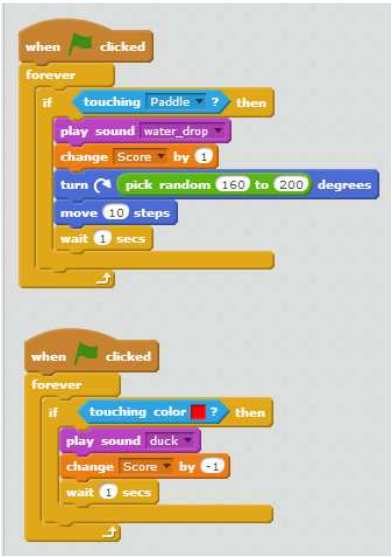
# Blocks/scripts you might use in games:

### Bouncing off an edge (Control, Motion, Sensing, and Operator Blocks)


In maze, pong, or collide type games, you usually want a sprite to bounce back or move when it touches an edge or an object. This can be accomplished by using a **Sensing** block such as the ones shown to the left. These blocks are usually placed in a <forever> block and then an <if then> block (which are **Control** blocks) along with **Motion** blocks that tell the sprite how far to move. The sample at the right tells the sprite to turn right 180 degrees and then move 10 steps. You can looked inside a Maze project that uses this script at: http://scratch.mit.edu/projects/16494083/


### Keeping Score (Data Block)




In the **Data** blocks, click on **[Make a Variable]**. Name it Score. Leave a check in the block next to [Score] so that it is visible in your project. Four blocks will be available that allow you to set, change, show and hide the Score.


You probably want to reset the score to 0 whenever the green flag is clicked as shown to the right.

In the example to the left, every time the ball sprite touches the paddle sprite, the score increases by 1.

Also, every time the ball sprite touched the red area below the paddle (misses the paddle), the score decreases by 1.

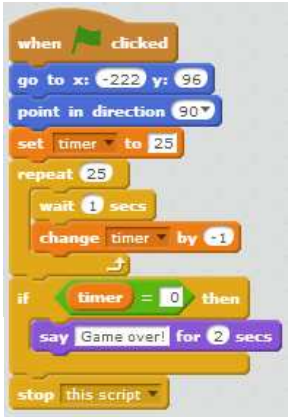
You can look inside a Pong project that uses these scripts at: http://scratch.mit.edu/projects/31884390/

### Timer (Data Block)


You can also create a timer as a variable in the **Data** blocks in the same way that you created a Score block.

In the example to the right, the timer is set to 25 when the green flag is clicked. The timer value decreases by one every second for 25 seconds and the sprite says "Game over".

If your Timer is visible on the stage, you can change the appearance by double clicking on it.





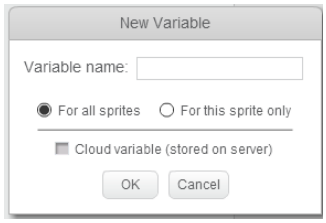
This is the initial appearance.


Double clicking on the timer changes it to this.

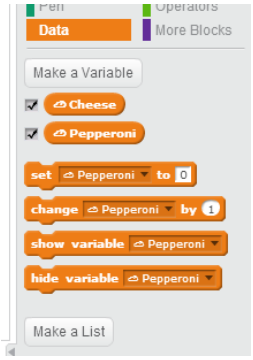

Double clicking again changes it to a slider version.

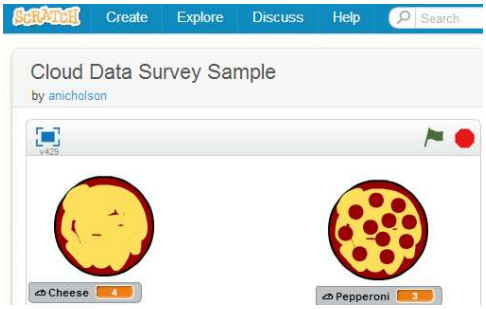
Right clicking on the timer variable gives you more options depending on which of the three versions you select, as shown to the right.

## Cloud Data (Data blocks)



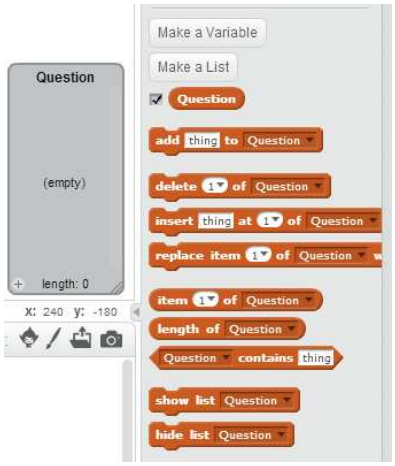

When you create a data variable, one of the options is to have it be a cloud variable. Cloud variables must be numeric values. Also, you must be an active "scratcher" for awhile before you can use Cloud Data. Cloud data variable have a cloud icon in front of the variable name. The data is stored on the server along with usernames of individuals who use the project.



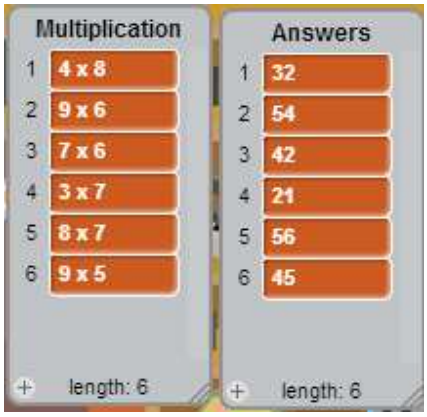




In the example to the left, people can indicate if they prefer cheese or pepperoni pizza. One person could log on to Scratch and ask a group of people to indicate their preference according to the directions on the project. Then, if they checked the cloud data, they would have a record of the information along with a time stamp. You can check this project out at: http://scratch.mit.edu/projects/36842176/

## Lists (Data blocks)




You can create lists under the **Data** blocks. Lists can be used to store or receive data. One way to use lists is to have a list for questions and another list for answers. When you create a list, several blocks are created that relate to the list. You can also decide if your list is visible by leaving a checkmark next the name of the list. In the example to the left, the list appears on the stage since there is a checkmark by the list name (Question). The list is empty, but you can click on the plus sign on the bottom left edge of the empty list and type in information. There are also blocks that will let you add, delete, or insert informationtinto the list.




To the right is an example of two lists that have been created for a math quiz. The answers must be in the same relative location as the multiplication questions.


The script to the left shows how the two lists are use in a multiplication math quiz. A **Question** variable is set to a random number from 1 to the length of the list. (The list may be added to and using "length of list" ensures that all the items will be randomized.)

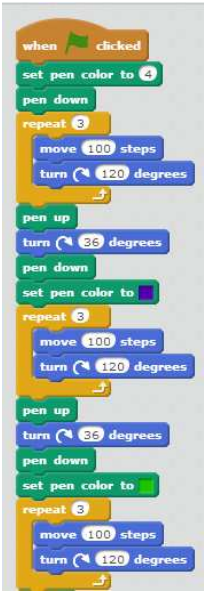
The sprite asks a multiplication question using the randomized item from the Multiplication list. Then the program checks to see if the inputted answer matches the corresponding item from the Answers list and provides appropriate feedback and scoring. You can check this project out at: http://scratch.mit.edu/projects/36698802/

## Naming Variables and Lists

It is important to give meaningful names to variables and lists, especially if you are using several in one project. One recommended convention is to use camel case for names: no spaces and capital letters for every word except the first. Shortened forms are okay as long as they make sense to you. Some examples are: numReps (for number of repetitions), lineLength (for length of line), or multProb (for multiplication problems).

**Note**: You can also right click any block in your script and add a comment to explain the block or script.

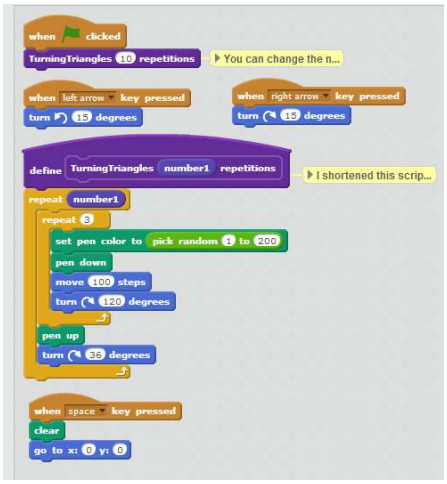
## More Blocks

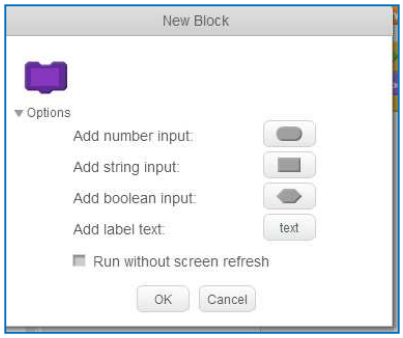



When you create complicated scripts that you use more than once in your project, you might want to create a new block. New Block is an option under the **More** Blocks.

In the example to the left, the script for drawing several triangles, turning slightly after each triangle, is long and repetitive.

In the example to the right, a TurningTriangles block has been created under **More** Blocks. Under Options, a number input was selected as well as a text label (repetitions). This block can also be copied to the Backpack and used in other projects.



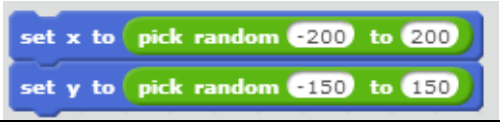
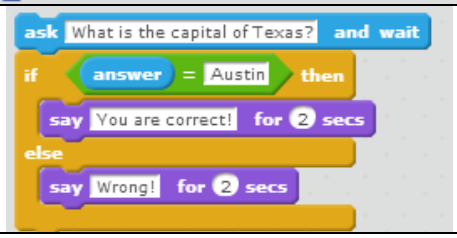

To see procedures in projects, check out the following:

http://scratch.mit.edu/projects/36834818/

http://scratch.mit.edu/projects/33415332/

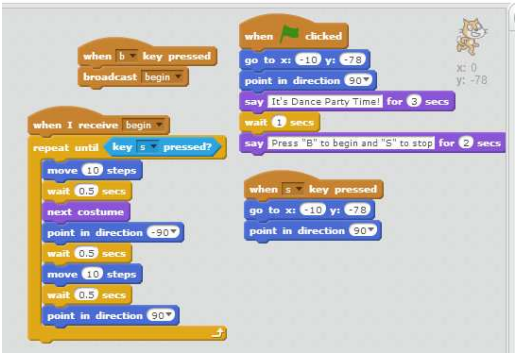
## Operators

Operator Blocks can be placed inside other blocks to randomize, compare or combine strings, round numbers, perform mathematical operations, etc. The table below shows some examples of operators added to blocks in a script.

| | |
|---|---|
|   | The **Join** operator block is added to the **Say** block from the **Look**s blocks to join text and the value of the score variable. Running this would have the sprite say "Your score is 5" (or whatever the current value of Score is). |
|   | The **Random** operator is added to the **Motion** blocks to set the x and y coordinates. This will place the sprite in a random location. |
|   | The **Equa**l operator can compare two. In the example to the left, the sprite asks a question and the **Equal** operator compares the inputted answer with the correct answer. |

## Broadcast (Events blocks)


Under the **Events** blocks, you can broadcast and receive messages. You can use the down arrow on the broadcast block to create a new message. You can use several different broadcasts in a project, so it is helpful to give them a meaningful name.

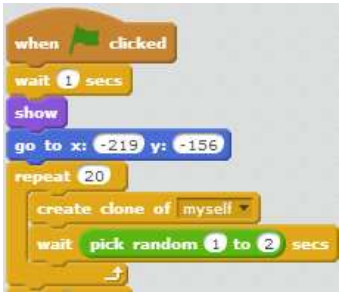



These messages can be received by any sprites in the project. This is a good way to sequence events or actions that involve more than one sprite. In the example to the left, pressing the b key broadcasts the message "begin". When this sprite receives that broadcast, he moves and changes costume. The script at the right shows what happens when a different sprite receives the broadcast "begin".

http://scratch.mit.edu/projects/36039980/


## Cloning (Control blocks)

Cloning allows you to quickly create copies of a sprite. Any scripts associated with the original sprite will be copied to all the sprites. If you change the script for the original sprite after the clones are created, the changes won't appear in the clones' scripts. The cloning blocks are under the **Control** blocks and they are: <When I start as a clone>, <create clone of ___>, and <delete this clone>.




The script to the left has a sprite appear at a certain location. Then, a clone is created every 1 to 2 seconds until 20 clones have been created.

In the script to the right, when each clone starts, it shows at a random location on the screen, but if it is touching the clone of another sprite, it moves away 50 steps. These scripts are used in a Collide game and you can check out the game and the cloning scripts at: http://scratch.mit.edu/projects/36667744/


## Code for full screen

Add #fullscreen at the end of the URL for your project to have it viewed automatically in full screen mode. For example, check out: http://scratch.mit.edu/projects/36698802/#fullscreen

## Hide and show Sprites easily while working on a project

If you are using several sprites and backgrounds, it can be a little confusing when you are figuring out where to put sprites on the stage. You can use **show** and **hide** blocks without putting them into a script to "temporarily" show and hide a sprite. Just select each sprite and drag a show block and a hide block into the script area. Whenever you need to see or hide the sprite, click the corresponding block. When you are done with the script for that sprite, you can either delete the show and hide blocks or place them in the appropriate location in the sprite's script.