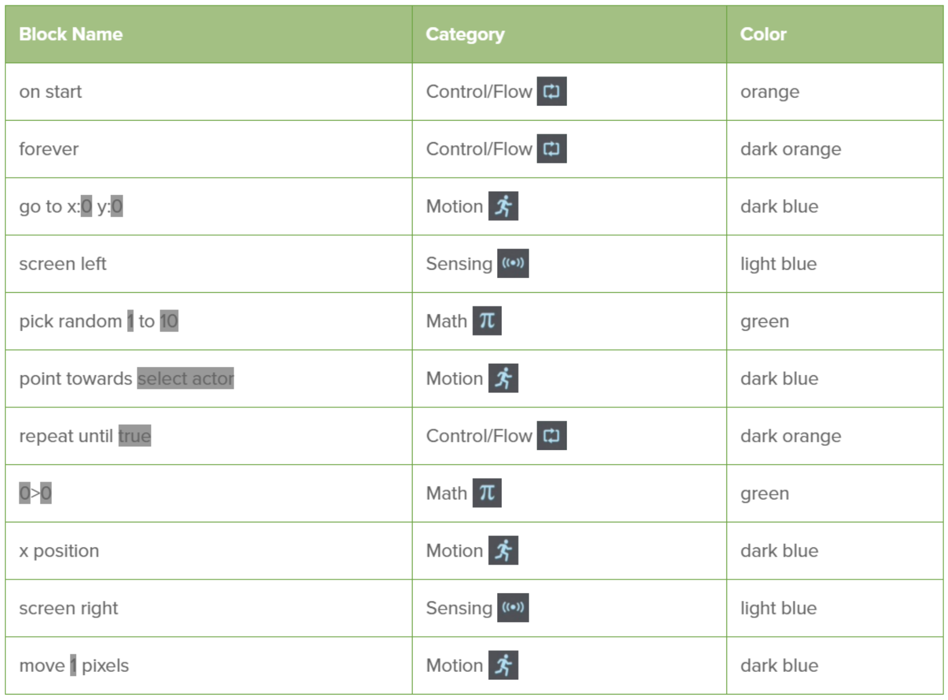
Activity 3 Building Computer Models

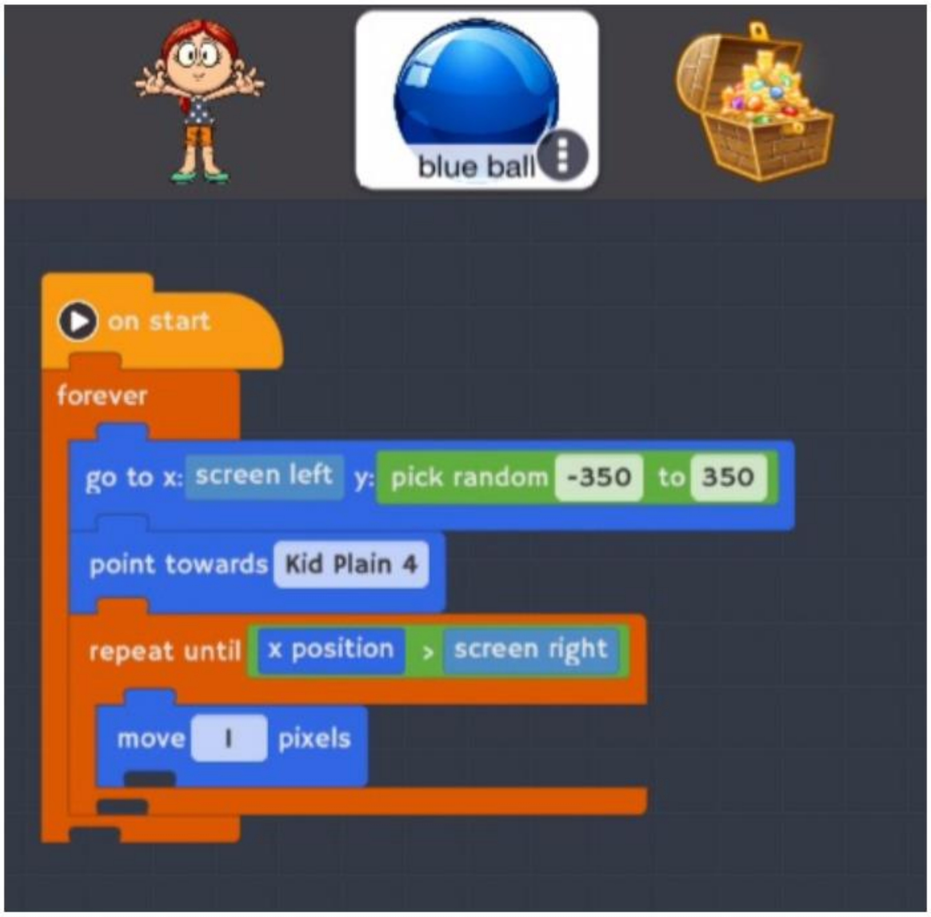
Part 4: Animate one ball and make it an obstacle for the main actor

This step adds a ball that aims for the kid every time it launches from the sideline. Later on you will learn how to make more copies of this ball. For now, just make one. This is a good example of breaking a problem down into smaller tasks.

- Ask: What is the desired behavior for the ball? When should the ball do this behavior?
    - When the game starts, the ball will start in a random location on the left side of the screen and fly across the screen to the right side of the screen. The ball should do this over and over again while the game is running.
- Program the behavior of the ball.
    - Program the ball to fly across the screen from random starting points. The following blocks show one way to make this happen.



Block Name	Category	Color
on start	Control/Flow	orange
forever	Control/Flow	dark orange
go to x:0 y:0	Motion	dark blue
screen left	Sensing	light blue
pick random 1 to 10	Math	green
point towards select actor	Motion	dark blue
repeat until true	Control/Flow	dark orange
0>0	Math	green
x position	Motion	dark blue
screen right	Sensing	light blue
move 1 pixels	Motion	dark blue



```
on start
forever
    go to x: screen left y: pick random -350 to 350
    point towards Kid Plain 4
    repeat until x position > screen right
        move 1 pixels
```

Questions About the Ball's Movement

Why is the y-coordinate a random number between –350 and 350?

- The screen is divided into a grid of x- and y-coordinates. The y-axis goes up and down the screen from –400 up to +400. Choosing a y value between –350 and +350 for the starting point of the ball makes the ball start off at different positions along the left side of the screen, but not within 50 pixels of the corners. The starting x-coordinate for the ball always stays the same. It is set to a system **variable** called *screen left*, which means the ball always starts at the leftmost, or lowest, x value.

What does it mean if x is greater than "screen right"?

- The repeat loop keeps going until the *x position* of the ball is greater than *screen right*. The ball starts off pointing at the kid so it will have a chance of hitting the kid if the player doesn't react. The *x position* of the ball increases as the ball moves across the screen. You can tell that the ball has reached the right side of the screen when its *x position* is greater than the highest number on the x-axis, which is represented by the system variable called *screen right*.

Note: If the ball goes off the top or bottom of the screen, the ball will still reach *screen right* even though you can't see it.

What does "point towards Kid Plain 4" mean?

- "Kid Plain 4" is the name of the girl character in this example. The *point towards* block tells the ball which direction it should face. The *point towards Kid* block makes the ball point in the direction of the kid. In the *repeat* block the *move 1 pixels* block makes the ball move toward the kid. *Point toward* only looks at the kid for just one moment, so the ball moves toward where the kid was in that moment. After the ball's direction has been set it does not matter if the kid moves—the ball will continue on its original path.

Part 5: Keep track of score and hits

- A very important idea in computer science is keeping track of values that change while a program runs. Things that hold values are called *variables*. This game keeps the value of the current score in a variable called "score." It also tracks the number of times the main actor has been hit by the ball in a variable called "hits."

What is a variable?

A variable is the name of a bucket that contains a value. A computer program uses a variable name to access the stored value. Separating the variable name from the value it holds allows a programmer to use a variable without knowing exactly what value it represents. The value of the variable may change while the program is running.

- In this particular game, the variables *score* and *hits* must be accessible to all of the actors in the game so they should be ***global variables.***

Global Variable

Any actor in the program can read and write to a global variable. A global variable can be created in any actor's program area.

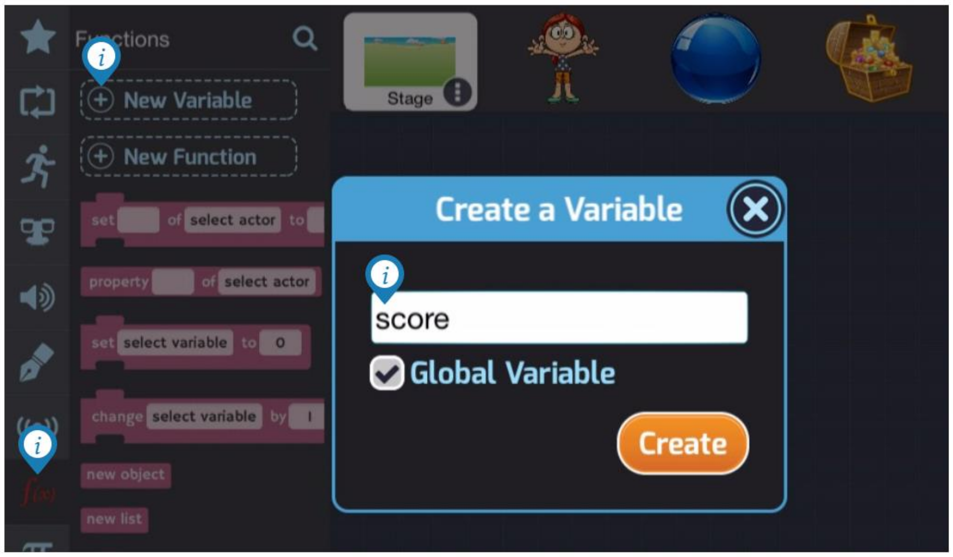
Local Variable

Only the owner of a local variable can read and write to it. A local variable is created in the program area of its owner.

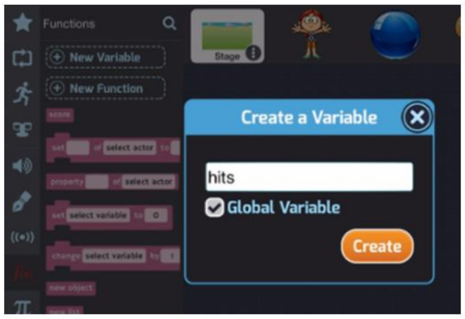
- Follow the steps below to set up two new variables.
  - Select the **Stage** in the Actors Panel and go to its **code area**.

Note: It is good practice to put code that relates to the entire program into the stage area.

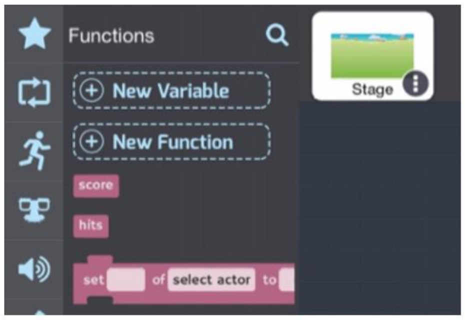
- Go to the stage, click on the Functions $f(x)$ tab on the far left of your screen, and add a new global variable named "score." Select the checkbox for Global Variable and then click Create.



- Repeat the same step to add another new variable named "hits."

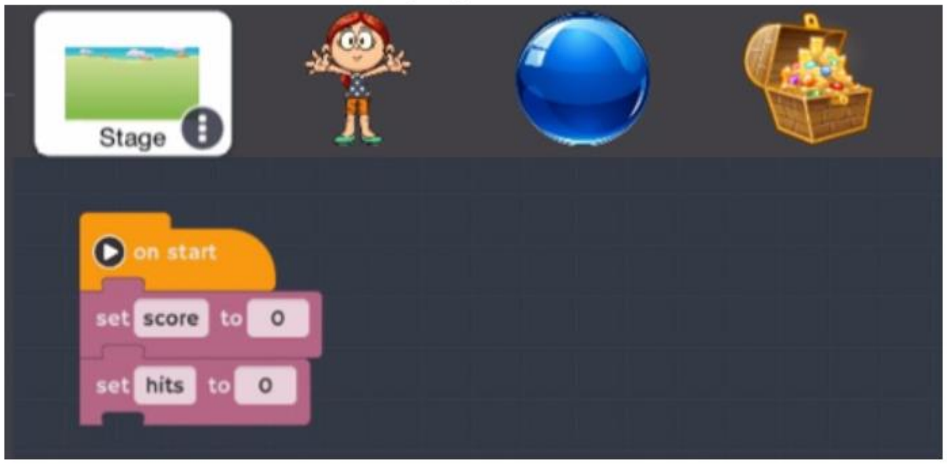


Create new Global Variable "hits."

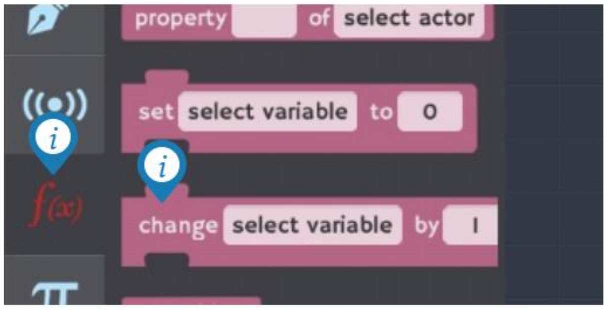


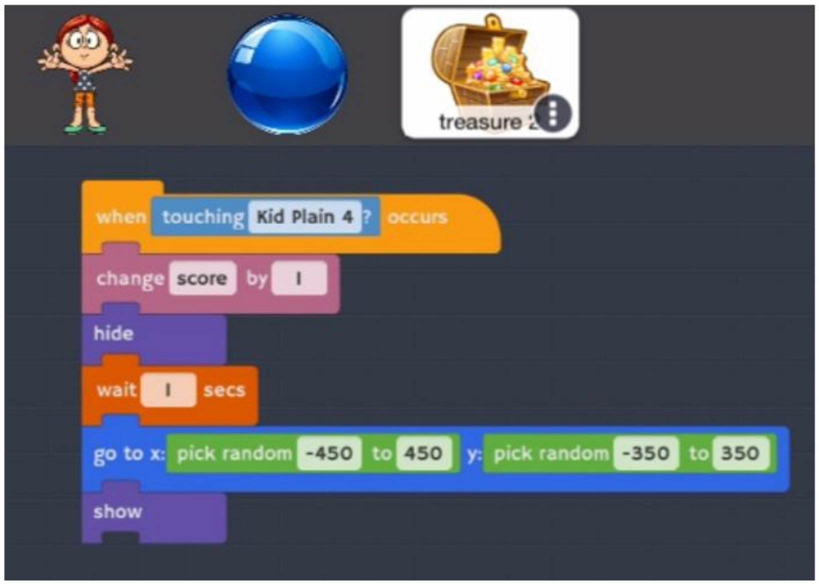
New variable blocks appear in Functions category.

- Initialize global variables *score* and *hits* to 0 as shown in the following picture. Setting the starting value of a variable is called *initializing* the variable. The *set select variable to 0* block can be found in the Functions category.



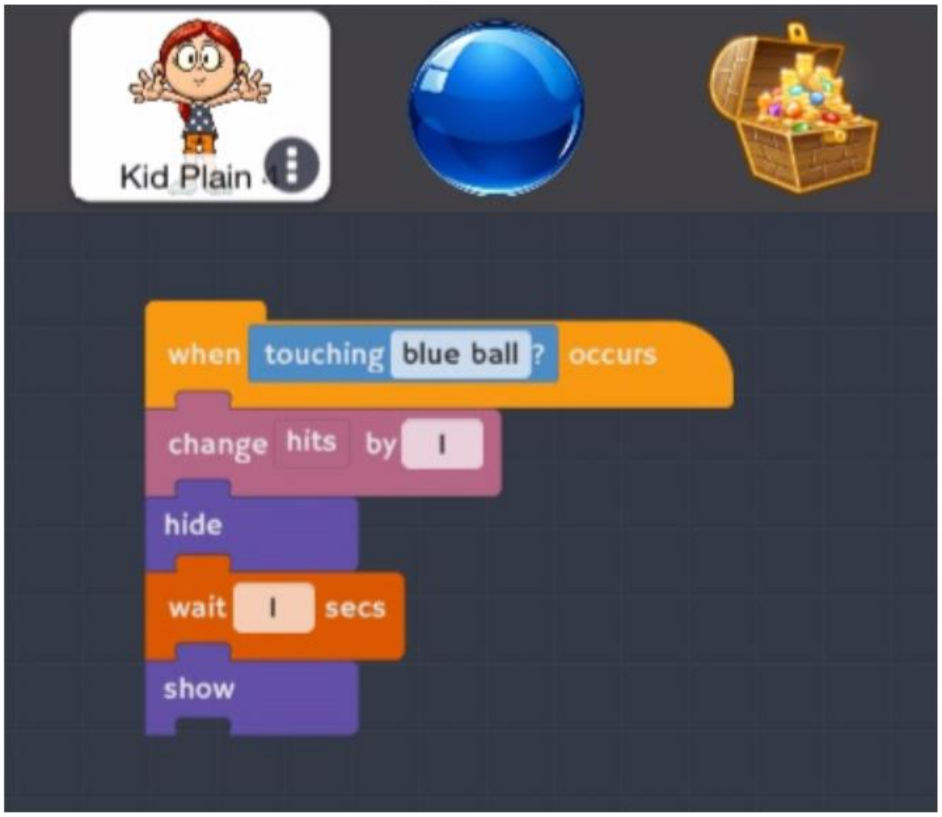
- Program the **event** when the kid captures the treasure. The treasure disappears for one second and the *score* is increased by one.
    - You have already written the code that makes the treasure disappear and reappear when it touches the kid. You only need to add one more block to **the treasure chest code** to increase *score* by 1.
    - Go to the Function category and drag out the *change select variable by 1* block. Remember to click on *select variable* and choose *score*.





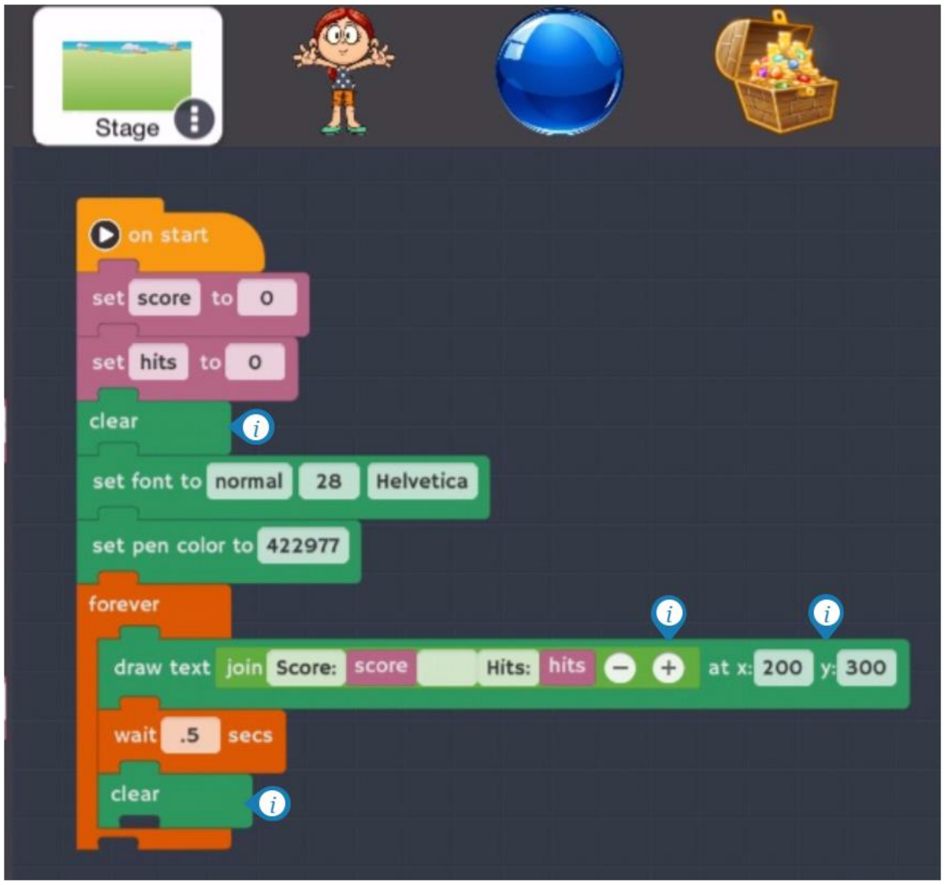
```
when touching Kid Plain 4 ? occurs
change score by 1
hide
wait 1 secs
go to x: pick random -450 to 450 y: pick random -350 to 350
show
```

- Program the **event** when the ball hits the kid. The kid disappears for one second and the *hits* variable is increased by one.
    - The procedure to track hits in the kid's code is very similar to what you did to track the score in the treasure's code. See the image that follows.



The player of the game will appreciate being able to see the score and the number of hits while playing. Add the following code blocks to the *on start* script for the **stage**.

Block Name	Category	Color
clear	Drawing	dark green
forever	Control/Flow	dark orange
wait 1 secs	Control/Flow	dark orange
set font to normal32Helvetica	Drawing	dark green
set pen color to #000000	Drawing	dark green
draw text at x:0 y:0	Drawing	dark green
join hello world	Math π	green



```
on start
set score to 0
set hits to 0
clear
set font to normal 28 Helvetica
set pen color to 422977
forever
    draw text join Score: score   Hits: hits - + at x: 200 y: 300
    wait .5 secs
    clear
```

- This simple way to display the game score erases the information every half-second and then redraws it at the location X=200, Y=300.
- The *draw text* block joins together words and variables:
  "Score: " + *score* + " Hits: " + *hits*
  Hint: Leave spaces after the colons and before the word Hits.
- The *set font* block sets the font size to 32-point Helvetica.
- The *set pen color* block makes the font color dark blue, because #422977 means dark blue in a special code that computers use to represent a color by its red, green, and blue components (RGB). Other color codes are listed in the table that follows. Use these in the *set pen color* block to make your text to display in a different color.

***set pen color* number**	**Color**
#FFFFFF	white
#000000	black
#FF00FF	pink
#8000080	purple
#00FF00	green
#FF0000	red
#0000FF	blue

Part 5B: Test, redesign, and retest

- You now have a working game! Play the game to make sure that everything is working as it should.
    - The score and number of hits should display on the stage while the game is being played.
    - The score should increase by 1 whenever the kid bumps into the treasure.
    - The hits should increase by 1 whenever the ball bumps into the kid.
    - The ball should launch from the left side of the screen and move toward the kid. When it reaches the right side it should disappear and then start again on the left side.
    - The treasure chest should disappear and reappear in a new spot when the kid bumps into it.
    - The kid should move all around based on how the tablet is being tilted.
- Is there anything about the game that isn't quite right, or that you would like to change? Computer programmers redesign quite often. The process of building, testing, and then fixing and rebuilding is also called *iterating* on the **design**.
- Remember that this game's design criteria specified that there should be several balls. Your game is working but it only has one ball. **In our third lesson out next Friday, you will move onto Part 6 to learn how to use cloning to create more balls.**