Unplugged

Lesson plans for offline (unplugged) Hour of Code activities.

Lesson 1: Programming Unplugged: My Robotic Friends Relay

This activity will begin with a short review of "My Robotic Friends," then will quickly move to a race against the clock, as students break into teams and work together to write a program one instruction at a time.

Lesson 2: Text Compression

At some point we reach a physical limit of how fast we can send bits and if we want to send a large amount of information faster, we have to find a way to represent the same information with fewer bits - we must compress the data.

Lesson 3: Simple Encryption

In this lesson, students are introduced to the need for encryption and simple techniques for breaking (or cracking) secret messages. Students try their own hand at cracking a message encoded with the classic Caesar cipher and also a Random Substitution Cipher. Students should become well-acquainted with idea that in an age of powerful computational tools, techniques of encryption will need to be more sophisticated. The most important aspect of this lesson is to understand how and why encryption plays a role in all of our lives every day on the Internet, and that making good encryption is not trivial. Students will get their feet wet with understanding the considerations that must go into making strong encryption in the face of powerful computational tools that can be used to crack it. The need for secrecy when sending bits over the Internet is important for anyone using the Internet.

Lesson 4: Dance Party: Unplugged

Unplugged | Events

Have a class dance party to learn about events.

Lesson 1: Programming Unplugged: My Robotic Friends Relay

Overview

This activity will begin with a short review of "My Robotic Friends," then will quickly move to a race against the clock, as students break into teams and work together to write a program one instruction at a time.

Purpose

There are many important components to this lesson. Students will be able to run around and get their wiggles out while building teamwork, programming, and debugging skills. Teamwork is very important in computer science. While **Pair Programming - Student Video** is common, it is more common for computer scientists to work in teams. These teams write and debug code as a group rather than individuals. In this lesson, students will learn to work together while being as efficient as possible.

This activity also provides a sense of urgency that will teach them to balance their time carefully and avoid mistakes, but not to fall too far behind.

Agenda

Warm Up

- **Introduction**

Main Activity (15 min)

- **Relay Programming Activity**

Wrap Up (15 min)

- **Flash Chat: What did we learn?**
- **Journaling**

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Practice communicating ideas through code and symbols.
- Use teamwork to complete a task.
- Verify the work done by teammates to ensure a successful outcome.

Preparation

- [ ] Read **My Robotic Friends - Teacher Prep Guide**.
- [ ] Locate a wide open space for this activity, such as the gym or outdoor field.
- [ ] Print out one **My Robotic Friends - Symbol Key** per group. This is "code" to be used.
- [ ] **Paper Trapezoid Template - Manipulatives** are provided if your class is not going to use cups.
- [ ] Print out one set of **Stacking Cup Ideas - Manipulatives** per group.
- [ ] Make sure each student has a**Think Spot Journal - Reflection Journal**.

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teacher

- **My Robotic Friends** - Teacher Prep Guide

For the Students

- **My Robotic Friends** - Symbol Key
- **Stacking Cup Ideas** - Manipulatives
- **Paper Trapezoid Template** - Manipulatives
- **Think Spot Journal** - Reflection Journal

Vocabulary

- **Algorithm** - A list of steps to finish a task.
- **Bug** - Part of a program that does not work correctly.
- **Debugging** - Finding and fixing problems in an algorithm or program.

Teaching Guide

Warm Up

Introduction

Recall that in "My Robotic Friends" we guided our teammate's Automatic Realization Machine (ARM) using arrows. Take a moment to go through a quick "My Robotic Friends" example as a reminder. It can either be one that you have already covered or one that is new.

We are going to do the same kind of thing today, but instead of controlling each other, we are going to work together to create a program one symbol at a time.

Main Activity (15 min)

Relay Programming Activity

The practice lesson was easy enough; let's add some action! We're going to do the same type of thing (create a program describing how the cups are stacked) but now we're going to do it in relay teams, one symbol at a time.

The rules of this game are simple:

- Divide students into groups of 3-5.
- Have each group queue up relay-style.
- Place an identical stack of cups at the other side of the room/gym/field from each team.
- Have the first student in line dash over to the cups, review it, and write down the first symbol in the program to reproduce that stack.
- The first student then runs back and tags the next person in line, then goes to the back of the queue.
- The next person in line dashes to the stack of cups, reviews the stack, reviews the program that has already been written, then either debugs the program by crossing out an incorrect symbol, or adds a new one. That student then dashes back to tag the next person, and the process continues until one group has finished their program.

Clarifications

Here are some clarifications that need to be shared from time to time:

- Only one person from each group can be at the image at one time.
- It is okay to discuss algorithms with the rest of the group in line, even up to the point of planning who is going to write what when they get to the cups.
- When a student debugs a program by crossing out an incorrect instruction (or a grouping of incorrect instructions) this counts as their entire turn. The next player will need to figure out how to correct the removed item.

First group to finish with a program that matches the stack of cups is the winner! Play through this several times, with images of increasing difficulty.

Wrap Up (15 min)

Flash Chat: What did we learn?

- What did we learn today?
- What if we were each able to do five symbols at a time?
  - How important would it be to debug our own work and the work of the programmer before us?
  - How about with 10 symbols?
  - 10,000? Would it be more or less important?
- Is it easier or harder to have multiple people working on the same program?
- Do you think people make more or fewer mistakes when they're in a hurry?

- If you find a mistake, do you have to throw out the entire program and start over?

Journaling

Having students write about what they learned, why it's useful, and how they feel about it can help solidify any knowledge they obtained today and build a review sheet for them to look to in the future.

Journal Prompts:

- What was today's lesson about?
- How did you feel during today's lesson?
- How did teamwork play a role in the success of writing today's program?
- How did you use your debugging skills in today's lesson?

Extended Learning

Use these activities to enhance student learning. They can be used as outside of class activities or other enrichment.

Pass the paper

- If you don't have the time or room for a relay, you can have students pass the paper around their desk grouping, each writing one arrow before they move the paper along.

Fill It, Move It

- As the teacher, create a stack of cups with as many cups as children in each group.
- Have the students write as many symbols in the program as it takes to get to the next cup (including putting the cup down) before passing to the next person.

Debugging Together

Create a stack of cups at the front of the room. Have each student create a program for the stack. Ask students to trade with their elbow partner and debug each other's code.

- Circle the first incorrect step, then pass it back.
- Give the students another chance to review and debug their own work.
- Ask for a volunteer to share their program.

Ask the class:

- How many students had the same program?
- Anyone have something different?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 2: Text Compression

Overview

At some point we reach a physical limit of how fast we can send bits and if we want to send a large amount of information faster, we have to find a way to represent the same information with fewer bits - we must **compress** the data. In this lesson, students will use the Text Compression Widget to compress segments of English text by looking for patterns and substituting symbols for larger patterns of text.

Purpose

The basic principle behind compression is to develop a method or protocol for using fewer bits to represent the original information. The way we represent compressed data in this lesson, with a “dictionary” of repeated patterns is similar to the **LZW compression scheme**, but it should be noted that LZW is slightly different from what students do in this lesson. Students invent their own way here. LZW is used not only for text (zip files), but also with the GIF image file format.

Agenda

Getting Started (5-7 mins)

- **Warm up: Abbr In Ur Txt Msgs (5-7 mins)**

Activity (45 mins)

- **Decode this Mystery Text (10-15 mins)**
- **Use the Text Compression Widget**

Wrap-up (10 mins)

- **Discuss properties and challenges with compression**

Extended Learning

View on Code Studio

Objectives

Students will be able to:

- Collaborate with a peer to find a solution to a text compression problem using the Text Compression Widget (lossless compression scheme).
- Explain why the optimal amount of compression is impossible or “hard” to identify.
- Explain some factors that make compression challenging.
- Describe the purpose and rationale for lossless compression.

Preparation

- [ ] Test out the Text Compression Widget
- [ ] Review the teaching tips to decide which options you want to use

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teacher

- **Activity Recap - Decode this Message** - Activity Recap **Make a Copy**

For the Students

- **Decode this message** - Activity Guide **Make a Copy**
- **Activity Guide - Text Compression** - Activity Guide **Make a Copy**
- **Video: Text Compression with Aloe Blacc** - Video (**download**)
- **Text Compression Widget on Code Studio** - Widget

Vocabulary

- **Lossless Compression** - a data

compression algorithm that allows the original data to be perfectly reconstructed from the compressed data.

Teaching Guide

Getting Started (5-7 mins)

Warm up: Abbr In Ur Txt Msgs (5-7 mins)

Prompt:

- **"When you send text messages to a friend, do you spell every word correctly?"**
  - Do you use abbreviations for common words? List as many as you can.
  - Write some examples of things you might see in a text message that are not proper English.

Give students a minute to write, and to share with a neighbor?

Discussion

As a warm up to thinking about Text Compression, connect to ways that most people already compress text in their lives, through abbreviations and acronyms with which most people have some experience in text messages.

Motivate some ideas about why someone would want to compress text.

- **"Why do you use these abbreviations? What is the benefit?"**
  - Possible answers:
    - to save characters/keystrokes
    - to hide from parents/teachers
    - to be cool, clever, funny
    - to "speak in code"
    - to say the same thing in less space

What's this about? - Compression: Same Data, Fewer Bits

> - Today's class is about **compression**
> - When you abbreviate or use coded language to shorten the original text, you are "compressing text." Computers do this too, **in order to save time and space.**
> - The art and science of compression is about figuring out how to represent the SAME DATA with FEWER BITS.
> - Why is this important? One reason is that storage space is limited and you'd always prefer to use fewer bits if you could. A much more compelling reason is that there is an upper limit to how fast bits can be transmitted over the Internet.
> - What if we need to send a large amount of text faster over the Internet, but we've reached the physical limit of how fast we can send bits? Our only choice is to somehow capture the same information with fewer bits; we call this **compression**.

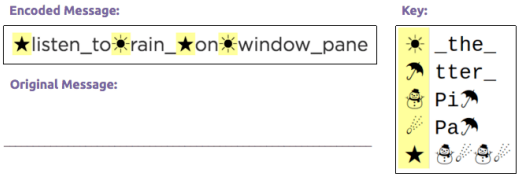
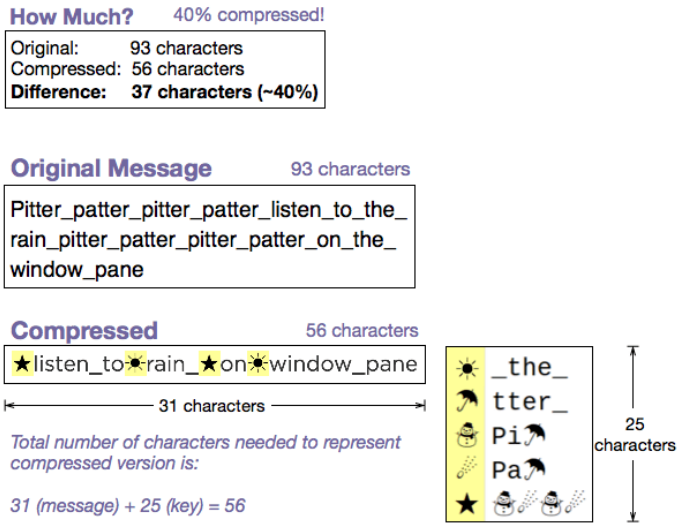
Transition:

Let's look at an example of a text message that's been compressed in a clever way.

Activity (45 mins)

Decode this Mystery Text (10-15 mins)

> - **Distribute or Display** the Activity guide: **Decode this message - Activity Guide**
> - Put students into partners or work individually.
> - **Task:** What was the original text?
> - Give students a few minutes to decode the text. The text should be a short poem (see activity recap below)

Student Activity Guide	Activity Recap
Distribute or Display Activity guide: **Decode this message - Activity Guide**	(Display or draw yourself) Activity Recap: **Activity Recap - Decode this Message - Activity Recap**
	

Recap: How much was it compressed?

To answer, we need to compare the number of characters in the original poem to the number of characters needed to represent the compressed version.

Let's break it down.

- **Display** or **Demonstrate** yourself ideas from: **Activity Recap - Decode this Message - Activity Recap** (shown in table above)
- **Important Note**:
  - The compressed poem is not **just** this part: ★listen_to☀rain_★on☀window_pane If you were to send this to someone over the Internet they would not be able to decode it.
  - The full compressed text includes BOTH the compressed text**and** the key to solve it.
  - Thus, you must account for the total number of characters in the message**plus** the total number of characters in the key to see how much you've compressed it over the original.

Transition

Now you're going to get to try your hand at compressing some things on your own.

Use the Text Compression Widget

Video: Text Compression with Aloe Blacc - Video

- Video explains compression
- Demonstrates the use of the Text Compression Tool.
- NOTE: This video pops up automatically when students visit the text compression stage in Code Studio.

Content Corner

The video explains a little bit about compression in general - the difference between lossless compression and lossy compression. Todays class is about **lossless compression** we'll do lossy compression in a class or two after looking at image encoding.

- Divide students into groups of 2
- Assign each pair one of the poems provided and challenge them, as a pair to compress their poem as much as possible.

- Deliver or put simple instructions on the board so students can follow.
  - **Challenge**: compress your assigned poem as much as possible.
  - Compare with other groups to see if you can do better.
  - Try to develop a general strategy that will lead to a good compression.
- After some time, have pairs that did the same poem get together to compare schemes. As a group their job is to come up with the best compression for that poem for the class.

- Optionally: you may hand out **Text Compression (optional) - Activity Guide**, which also explains the instructions and gives students tasks. It may work well as an out-of-class activity or assessment.

Teaching Tip

Teacher's Choice whether to show the video to the whole class or let students watch it from within Code Studio. There are benefits and drawbacks to each.

Option to Consider: Get students into the text compression tool BEFORE showing the video. You might find students are more receptive to some of the information in the video if they have tried to use the tool first.

Communication and Collaboration: To develop communication and collaboration between students, include one of the following scenarios in class:

- Have students who were assigned the same poem compare results, or seat them in the same area of the room.
- Have a little friendly competition - but be careful not to let "bad" competition seep in - to see which pair can compress a poem the most. Use a poem that none of the students have compressed yet.
- For each poem, have the group(s) who did it figure out the best in the class, and record it on the board or somewhere that people can see.
  - Have a class goal of getting the compression percentages for the four poems as high as possible.
  - The groups with the best compression percentages may be asked to share their strategy with the class.

Students may be reluctant to share if they feel they don't have the best results, but students should see others' work and offer advice and strategies.

Wrap-up (10 mins)

Discuss properties and challenges with compression

Ask groups to pause to discuss the questions at the end of the activity.

Prompts:

- **"What makes doing this compression hard?"**
  - Invite responses. Some of these issues should surface: You can start in lots of different ways. Early choices affect later ones. Once you find one set of patterns, others emerge.
  - There is a tipping point: you might be making progress compressing, but at some point the scale tips and the dictionary starts to get so big that you lose the benefit of having it. But then you might start re-thinking the dictionary to tweak some bits out.
- **"Do we think that these compression amounts that we've found are the the best? Is there a way to know what the best compression is?"**
  - We probably don't know what's best.
  - There are so many possibilities it's hard to know. It turns out the only way to guarantee perfect compression is brute force. This means trying every possible set of substitutions. Even for small texts this will take far too long. The "best" is really just the best we've found so far.
- **"But is there a process a person can follow to find the best (or a pretty good) compression for a piece of text?"**
  - Yes, but it's imprecise -- you might leave this as a lingering question.

Extended Learning

Real World: Zip Compression

- Experiment with zip using text files with different contents. Are the results for small files as good as for large files? (On Macs, in the Finder choose "get info" for a file to see the actual number of bytes in the file, since the Finder display will show 4KB for any file that's less than that.)

- Warning: results may vary. Zip works really well for text, but it might not compress other files very well because they are already compressed or don't have the same kinds of embedded patterns that text documents do.

Challenge: Research the LZW algorithm

- .zip compression is based on the **LZW Compression Scheme**
- While the idea behind the text compression tool is similar to LZW (zip) algorithm, tracing the path of compression and decompression is somewhat challenging. Learning more about LZW and what happens in the course of this algorithm would be an excellent extension project for some individuals.

Standards Alignment

CSTA K-12 Computer Science Standards (2011)

- **CL** - Collaboration
- **CPP** - Computing Practice & Programming
- **CT** - Computational Thinking

Computer Science Principles

- **2.1** - A variety of abstractions built upon binary sequences can be used to represent all digital data.
- **2.2** - Multiple levels of abstraction are used to write programs or create other computational artifacts
- **3.1** - People use computer programs to process information to gain insight and knowledge.
- **3.3** - There are trade offs when representing information as digital data.

Lesson 3: Simple Encryption

Overview

In this lesson, students are introduced to the need for encryption and simple techniques for breaking (or cracking) secret messages. Students try their own hand at cracking a message encoded with the classic Caesar cipher and also a Random Substitution Cipher. Students should become well-acquainted with idea that in an age of powerful computational tools, techniques of encryption will need to be more sophisticated. The most important aspect of this lesson is to understand how and why encryption plays a role in all of our lives every day on the Internet, and that making good encryption is not trivial. Students will get their feet wet with understanding the considerations that must go into making strong encryption in the face of powerful computational tools that can be used to crack it. The need for secrecy when sending bits over the Internet is important for anyone using the Internet.

Purpose

“Encryption” is a process for transforming a message so that the original is “hidden” from anyone who is not the intended recipient. Encryption is not just for the military and spies anymore. We use encryption everyday on the Internet, primarily to conduct commercial transactions, and without it our economy might grind to a halt.

This lesson gives students a first taste of the kind of thinking that goes into encrypting messages in the face of computational tools. Computational tools dramatically increase the strength and complexity of the algorithms we use to encrypt information, but these same tools also increase our ability to crack an encryption. Developing strong encryption relies on knowledge of problems that are “hard” for computers to solve, and using that knowledge to encrypt messages. As a resource, you may wish to read all of Chapter 5 of **Blown to Bits**. It provides social context which you may want to bring to your classroom.

Agenda

Getting Started (15)

- **Classic Encryption - The Caesar Cipher**

Activity (35)

- **Part 1 - Crack a Caesar Cipher**
- **Part 2 - Crack a Random Substitution Cipher**

Wrap-up (10)

- **Video: Encryption and Public Keys**
- **Discussion**

View on Code Studio

Objectives

Students will be able to:

- Explain why encryption is an important need for everyday life on the Internet.
- Crack a message encrypted with a Caesar cipher using a Caesar Cipher Widget
- Crack a message encrypted with random substitution using Frequency Analysis
- Explain the weaknesses and security flaws of substitution ciphers

Preparation

- [ ] Examine both versions of the widget

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Students

- **Encryption Widgets on Code Studio**
- **Encryption and Public Keys** - Video (**download**)

Vocabulary

- **Caesar Cipher** - a technique for encryption that shifts the alphabet by some number of characters
- **Cipher** - the generic term for a technique (or algorithm) that performs encryption
- **Cracking encryption** - When you attempt to decode a secret message without knowing all the specifics of the cipher, you are trying to "crack" the encryption.
- **Decryption** - a process that reverses encryption, taking a secret message and reproducing the original plain text
- **Encryption** - a process of encoding messages to keep them secret, so only "authorized" parties can read it.

Extended Learning

- **Random Substitution Cipher** - an encryption technique that maps each letter of the alphabet to a randomly chosen other letters of the alphabet.

Teaching Guide

Getting Started (15)

Remarks

> Secrecy is a critical part of our lives, in ways big and small. As our lives increasingly are conducted on the Internet, we want to be sure we can maintain the privacy of our information and control who has access to privileged information.
>
> Digital commerce, business, government operations, and even social networks all rely on our ability to keep information from falling into the wrong hands.
>
> We need a way to send secret messages...

Content Corner

If necessary provide context of some facts about the Internet:

- The Internet is not inherently secure.
- Packets traveling across the Internet move through many routers, each of which could be owned by different people or organizations.
- So we should **assume** all information traveling across the Internet to be public, as if written on a postcard and sent through the mail.

Classic Encryption - The Caesar Cipher

Background:

Many of the ideas we use to keep secrets in the digital age are far older than the Internet. The process of encoding a plain text message in some secret way is called Encryption

For example in Roman times Julius Caesar is reported to have encrypted messages to his soldiers and generals by using a simple alphabetic shift - every character was encrypted by substituting it with a character that was some fixed number of letters away in the alphabet.

As a result an alphabetic shift is often referred to as the Caesar Cipher.

Prompt:

- This message was encrypted using a Caesar Cipher (an "alphabetic shift").
- Let's see how long it takes you to **decode this message** (remember it's just a shifting of the alphabet):

Display or write this on the board

```
serr cvmmn va gur pnsrgrevn
```

- **Give students about 3-5 minutes** to work on cracking the message.
  - **ANSWER: "free pizza in the cafeteria" - the A-Z alphabet is shifted 13 characters.**

Recap:

- With this simple encryption technique it only took a few minutes to decode a small message.
- What if the message were longer BUT you had a computational tool to help you?!

Teaching Tip

Resist the urge to give students a tool or device to aid in cracking this message -- that's coming in the next part of the lesson! Part of the point here is that it's possible without tools. With tools it becomes trivial, as we'll see next.

If students are struggling to start here are a few strategy suggestions:

- Find a small word and try alphabetic shifts until it's clear that it's an English word
- Remember the letters aren't randomly substituted - the alphabet is just shifted.
- Once you have found the amount of shift the rest comes easily.

Activity (35)

Cracking Substitution Ciphers

In this set of activities students will use two different versions of a simple widget in Code Studio to "crack" a messages encoded with substitution ciphers, including an alphabetic shift and random substitution.

Transition to Code Studio : Encryption

Widgets on Code Studio

Content Corner

If you'd like your students to read a little bit about **Historical Cryptography** and cracking ciphers, check out 'Substitution Ciphers and Frequency Analaysis' in **Blown to Bits, Chapter 5 - Reading** pp. 165-169.

Teaching Tip

Don't rush it, but don't linger on cracking caesar ciphers. Presenting and cracking a caesar cipher should go **pretty fast**.

The widget is pretty self-explanatory. Let students figure out how to use it on their own.

The goal here is make points about cracking encryption with computational tools, and start to use some common terms.

You should move on to cracking random substitution relatively quickly.

Part 1 - Crack a Caesar Cipher

The instructions for this activity are simple - there is no handout:

- Put students in pairs/partners

Goal: Select a message encrypted with a caesar cipher and use the provided widget to "crack" it.

- Experiment with the tool - Click things, poke around, figure out what it's doing.
- **Choose one of the messages from the pull down menu and try to crack it** using the tool.
- If you want to, enter you own message, encrypt it, and have a friend decrypt it.

Give students about 5 minutes to get into the tool and crack a few messages

- Aided with the tool, cracking an alphabetic shift is trivial.
- Once you've done one, it only takes a matter of seconds to do others.

Optional - Pause and Recap:

There is a page in Code studio which recaps terminology (encryption, decryption, crack, cipher, Caesar ciper) and poses the next problem.

You may optionally pause here to recap and go over terms if you like or just let students proceed (see activity part 2 below).

Part 2 - Crack a Random Substitution Cipher

After re-capping the first activity make sure **students understand the following before proceeding:**

- **Cracking a Caesar cipher is easy...trivial with a computational tool like the one we used.**
- **The next step is to make the encryption slightly harder...**

New Challenge:

- **What if instead of shifting the whole alphabet, we mapped every letter of the alphabet to a random different letter of the alphabet? This is called a random substitution cipher.**
- **The new version of the widget you'll see is a more sophisticated version of the encryption tool that shows you lots of different stuff.**
- **But what it does is bit of a mystery!** Let's check it out...

Get Cracking

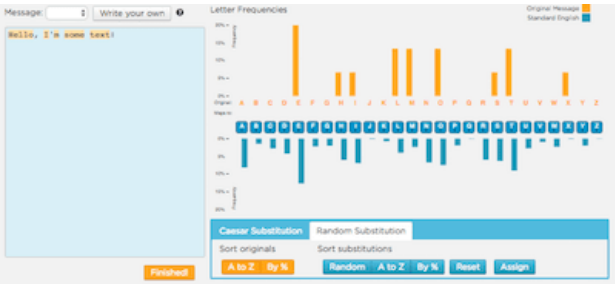
- Have students click to the next bubble to see the frequency analysis version of the widget. (It should look like the screen shown below)

Goal: let students explore for **5-10** minutes to see if they can discover what the tool is showing them and allowing them to do.

The tasks laid out for students in code studio are:

- Figure out what is going on in this new version of the tool
- What information is being presented to you?

- Figure out what the the tool let's you do
- As usual: you can't break it. So click on things, poke around.
- If you figure it out you might be able to crack a message encoded with random substitution.



- **After some exploration time regroup** to clarify what the tool is and how it works.
- **If necessary** point out to students that the next level in code studio (the one after the frequency analysis tool) explains a little bit about how frequency analysis works and suggests a few strategies for how to get started.

Give students about 15-20 minutes to crack one of the messages.

- If they finish there are more to try.
- Students can enter their own messages, do a random substitution to encrypt it, then copy/paste the encrypted version and see if a friend can crack it.
- It is possible to get pretty proficient at cracking these messages with the tool.

Use a Discovery-based approach

REMINDER: Discovery-based introduction of tools in a nutshell:

- Get students into to the tool without much or any introduction
- Give students working in partners a fixed amount of time (5 minutes or so) to poke around and see if they can figure out what it does and doesn't do – typically this might be presented as a mystery worth investigating
- Ask the group to report what they found
- Teacher fill in any gaps or explanations of how the tool works afterwards

This widget, like all others, are meant as a learning tool. You cannot break it so you are encouraged to let students play and investigate to figure out how the tools work.

These discovery-based methods of introducing tools have been tested in professional development and have worked well for teachers who use this curriculum. This method is effective for a few reasons, but overall students find this approach more engaging and fun, and they tend to be more receptive to, and motivated to hear, explanations of how the tool works after trying to "solve the mystery" themselves.

Wrap-up (10)

Video: Encryption and Public Keys

- Show the **The Internet: Encryption & Public Keys - Video**

You should know about this video:

- **0:00 to 4:11** covers Caesar and Vigenere ciphers and explains why they are hard to crack
- **After 4:11**...it explains the difference between encryption that uses **symmetric** v. **asymmetric** keys which is **related to material on public key encryption** and is intended as a preview/teaser for more modern encryption techniques.

Wrap up

The video re-iterates a number of points that came out in this lesson.

In wrapping-up, make sure students:

Understand the relationship between cryptographic keys and passwords.

- A **Key** is an input to an encryption algorithm. A password is basically the same thing.

Understand why using longer passwords makes them harder to guess.

- Longer passwords increase the number of possible keys making it **Computationally hard** to guess what the key is.

Discussion

As part of wrap up **the major points we want to draw out are:**

- Encryption is essential for every day life and activity
- The "strength" of encryption is related to how easy it is to crack a message, assuming adversary knows the technique but not the exact "key"

- A random substitution cipher is very crackable by hand though it might take some time, trial and error.
- However, when aided with computational tools, a random substitution cipher can be cracked by a novice in a matter of minutes.
- Simple substitution ciphers give insight into encryption algorithms, but as we've seen fall way short when a potential adversary is aided with computational tools...our understanding must become more sophisticated.
- If we are to create a secure Internet, we will need to develop tools and protocols which can resist the enormous computational power of modern computers.

Here are a couple of thought-provoking prompts you can use to bring closure to the lesson and as an avenue to draw out the points above. Choose one or more.

Prompts:

How much easier is it to crack a caesar cipher than a random substitution cipher? Can you put a number on it?

- **For Caesar's Cipher there are only 25 possible ways to shift the alphabet. Worst case, you only need to try 25 different possibilites. A random substitution cipher has MANY more possibilities (26 factorial = $4x10^{26}$ possibilities). However, as we learned, with frequency analysis we can avoid having to try all of them blindly.**

Was it difficult to crack a Random Substitution cipher? Did it take longer than you thought? shorter? Why?

- **Computational tools aid humans in the implementation of encryption, decryption, and cracking algorithms. In other words, using a computer changes the speed and complexity of the types of encryption we can do, but it also increases our ability to break or circumvent encryption.**

Any encryption cipher is an algorithm **for transforming plaintext into ciphertext. What about the other way around? Can you write out an algorithm for cracking a Ceasar cipher? What about a random substitution cipher?**

- **An algorithm for cracking a Caesar cipher is pretty easy - for each possible alphabetic shift, try it, see if the words come out as english.**
- **An algorithm for cracking random substitution is trickier and more nunanced. There might not be a single great answer but through thinking about it you realize how tricky it is to codify human intelligence and intuition for doing something like frequency analysis into a process that a machine can follow. It probably requires some human intervention which is an interesting point to make.**

Review of Terminolgoy -- you can use this opportunity to review new vocabulary introduced in the activity and respond to questions students may have encountered during the activity.

- Definitions of cryptography, encryption, decryption, cracking/breaking an encryption, cipher, etc.

Extended Learning

Read Blown to Bits

- Read pp. 165-169 of **Blown to Bits, Chapter 5 - Reading**.
- Answer the questions provided in the reading guide and worksheet **Reading Guide for Encryption - Worksheet**

Teaching Tips

Students should be encouraged to chat with their partner while completing the worksheet. The questions are fairly straightforward and the point is more to use the questions as a guide to the reading, than to find all the answers as quickly as possible.

More Blown to Bits

- The earlier sections of Chapter 5 of Blown to Bits make reference to the significance of and controversies surrounding encryption in the aftermath of September 11th. This reading may be a useful tool for further introducing the impact of cryptography on many aspects of modern life.

- Ask students to review the history of their Internet browsing and calculate roughly what percentage they conduct with the assumption that it is “private.” Do they have any way of being sure this is the case? Are there any websites they visit where they feel more confident in the secrecy of their traffic than others? Are they justified in this conclusion?

Standards Alignment

CSTA K-12 Computer Science Standards (2011)

- **CI** - Community, Global, and Ethical Impacts
- **CL** - Collaboration
- **CPP** - Computing Practice & Programming
- **CT** - Computational Thinking

Computer Science Principles

- **1.2** - Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.
- **3.3** - There are trade offs when representing information as digital data.
- **6.3** - Cybersecurity is an important concern for the Internet and the systems built on it.
- **7.3** - Computing has a global affect -- both beneficial and harmful -- on people and society.

Lesson 4: Dance Party: Unplugged

Unplugged | Events

Overview

Students will learn that events are a useful way to control when an action happens, and can even be used to make make multiple things act in sync. In programming, you can use events to respond to a user controlling it (like pressing buttons or clicking the mouse). Events can make your program more interesting and interactive.

Purpose

Students will learn to think about controlling actions using events. Events are widely used in programming and should be easily recognizable after this lesson.

Agenda

Warm Up (10 min)

- **Vocabulary**
- **Warming up to Events**

Main Activity (25 min)

- **Dance Rehearsal**
- **Dance Party**
- **Extensions**

Wrap Up (5 min)

- **Flash Chat: What did we learn?**

Go Viral!

View on Code Studio

Objectives

Students will be able to:

- Respond to commands given by an instructor.
- Recognize movements of the teacher as signals to start an action.
- Keep track of actions associated with different events.

Preparation

- [ ] Project the **The Big Dance Party Slides**.

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teacher

- **The Big Dance Party Slides**
  Make a Copy
- **Event Controller**
- **Spotify Playlist (all ages)**

Vocabulary

- **Event** - An action that causes something to happen.

Teaching Guide

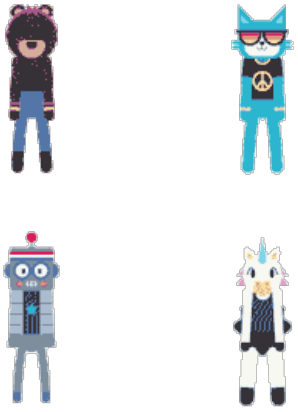
Warm Up (10 min)

Vocabulary

This lesson has one new and important vocabulary word:

Event

An event is an action that causes something to happen.



Warming up to Events

- Introduction
  - **Today we're going to have a dance party! Does anyone have a favorite dance move?**
  - **Have you ever watched a dance team perform to music together? How do they stay in sync?**
    - **One way to do this might be to plan out all of the moves in advance. It's almost like the dancers are programmed! Computer scientists would call this an algorithm because it's a list of steps to get something done.**
    - **Another way to keep in sync is to have a cue that tells everyone when to change to a different move. Everyone would still need to know what moves to perform and agree on what the cues mean.**
  - **If I want everyone in class to clap at the exact same time, I could do that by giving you a countdown from 3.** (Try it!)
  - **When I reached "1", that was the event that gave you all the signal to clap.**
- Ask the class if they can think of any other events that could give signals.
  - You may need to remind them that you're not talking about an event like a birthday party or a field trip.
  - If they have trouble, you can remind them that an event is an action that causes something to happen.
    - Blowing a whistle
    - Waving a flag
    - Saying a magic word
    - Pressing a button
- **Today, we're going to organize our class dance party using events.**

Main Activity (25 min)

Directions:

Dance Rehearsal

- Project the Dance Moves Slides onto your classroom screen.
- Practice each of the moves until students feel secure with them.
  - Consider expectations you might need to set around safety and personal space.
- When you reach the last slide, decide with your class what each button does. We suggest:
  - Green Button -> **High Clap**
  - Orange Button -> **Dab**
  - Teal Button -> **Star**

Lesson Tip

It is important to note that each move is performed continuously but also in left/right pairs. For example, when performing **Clap High** students will clap once to the left, then once to the right, then repeat that until the next move starts.

    - Purple Button -> **Body Roll**
    - Pink Button -> **This Or That**
- Practice tapping the buttons on the overhead and having your class react.
- Let your class know that every time you push a button, it is an “event” that lets them know what they are expected to start doing next.

Dance Party

- Start playing some music.
    - Check out this **Spotify Playlist**. We are using radio-safe versions of all songs. For younger students, you may want to use this further filtered list filtered list **Spotify Playlist (all ages)**.
- Use the controller buttons to have class change dance moves while the music plays.

Lesson Tip

At the beginning, give enough space between button presses for students to perform the move in both directions a couple times. You can get faster over time.

Extensions

- Mix up the dance moves using suggestions from the class. Encourage students to teach each other what they know.
    - If only a few students can perform a complex move, you can make them the lead dancers for a particular event.
        - For example, pressing one button might cause 3 students to start **flossing** while the rest of the class performs a**body roll**.

Lesson Tip

Feel free to change up the music or to re-program each of your events. This is called iteration and it's a big part of what computer scientists do!

Wrap Up (5 min)

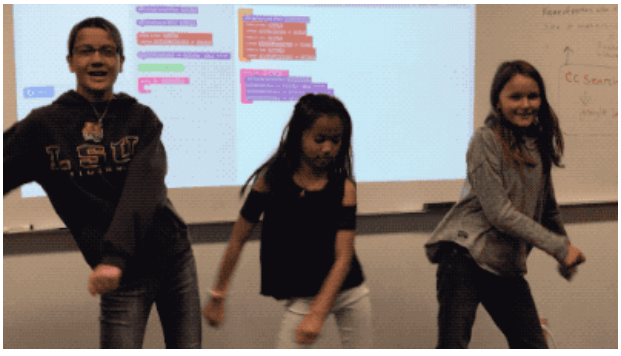
Flash Chat: What did we learn?

- Why do we need to be able to handle events in a program?
- What are some other kinds of events that you can think of?

Go Viral!

The Hour of Code is about creativity, and we can’t wait to see what you create! Please share student creations, photos, and videos on social media! Teachers, record your classroom coding a dance, or dancing the dance. Make your video special by adding an **ending clip**.

Be sure to include #HourOfCode and tag us on Facebook, Twitter and Instagram. Bonus points for tagging the artist whose music your students used. Code.org will re-share our favorite posts to our millions of followers.



Of course, make sure to respect your school's social media policy

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming