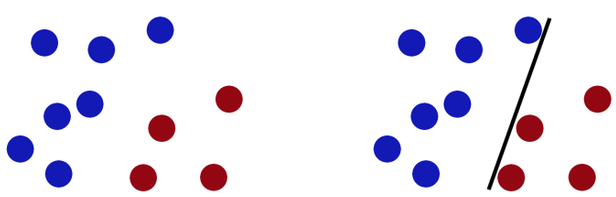
Fortunately, your station has already received reports from Akihabara's neighboring towns. You then decide to go over the reports from the 7 closest towns. The sun is shining brightly in 5 of these towns while heavy rain is pouring in the 2 remaining towns. From the data that you have gathered, you conclude that it is most likely sunny in Akihabara.

This is the idea behind the first machine learning algorithm that we will introduce in this handout: ***k*-nearest neighbors**. The classification of an unknown object is determined by performing a majority vote on the classifications of the *k* objects that are closest to it. Choosing the best value of *k* is beyond the scope of our discussion, but, if you are interested, you might want to look up the term **cross-validation**.

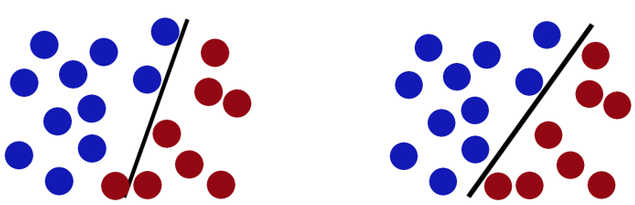
> Open the Scratch file k-Nearest Neighbors.sb3 for an interactive demonstration of this machine learning algorithm[2].

Machine Learning Support Vector Machine

Support vector machine is another machine learning technique that is used to determine the classification of an unknown object. Suppose that you have several balls on top of a table and you plan on using a stick to separate them based on color[3]:

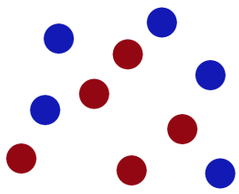


Your naughty friend decides to add another red ball but places it on the side of the blue balls. This forces you to adjust the placement of the stick:

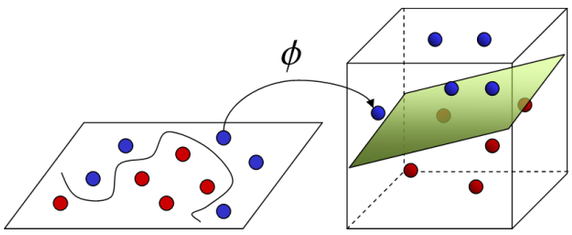


[2] Adapted from https://scratch.mit.edu/projects/235109538/editor/
[3] Adapted from https://www.reddit.com/r/MachineLearning/comments/15zrpp/please_explain_support_vector_machines_svm_like_i/

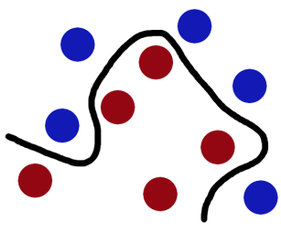
Your naughty friend still has tricks up his sleeve, and he disarranges the balls in such a way that it is impossible to separate them using a humble stick:



You formulate a superb strategy by tossing the balls into the air and swiftly inserting a sheet of paper to separate the blue balls from the red balls. In mathematical language, you transformed the problem from a two-dimensional space to a three-dimensional one:



From your naughty friend's point of view, the paper will appear as a curve separator:



More formally:

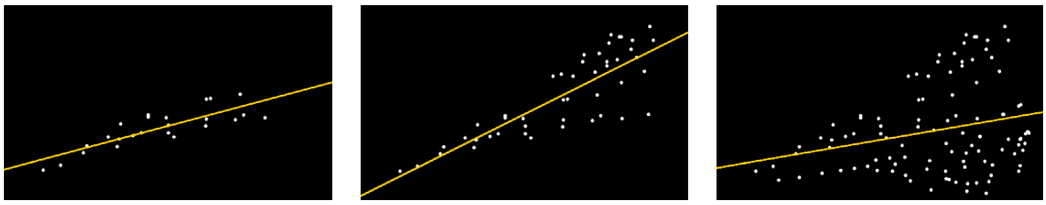
- The stick and, later on, the sheet of paper for separating the balls is called a hyperplane.
- The strategy of tossing the balls translates to applying a kernel function.
- Note that, in adjusting the stick (or the sheet of paper), we do not need all the balls. It is enough to consider only those balls that are close to the stick's current position. We refer to these balls as the support vectors.

Machine Learning Linear Regression

Unlike the first two machine learning algorithms, which are utilized for classification, linear regression is intended for prediction.

Suppose that 2 people can bake 7 cupcakes, 3 people can bake 10 cupcakes, 4 people can bake 13 cupcakes, and 5 people can bake 16 cupcakes. How many cupcakes can 100 people bake? You may have noticed that the number of cupcakes that x people can bake follows a pattern: 3 times x plus 1. Therefore, 100 people can bake 301 cupcakes.

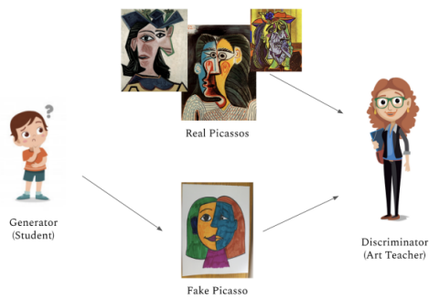
These patterns are termed **linear equations**. We will gloss over the mathematical details, but, for now, it is enough to view them as equations that can be interpreted as lines when plotted on a Cartesian plane. As the system learns more information (represented in the figures below as dots), it automatically adjusts the line to better capture the trend.



Open the Scratch file Linear Regression.sb3 for an interactive demonstration of this machine learning algorithm[4].

Machine Learning Generative Adversarial Network

Introduced by Ian Goodfellow and his colleagues in 2014, the idea behind a **generative adversarial network** (GAN) can be likened to a student and an art teacher[5]. The teacher asks the student to draw a painting of Pablo Picasso. But the student has never heard of Picasso, so he just comes up with a random scribble.

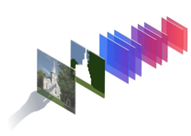


The teacher sees his work and decides whether to accept or reject it. She also offers some pieces of advice from time to time (but she never shows him an actual work of Picasso). This cycle is repeated over and over, and the student gradually improves with each attempt. At the end of almost a hundred tries, he finally wins his teacher's approval!

We are now equipped to understand the name *generative adversarial network,*

- The student is the **generator**.
- The teacher is the **discriminator**.
  It is the critique or, in other words, the "**adversary**" (enemy) of the generator.
- The teacher and the student represent **neural networks.**
  A neural network is an extremely fascinating machine learning technique inspired by how our nerve cells or neurons work.

To explore how GANs are used to create computer-generated artwork, you may experiment with this project by the MIT-IBM Watson AI Lab: https://gan-paint-demo.mybluemix.net/



`-- return 0; --`

[4] Adapted from https://scratch.mit.edu/projects/376884515/editor/
[5] Adapted from https://raise.mit.edu/PDFs/whataregans_eaai2021.pdf