PLTW | Launch

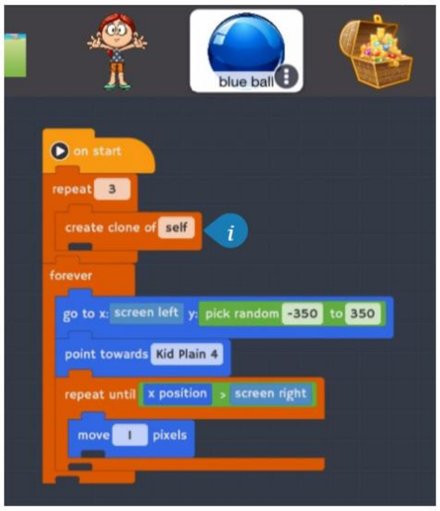
Activity 3 Building Computer Models

Part 6: Clone the ball to make more balls fly across the screen

30. Imagine if you wanted to make a simulation with 100 agents—it would take a really long time to program one actor for each agent! Instead, you can use cloning—a powerful computational technique that allows you to make copies of an actor.
    The new actors, called "clones," will have all the same rules and properties of the original actor when they start out. Once created, you can change the properties and variables of each clone as needed.
31. Go to the **ball's *on start* script** and add the blocks as shown below. Remember to change the values in the white boxes as shown in the image. If you leave the *repeat 10* block as it is your player will have to dodge a lot of balls!

Block Name	Category	Color
repeat 10	Control/Flow	dark orange
create clone of select actor	Control/Flow	dark orange

Notice that you have to put these blocks before the *forever* block. What do you think would happen if there were blocks after the end of the *forever* loop? Would they ever run?



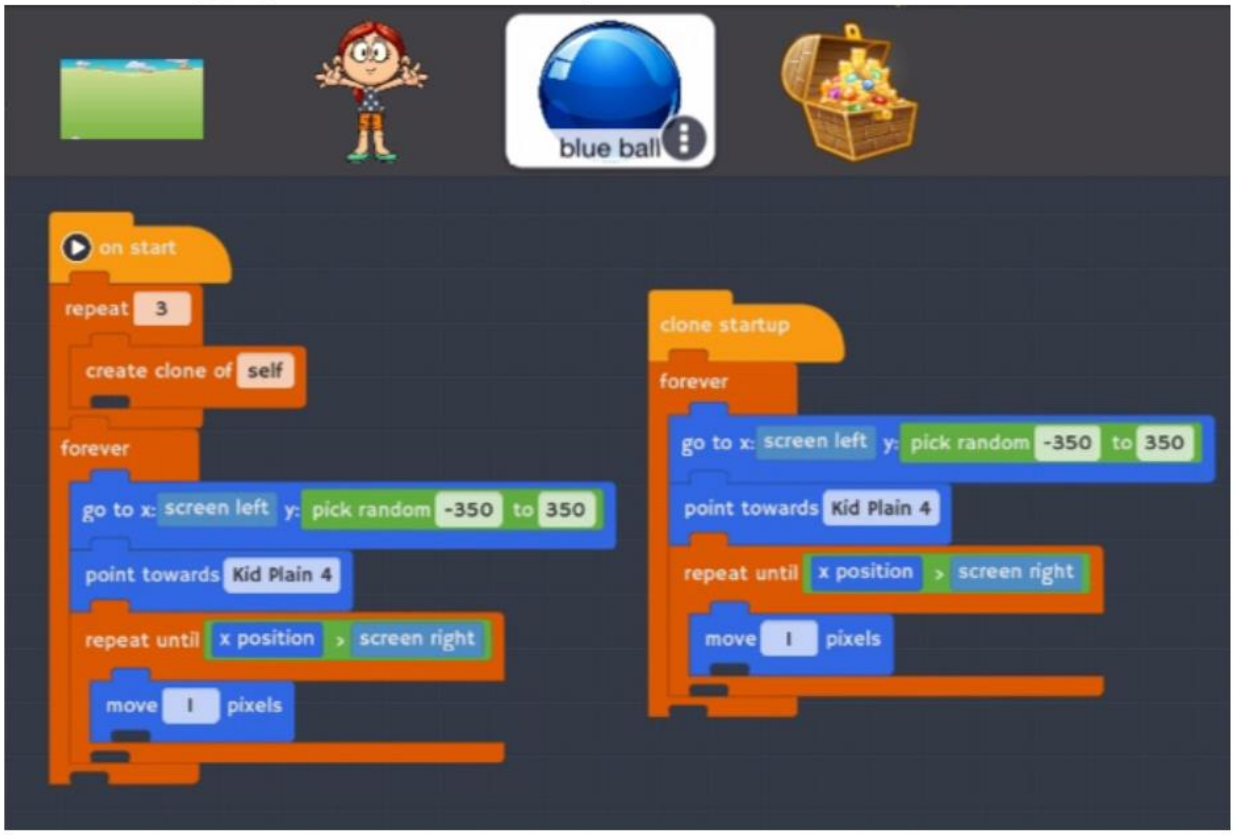
```
on start
repeat 3
    create clone of self
forever
    go to x: screen left y: pick random -350 to 350
    point towards Kid Plain 4
    repeat until x position > screen right
        move 1 pixels
```

32. Think about the number of copies of the ball the *repeat 3* loop will create. Click Play and see what happens.
33. Does it appear that there is only one new clone of the ball, and that it is not moving? Think about what might be happening.
34. When a clone starts up, it makes a complete copy of everything in the original actor's program. But there is one important difference:
    1. A clone runs the blocks under ***clone startup*** when it is created.

2. Clones <u>do not</u> run the blocks in the *on start* script.
3. Only the original actor runs the *on start* script.

This is a very important difference between an actor and its clones!

35. The reason that you cannot see the three clones of the ball is that they are stacked directly on top of each other and there is no code to make them move. The reason the clones are not moving is because they do not follow the blocks in the *on start* script.
36. Drag out a *clone startup* block from the Control category. Copy the movement blocks from the *on start* script to the *clone startup* script as shown in the following image. The **ball's code area** should look like the image that follows.
    1. **Remember that you can make a copy of the *forever* loop blocks by grabbing the *forever* block and dragging it onto the blue ball's picture in the Actors Panel.**



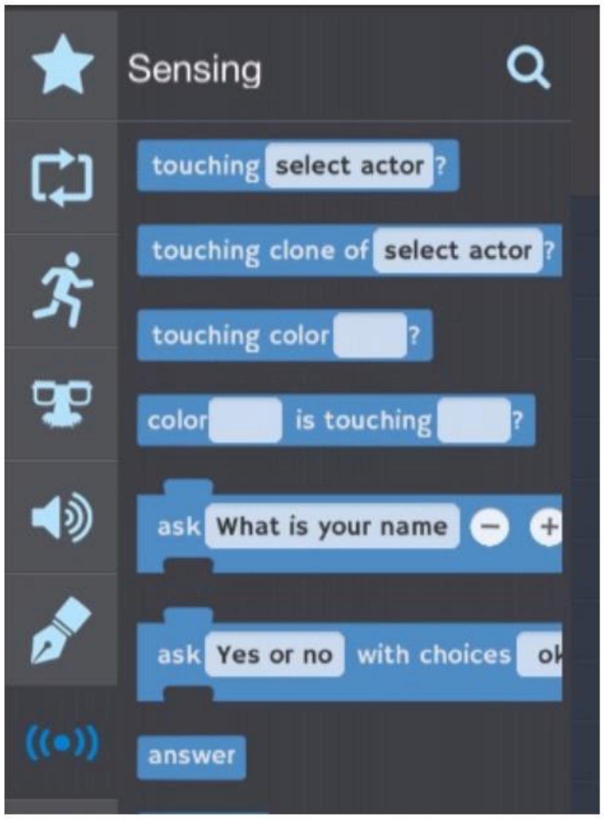
```
on start
repeat 3
    create clone of self
forever
    go to x: screen left y: pick random -350 to 350
    point towards Kid Plain 4
    repeat until x position > screen right
        move 1 pixels

clone startup
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
```

Use the same movement blocks in original ball actor script and cloned actor script.

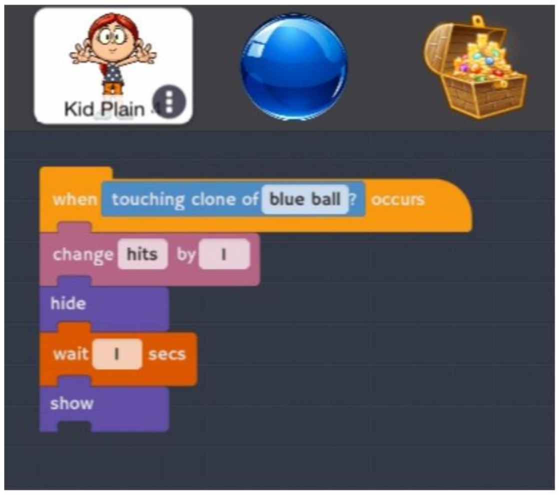
37. Click Play and test your game again. For the purpose of testing, let the kid get hit by the ball a few times and pay close attention to what happens. The kid is supposed to hide for a second and then reappear. But sometimes when the kid gets hit nothing happens. What has changed?

38. Time for some **debugging**! Go to the **code area** for the **kid actor** and have a look around at all of blocks. What is the event that triggers the kid to disappear?
39. Read the *when* block and think carefully about what it says: *when touching blue ball occurs.*
40. Look at the sensing category and notice that there are two different touching sensors. One detects *touching select actor*, and another detects *touching clone of select actor.* Which one do you think you need?



41. Look again at the blocks in the **kid's code area**. The *when touching blue ball* event is true only when the kid is touching the original blue ball actor, not when the kid is touching one of the clones.

- Change the kid's event to *when touching clone of blue ball*. This event will trigger when the kid is touching <u>either</u> the original blue ball <u>or</u> one of its clones. Your code should look like the picture that follows.



```
when touching clone of blue ball ? occurs
change hits by 1
hide
wait 1 secs
show
```

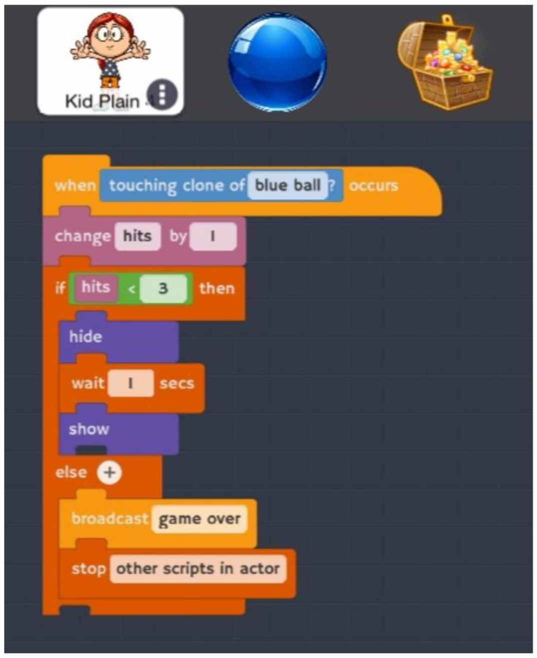
42. Try playing your game again. If everything is working right, go on to the next part.

- You may wonder why we didn't just create three additional actors instead of using cloning to create the balls for this game. For one thing, it is more efficient to program an actor once and then reproduce that actor even if it is just copied a few times. Additionally, when you build simulations in parts 4 and 5 of this module you will need to clone dozens of copies of one actor, so this activity is a good way for you to learn how to do that. Cloning is a very powerful programming technique used in modeling and simulation!

Part 7: Limit the length of the game

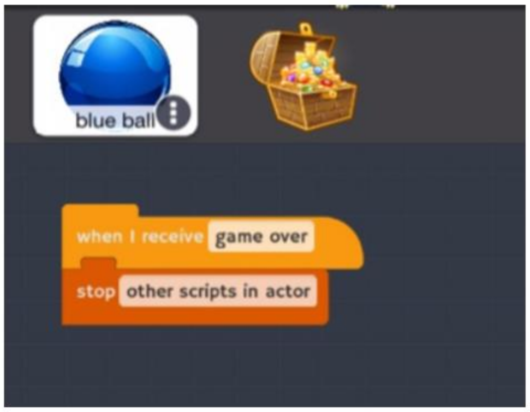
43. To set a limit on how long the game will last, limit the number of times the ball can hit the kid before the game is over. After three hits, end the game.

The blocks to the right show you how to count the hits and stop the game when the kid has been hit three times. This code takes advantage of the *broadcast* and *when I receive* blocks, which can be found in the Control/Flow category (orange). The *stop other scripts in actor* is also in Control/Flow. It says *stop all* before you change it to *other scripts in actor*. Actors communicate with each other through the ***broadcast*** and ***when I receive*** blocks. When the kid actor detects that the total number of hits=3 she broadcasts "game over" and stops her scripts. When the treasure and the ball (and all ball clones) receive the "game over" message they stop their scripts. This chain of events make all actors in the game stop moving.

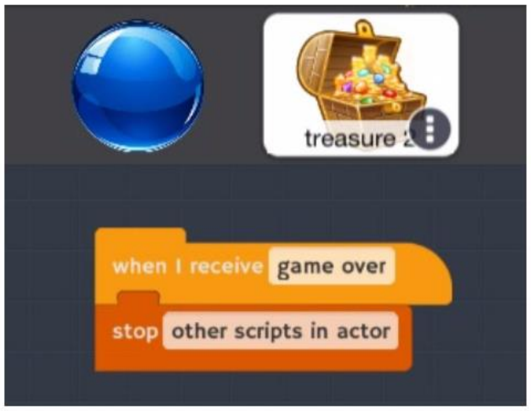


```
when touching clone of blue ball ? occurs
change hits by 1
if hits < 3 then
    hide
    wait 1 secs
    show
else
    broadcast game over
    stop other scripts in actor
```

a. Add an *if-then-else* block to the **kid's code**, as shown in the picture above.
   - Test: if hits < 3
     - Yes, hits <3: Hide, wait 1 sec, show
     - No, hits ≥3: broadcast game over, stop other scripts in actor



```
when I receive game over
stop other scripts in actor
```



```
when I receive game over
stop other scripts in actor
```

b. Add a *when I receive* block and *stop other scripts in actor* block to the blue ball's code and to the treasure chest's code.

44. Congratulations on building a working dodgeball game! Test your game extensively. Allow others to play with your game. People who have not helped develop the game often do things you were not expecting so they make great testers.
45. Is there anything about your program that you would like to change? If you have extra time, **experiment** with the code and see what happens. If you are going to make a lot of changes, it is a good idea to save a copy of your project before starting the changes. Backups are always good insurance!
46. Answer the Conclusion Questions at the bottom of this post.