P4 Game Scratch Notes

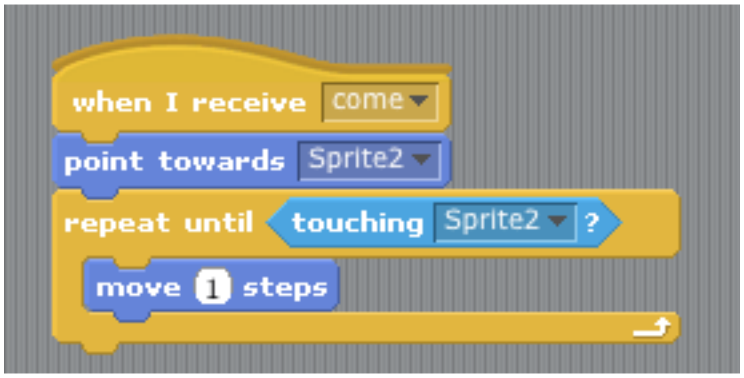
Chapter 5: Broadcast and Receive

Example Code:

- ex_comeHither.sb
- ex_scareBabies.sb

Often you will want one script to alert another script that something has happened and for that other script to then take an action. In Scratch this is done with Broadcast and Recieve.   For example, consider the example in **ex_comeHither.sb.** Lets assume I want to create two sprites, a boy and a girl. When I click on the girl she says "Come Hither" and then the both dutifully moves over to her. We can write the girls sprites script so that when she is clicked on, she says "Come Hither" and then broadcasts a message "come":

When a broadcast message is sent any sprite that wants to listen for it can. If we want the boy to move towards the girl the script needs to use the "When I receive" statement and then move towards the girl as so:



```
when I receive come
point towards Sprite2
repeat until touching Sprite2 ?
    move 1 steps
```

Note that multiple sprites can hear the message, it is **broadcast** for all to hear. If multiple sprites have a "when I receive come" block each of them will run the code when the message is broadcast.

ex_scareBabies.sb: In this example there are multiple sprite babies, each of whom waits for the message "Cassy is mad" to be broadcast.

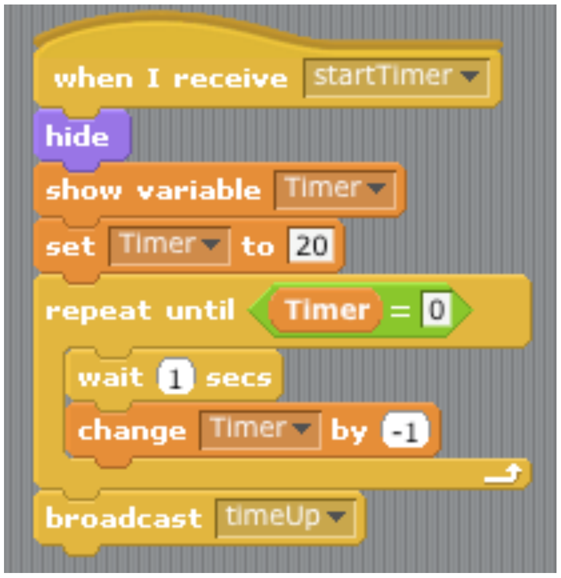
Chapter 6: Timer

Example Code:

- timer.sb

Often a game requires a timer, for example you have to get all the bad guys before the time runs out, or, you want to compete to see who can do a task the fastest. A timer can be created by using a global (for all sprites) variable named Timer along with a Timer sprite that controls the count down of the variable.

The following code will make the Timer count down to zero and then broadcast a message of "timeUp" when it gets to zero. Note, the "hide" command hides the sprite, whereas the show variable puts the variable on the screen to show the time counting down.



```
when I receive startTimer
hide
show variable Timer
set Timer to 20
repeat until Timer = 0
    wait 1 secs
    change Timer by -1
broadcast timeUp
```

Chapter 7: Simple Game

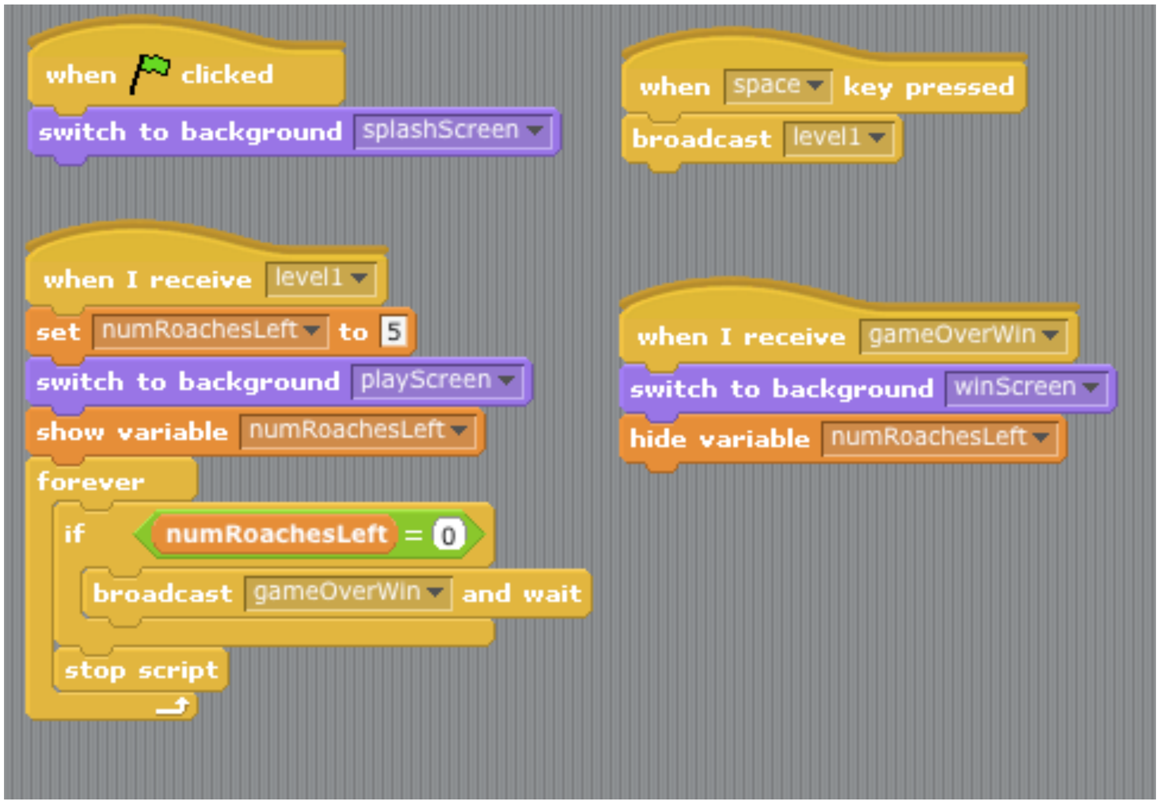
Example Code:

- ex_roachSquish_v1.sb
- ex_roachSquish_v2.sb

In this chapter we create a simple game. We propose the following simple model for a game. A game will have four game states, each with its own background: 1) Splash Screen state; 2) Level 1 state; 3) Win state; and 4) Lose state. A game starts out is the splash state with a Splash Screen. A splash screen is the starting screen and typically contains the instructions or any other info the player should have before starting. In our model the game will start when the player hits the space bar. Once the space bar is hit the game transitions to the next game state, i.e. Level 1. Different levels may have different backgrounds, number of sprites, speeds, etc. Our model is easily adapted to multiple levels. In this example we will have one level. There needs to be a win condition and a lose condition. When/if the player wins the game transitions to the Win State. When/if the player loses the game transitions to the Lose State.

There are many ways to transition between states in Scratch. Perhaps the easiest way, and equally important easiest way to understand, is to put the logic for detecting state changes in the Script for the Stage object. Further, when a transition is to be made a Broadcast message is sent to trigger the change.

Enough talk, lets make a game! Our game will be called "Roach Squish" and we will develop it in two steps. In the first step we will have a splash screen, level 1 screen, and a win screen. In this first version of the game there will be a win condition only. The code is in **ex_roachSquish_v1.sb.** (the v1 is short for "version 1"). Up until now all of our script code has gone in the sprites, now we will be adding code to the Stage object. Here is the code in the stage object for version 1:



```
when [flag] clicked
switch to background [splashScreen]

when [space] key pressed
broadcast [level1]

when I receive [level1]
set [numRoachesLeft] to 5
switch to background [playScreen]
show variable [numRoachesLeft]
forever
    if <(numRoachesLeft) = 0>
        broadcast [gameOverWin] and wait
    stop script

when I receive [gameOverWin]
switch to background [winScreen]
hide variable [numRoachesLeft]
```

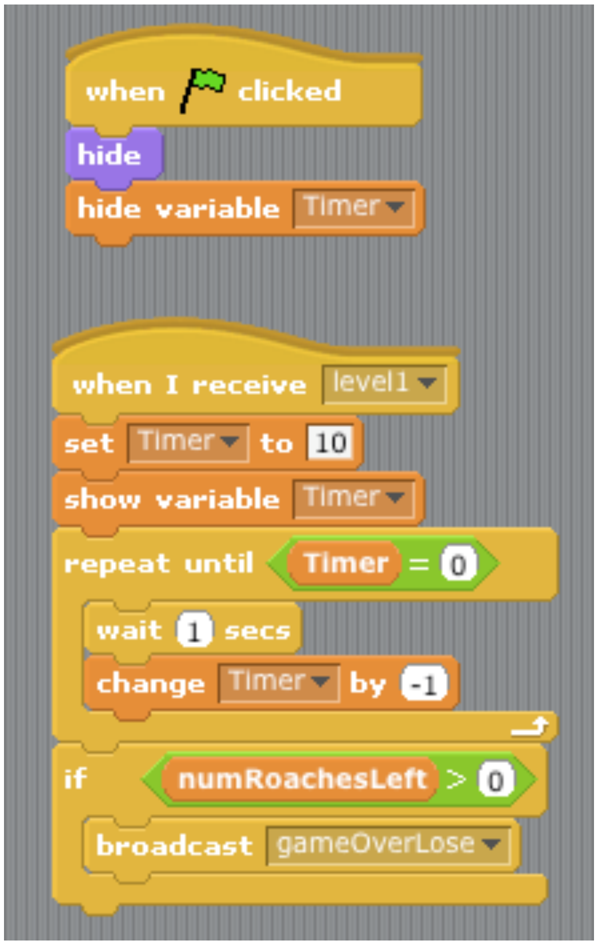
When the flag is clicked we set the background to “splashScreen”. Look at the Backgrounds tab to so we have three backgrounds for this game: splashScreen, playScreen, and winScreen. The splashScreen background contains instructions, it should probably have some art also to make it look more like a game. The instructions on the splashScreen say to click on the space bar to begin.

In the script, we see that when the space key is pressed a “broadcast level1” is done. In the same Stage script, we see on the bottom right that when the stage script receives the level1 message, it sets up level1: sets the numRoachesLeft to 5, switchs to the playScreen, shows the variable numRoachesLeft and then forever checks to see if numRoacheLeft = 0. This is our WIN CONDITION: numRoachesLeft = 0 . When this win condition becomes true the message “gameOverWin” is broadcast. As can be seen in the bottom right script, when the message “gameOverWin” is received, the background is switched to the winScreen, and, the numRoachesLeft variable is hidden so the winScreen looks pretty.

At first it might seem a bit odd to use broadcasts to drive the game state transitions. Could we not just do it with one script? Probably, but as games get more complex the code becomes very hard to understand, hard to modify, and hard to get right. This broadcast/receive paradigm for driving state transitions makes it easy to modify the game for more levels and more complicated conditions.

Now lets modify the game to have a win and lose condition. Lets add a timer. If the timer gets to zero the player loses. If the player gets all the roaches before the time gets to zero the player wins. You can find code for this in **ex_roachSquish_v2.sb.**

The code for the timer class is:



```
when [flag] clicked
hide
hide variable Timer

when I receive level1
set Timer to 10
show variable Timer
repeat until Timer = 0
    wait 1 secs
    change Timer by -1
if numRoachesLeft > 0
    broadcast gameOverLose
```

The timer code counts down by one unit every second. It does this until it gets to zero. Once the timer gets to zero the user has lost so we want to broadcast a "gameOverLose" message, but, first we check there are still some roaches left. The reason for this check is that if all the roaches are squished before the timer runs out, the timer is still running, thus, without this check it would see win, and then change to lose when the timer runs out.