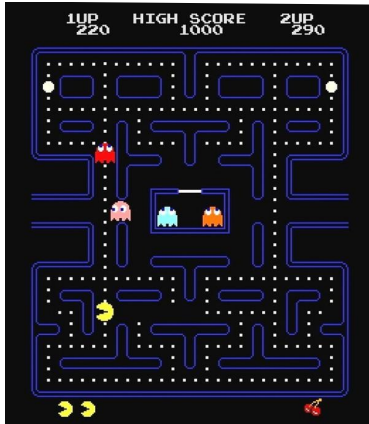
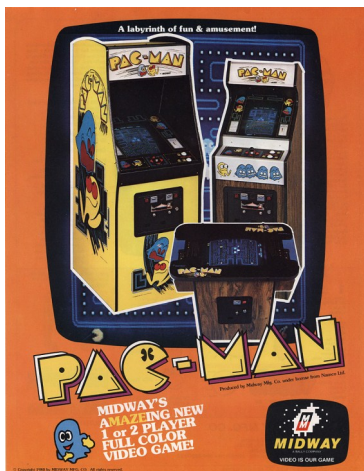


# How to Build a Simple Pac-Man Game

For today's program, we are going to build a simple Pac-Man game. Pac-Man was one of the very first arcade games developed around 1980. For our version of Pac-Man we are going to focus on the following programming and problem solving techniques:



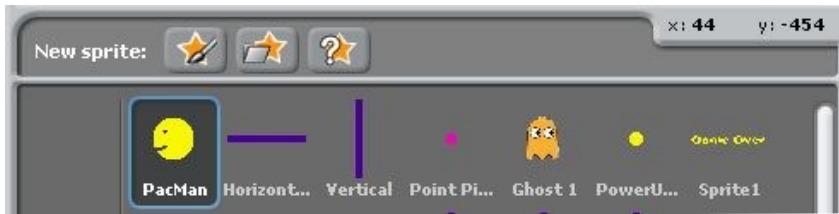
a. Using Scratch to develop computer objects (also known as Object Oriented Programming classes) that will interact with each other in the game. We will only develop five sprites with scripts for this program (7 in total for the game) but will be using copies of the sprites to make a complete fully interactive video game (inheritance principle). Object Oriented Programming is a complex topic and this program will not use design patterns which are a hallmark of OOP. Your next project, the Marble Roll Game, will start to explore this area however.



b. Developing simple animation of characters in the game. Animation code (blocks) will be kept separate from functionality code (blocks) so that sprites can be reused with simple changes to the costumes and minor changes to some variables. This is how computer languages and scripts that are fully object oriented (such as Action Script 3 for Flash or Java) work.

*This lesson is designed for intermediate to advanced users of Scratch who have had some experience in working with the software and developing Scratch applications. Basic concepts such as how to create new sprites, where to write scripts, etc. will not be reviewed. The software in this series of lessons only covers creating a single level for this version of Pac-Man, however, the software has been written so that multiple levels are easy to add with simple changes to variables. We will discuss how to do this briefly at the end of the lesson.*

## Getting Started



To get started creating our Pac-Man program, we are going to build a total of seven sprites. Start by creating a sprite named **Pac-Man** with only a single costume.

Then create a sprite called **Horizontal** to be a horizontal bar and **Vertical** which will be a vertical bar. The sprites will be used to build the maze for Pac-Man and should both be the same color. Next we will create a sprite called **Point Pill** that Pac-Man will eat as he goes through the maze. We will then create a sprite called **Ghost 1** with a single costume. This ghost will be Pac-Man's enemy and will be copied to make further ghosts later in the game. Lastly, we will create a sprite called **PowerUp** in any color that is different from the Point Pill sprite. When Pac-Man eats a PowerUp, he will have the ability to eat the ghosts for about 10 seconds and score 100 points for every ghost he eats. Our last sprite will be called **GameOver** and will display a game over message on screen once three Pac-Man characters have been eaten by the ghosts and the game is finished.



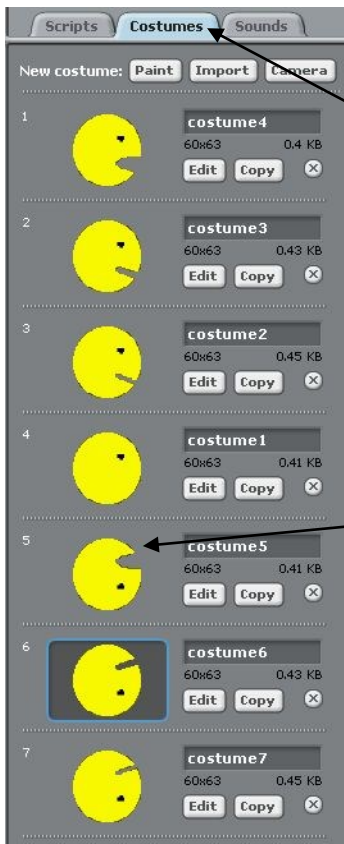
Next we will create a total of eight *variables* for our game. Our first variable will be called *Chase Pac-Man*. This variable will be used to determine if Pac-Man can eat the ghost or if the ghost can eat Pac-Man. Our next variable will be called *Direction*. The Direction variable will be used to randomly determine which directions the ghosts travel on the screen (up, down, left, or right). Each ghost needs to have its own direction variable. In our example we will create a total of four ghosts. Thus *Direction Ghost 2*, *Direction Ghost 3*, and *Direction Ghost 4* are used so that all ghosts can travel separate directions from each other. The next variable we will create is called *Ghost Speed*. Ghost speed will be used to determine how fast the ghost will travel and can be increased with each level of the game to make the game more challenging. Our next variable will be called *Lives*. Lives will be used by Pac-Man to

determine when the game is over once he has been eaten three times by the ghosts. Our last variable is called *score* and will be *checked* so that it appears on the screen. This is used to keep track of the score in the game. With our initial sprites and variables created let's turn to making our first character the Pac-Man.

## PacMan Set-up

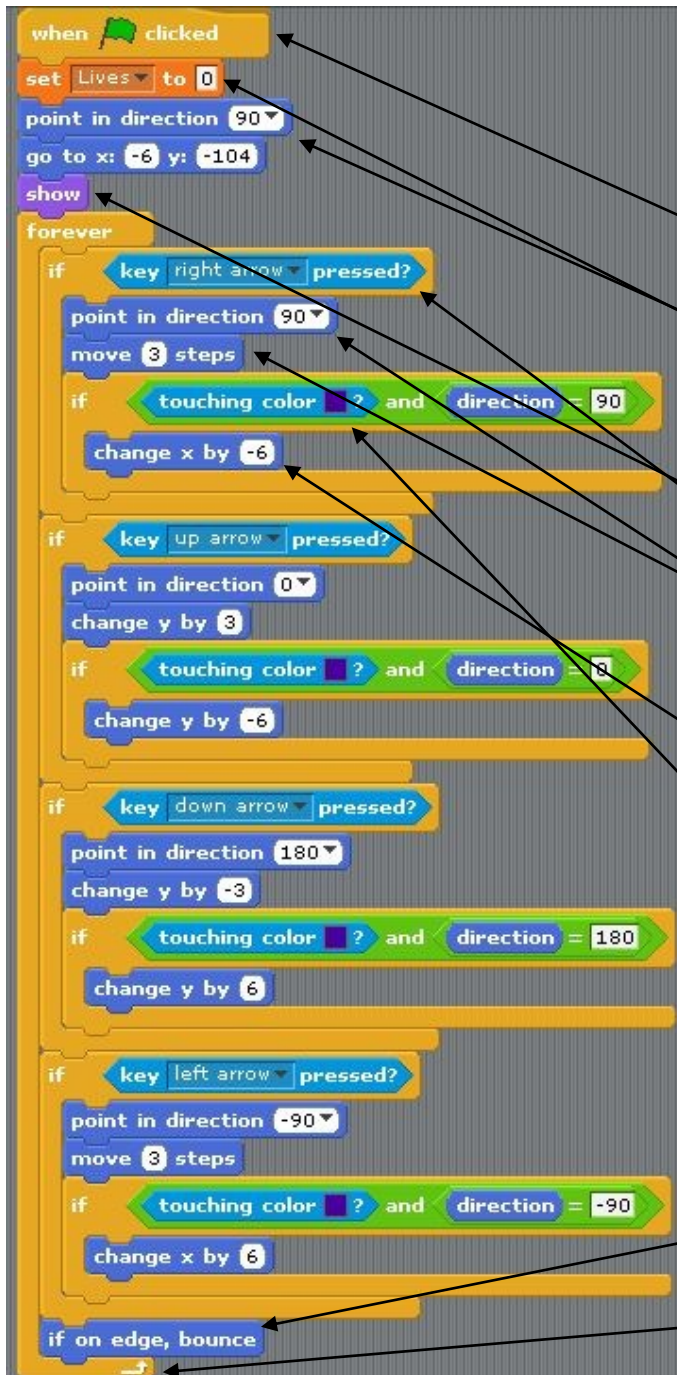


Let's start by setting our initial condition for Pac-Man. The first thing we need to decide is how our character will move on the screen. To simplify the animation of Pac-Man the first thing I'm going to do is select the *can rotate* button to control his direction on the screen. This gives us the ability to easily allow Pac-



Man to turn and move in four directions. The disadvantage to this selection is that Pac-Man's character will appear upside down when he is going left on the screen as selecting this flips the character in the opposite direction. To get around this we will use some creative programming of our costumes when Pac-Man is facing this direction. This will be explained later in the directions.

Next we will create a total of seven different costumes. One of the costumes must have Pac-Man's mouth closed as this will be the default costume. The next three costumes will show his mouth in various states of being open. By flipping through these costumes we will add animation to Pac-Man and make him appear to eat. For costumes numbers 5 to 7 we will *flip him upside down*, again, with his mouth being open in various states. When our default costume is combined with these upside down costumes and Pac-Man is facing to the left, he will be rotated around (rather than being upside down) and appear to be eating while going in the left direction. This will be further explained when we look at the animation blocks. Feel free to add more or less costumes to your Pac-Man, however, you will need to adjust the number of blocks when we get to the animation scripts.



## PacMan Main Script

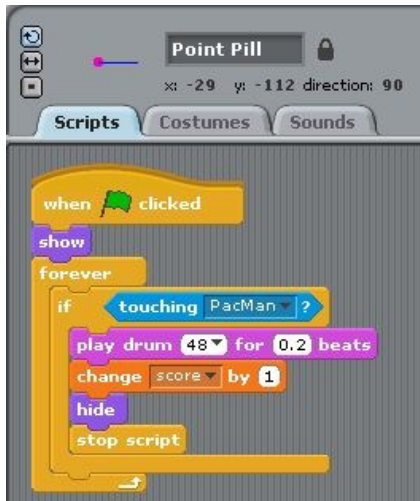
The main Pac-Man script is used to set Pac-Man up in his initial location and allow him to move around the screen. It also determines what happens when Pac-Man hits the wall the maze or the edge of the screen. When the **green arrow** is clicked the variable **Lives are set to zero** (meaning he now has three lives) he is **rotated to 90°** (his) starting direction, moved to his **initial spot on the screen** and is **shown**. The Pac-Man script then enters a **forever loop** that will check to see if keys are being pressed that would allow him to move.

First, it checks to see **if the right arrow key is being pressed**. If it is, it **rotates Pac-Man around facing the right direction (90°)** and **moves him 3 steps to the right**. If however, he bumps into a maze (or he is **touching a maze color**) it pushes him away from the maze wall in the **opposite direction** so we cannot move through the maze. The rest of the **if statements** in this loop do similar things checking to see if Pac-Man should move to the left, up, or down and moving in that direction unless he hits the maze wall in which case he is stopped. At the bottom of the loop **if he is touching the edge** he simply bounces back onto the screen. By controlling his movement through the use of a **forever block**, the Pac-Man character is much more responsive to key presses than he would be through the

use of a control block to move him.



him to the *start costume* for the program, places him back in the *starting position, points him in the correct direction* and then *shows* him on the screen and continues the game.



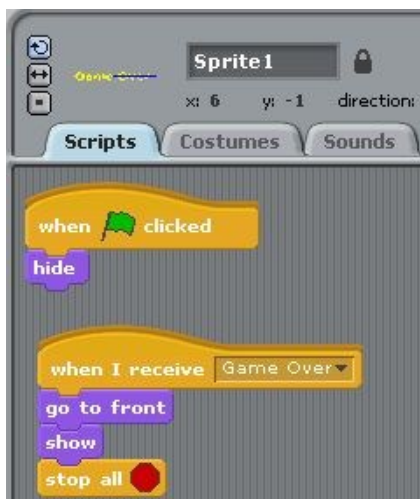
### Point Pill Script

The point pill script is used to have pills the Pac-Man can eat and score points on as he goes around the screen. When the *green flag is clicked* the pill is *shown* on the screen. It then enters a *forever loop* which checks to see *if the pill is touching the Pac-Man*. If it is, it *plays a sound, changes the score by one* (adding point to the Pac-Man score) *hides* the pill and then *stops all scripts*. The stopping of the scripts speeds up other animations in the software. If you're creating multilevel game you may just wish to hide the point pill and re-show it at a new level. This will be discussed further at the end of this tutorial.



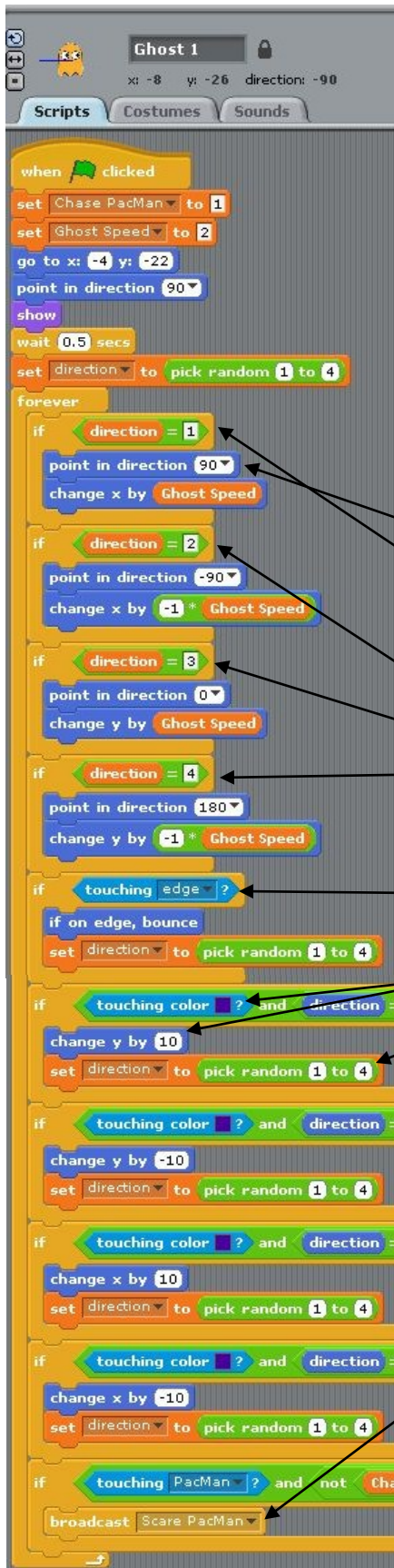
### Power Up Script

The power up script allows the Pac-Man to eat the ghosts for a limited amount of time once he has eaten the power up pill. When the *green flag is clicked* the power up pill is *shown* on the screen and enters a *forever loop*. It checks to see *if the pill is touching the Pac-Man*. If it is, it *broadcasts a power up message* (that is received by the ghosts) *hides* the pill that has been eaten by the Pac-Man and *stops the pills script* speeding up the software. Again if you're creating a multilevel game you may just wish to hide the PowerUp pill and re-show it at a new level. This will be discussed further at the end of this tutorial.



### Game Over Script

The game over script is simply a way to indicate to the user that they have lost the game. Create a sprite that simply says "Game Over". When the *green flag is clicked* *hide this sprite*. In the Pac-Man sprite we had it broadcast a message called "Game Over" when it lost more than three lives. That message is *received by the Game Over sprite*. When it receives a game over it brings the game over sprite *to the front, shows it*, and then *stops all scripts* in the program. This ends the game and the user must re-click the green flag in order to start the game again.



## Main Ghost Script

The main group ghost script controls the movement and interactivity of the ghosts in the program.

When the *green flag is clicked* a *variable called Chase Pac-Man is set to 1* (meaning the ghosts are able to eat Pac-Man) and the *ghost speed variable is set to 2*. The ghost speed variable determines how quickly the ghosts move and can be increased (such as in different levels) to make the game more difficult. The ghost is then set to its initial position point in the correct direction and shown on the screen. A *random number between one and four* is then selected and *given to the direction variable*. This random number will determine the direction the ghost travels until it hits a wall and as such, each ghost will need its own direction variable (which we set up earlier) so they can operate in different directions from each other.

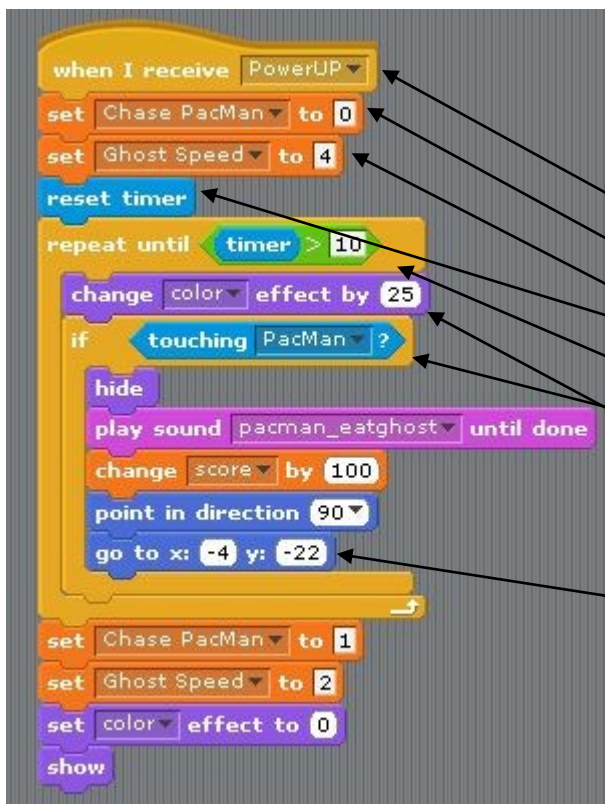
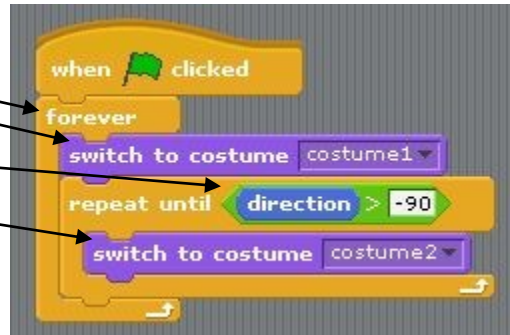
The program then enters a *forever loop* and checks to see which direction the ghost was set in. *If the direction variable is 1* the ghost goes right, 2 the ghost goes left, 3 the ghost goes up, or 4 the ghost goes down. This is controlled by a *bunch of if statements* that checks to see what the direction variable is set to *points the ghost in that direction* and then changes its *X or Y direction* by the *ghost speed variable* which determines the speed of the ghost. If the ghost hits either the *edge of the stage or part of the wall* it is told to *bounce away from the object* and then *pick a new*

*random number*. When it goes back up to the loop it will have a new direction which it will follow until it again hits the maze or the edge. At the bottom of the main loop there is an *if statement that checks to see if it's touching the Pac-Man and its Chase Pac-Man variable is not = 0* (meaning the Pac-Man is not powered up to eat the ghosts). If this is the case the ghost *broadcasts a "Scare PacMan"* message which is received by the Pac-Man character costing the Pac-Man a life in the game.



### Ghost Costumes & Rotation

We will use the same *rotation* and costume tricks for the ghosts as we did for the Pac-Man to aid in animation and travel direction. Start by creating two ghost costumes, one that is upright and one that is *upside down*. Then selected the can *rotate button up at the top of the screen*. Now we'll create a costume script for the ghost. When the *green flag is clicked* enter *forever loop* that will *switch to costume number one* until the *direction of the ghost is greater than -90°*. When this happens the *upside down costume (2)* will be selected, rotated around to be the correct direction, and repeat like this until the direction of the ghost is greater than 90°.



### Ghost Power Up Script

The ghosts power up script allows the Pac-Man to eat the ghost when this script is running. When the Pac-Man eats a Power Up Pill it sends a message called *PowerUP that is received by the ghost*. The ghost sets the *Chase Pac-Man to zero* (meaning that the ghost can't eat Pac-Man), sets the *Ghost Speed to 4* (meaning the ghost runs a little faster) and *set the timer in the game to zero*. The ghost then enters *a repeat loop that continues until the timer is greater than 10*, meaning 10 seconds have gone by. The ghost then has a *color change* meaning it flashes while the Pac-Man is able to eat it and checks to see *if the Pac-Man is touching the ghost*. If it is, it *hides the ghost plays a sound to show it's been eaten, adds 100 points the Pac-Man score points it in the starting direction* and *moves the ghost back to its starting base in again*. At the end of 10 seconds the repeat loop ends, *sets the Chase Pac-Man back to one* (mean the ghost can now eat the Pac-Man)

*sets the ghost speed back to normal, stops the flashing of the ghost* and make sure *it shows it on the screen again*. If the Pac-Man had eaten the ghost it will reappear at this point, if it had not, it will simply stop flashing and return to its regular color.



### **Finishing the Games First Level**

To finish the first level on the game start by using the horizontal and vertical line sprites to create a maze. Simply make as many copies of these pieces as you need in order to create a maze. Have your Pac-Man run through the completed maze before you add ghosts point pills and power ups. Make sure the Pac-Man fits in all the tunnels and corners that you create. You can also draw out a maze on the main stage making sure that the colors that your ghosts and Pac-Man are looking for are correct so they can bounce off the maze.

Once you have your maze set up, you can duplicate as many point pills and power ups as you wish to fill up your maze. If you plan on building a multilevel game you may wish to place a variable that will allow you to show and hide the point pills and power ups for each level. These can then be turned on and off as your Pac-Man progresses through levels.

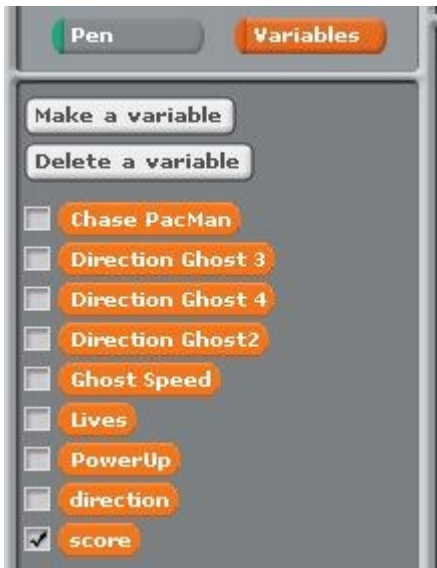
Lastly, you want to copy your ghost three more times and give it a unique name. For each of these ghosts you'll then need to replace the "direction" variable in the main script with a unique one for each ghost. We created these at the start and call them direction ghost 2, etc. This will allow all the ghosts to move independently in different directions. You should then change the costumes of each ghost giving it a unique appearance. The nice thing about building your sprites so they are reusable, is you are now able to create four unique enemies with very little extra work. Now try your game out !!

### **Hints on Adding Multiple Levels**

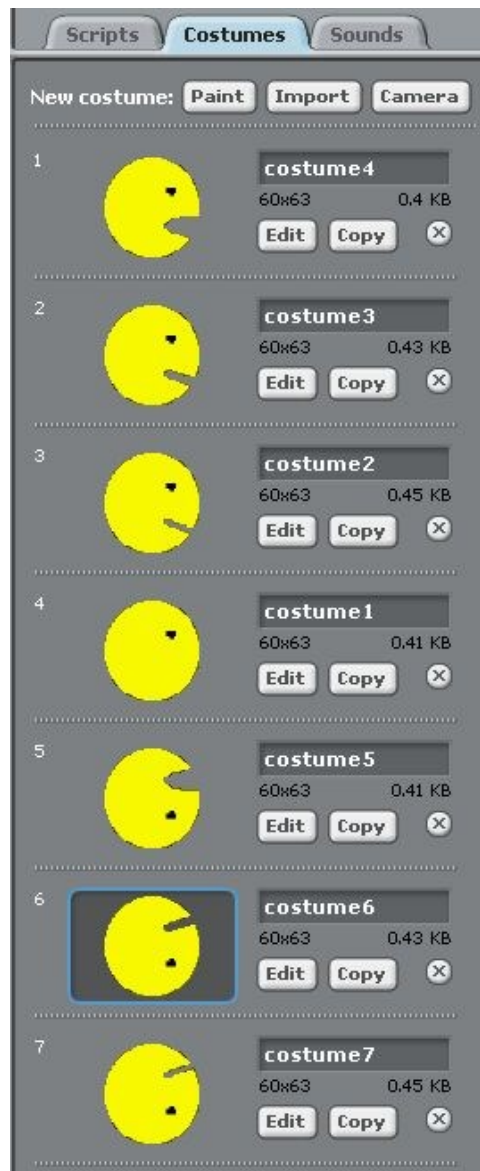
You should be able to add multiple levels to this game with minimal extra work. There are several things you will need to do with the current program to make this happen. First, you'll need to determine a way to tell when all the pills have been eaten and then reset them on the screen to their original positions. This can be done by creating a variable to keep track of this for each level in the game and then resetting it for each new level. Second, the ghost speed at each level can be increased making the game more difficult as the levels progress. Third, bonuses like fruit and other items could be added to make gameplay more interesting. I look forward to seeing which items you can add to make this game more interesting and more challenging through multiple levels !!

# PacMan Game Complete Sprite & Scripts List

## Set-up Sprites & Variables



## PacMan Costumes & SetUp

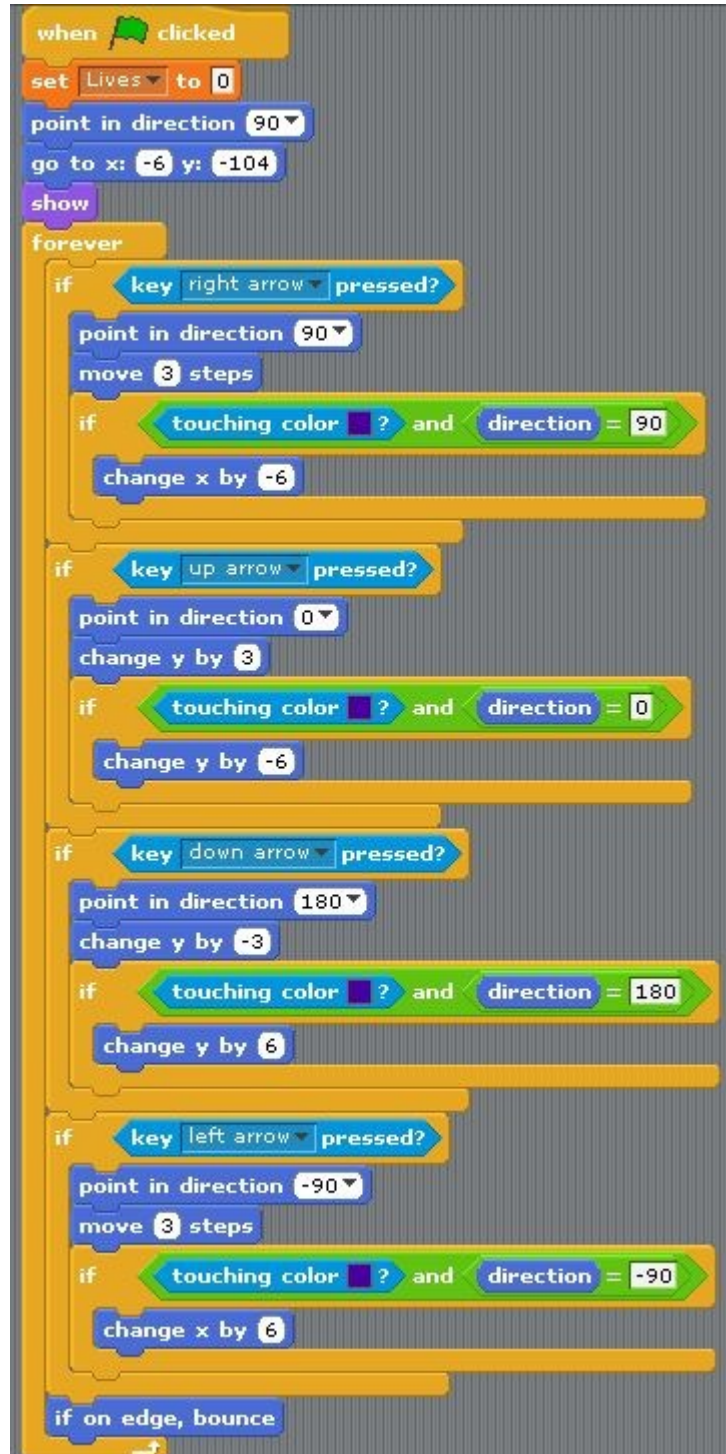


## PacMan Eaten Script



```
when I receive [Score PacMan]
hide
play sound [pacman_death] until done
if Lives > 3 or Lives = 3
broadcast [Game Over]
wait 2 secs
change Lives by 1
switch to costume [costume4]
go to x: -6 y: -104
point in direction 90
show
```

## PacMan Main Script



```
when clicked
set Lives to 0
point in direction 90
go to x: -6 y: -104
show
forever
if key [right arrow] pressed?
point in direction 90
move 3 steps
if touching color [ ] and direction = 90
change x by -6
if key [up arrow] pressed?
point in direction 0
change y by 3
if touching color [ ] and direction = 0
change y by -6
if key [down arrow] pressed?
point in direction 180
change y by -3
if touching color [ ] and direction = 180
change y by 6
if key [left arrow] pressed?
point in direction -90
move 3 steps
if touching color [ ] and direction = -90
change x by 6
if on edge, bounce
```

## PacMan Animation Script



```
when clicked
forever
switch to costume [costume4]
wait 0.1 secs
next costume
wait 0.1 secs
next costume
wait 0.1 secs
next costume
wait 0.1 secs
next costume
wait 0.1 secs
switch to costume [costume1]
repeat until direction > -90
switch to costume [costume5]
wait 0.1 secs
next costume
wait 0.1 secs
next costume
wait 0.1 secs
next costume
wait 0.1 secs
switch to costume [costume1]
```

### Game Over Script



### PowerUp Script



### Pont Pill Script



### Costumes



Ghost 1  
x: -8 y: -26 direction: -90

Scripts | Costumes | Sounds

```

when green flag clicked
  set Chase PacMan to 1
  set Ghost Speed to 2
  go to x: -4 y: -22
  point in direction 90
  show
  wait 0.5 secs
  set direction to pick random 1 to 4
  forever
    if direction = 1
      point in direction 90
      change x by Ghost Speed
    if direction = 2
      point in direction -90
      change x by -1 * Ghost Speed
    if direction = 3
      point in direction 0
      change y by Ghost Speed
    if direction = 4
      point in direction 180
      change y by -1 * Ghost Speed
    if touching edge
      if on edge, bounce
      set direction to pick random 1 to 4
    if touching color and direction = 180
      change y by 10
      set direction to pick random 1 to 4
    if touching color and direction = 0
      change y by -10
      set direction to pick random 1 to 4
    if touching color and direction = -90
      change x by 10
      set direction to pick random 1 to 4
    if touching color and direction = 90
      change x by -10
      set direction to pick random 1 to 4
    if touching PacMan and not Chase PacMan = 0
      broadcast Score PacMan
  
```

Ghost Main Script

Ghost PowerUp Script

```

when I receive PowerUP
  set Chase PacMan to 0
  set Ghost Speed to 4
  reset timer
  repeat until timer > 10
    change color effect by 25
    if touching PacMan
      hide
      play sound pacman_eatghost until done
      change score by 100
      point in direction 90
      go to x: -4 y: -22
  set Chase PacMan to 1
  set Ghost Speed to 2
  set color effect to 0
  show
  
```

Ghost Costume Script

```

when green flag clicked
  forever
    switch to costume costume1
    repeat until direction > -90
      switch to costume costume2
  
```