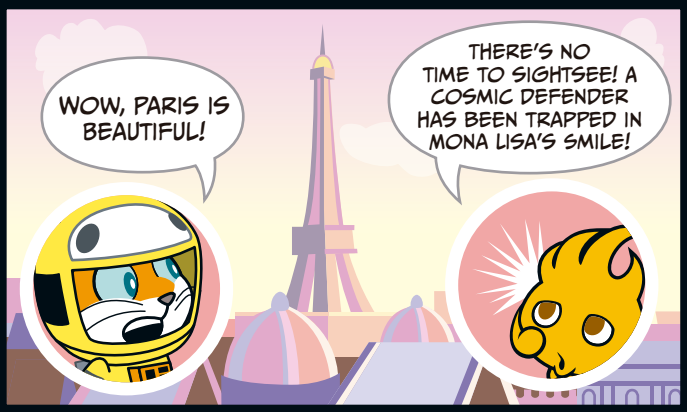
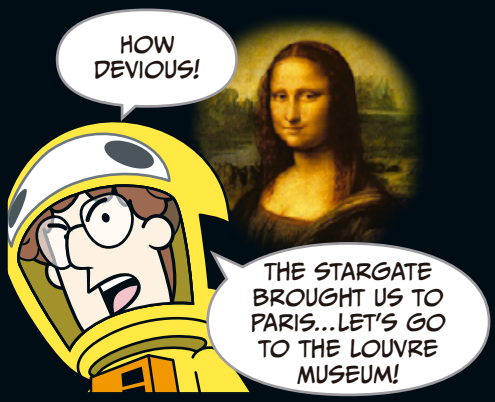
TRAPPED BY MONA LISA'S SMILE

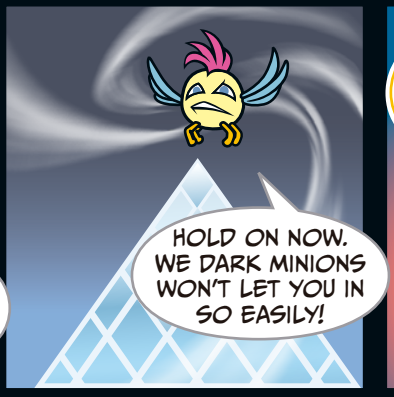
STAGE 3

STAGE 3


WOW, PARIS IS BEAUTIFUL!
THERE'S NO TIME TO SIGHTSEE! A COSMIC DEFENDER HAS BEEN TRAPPED IN MONA LISA'S SMILE!


HOW DEVIOUS!
THE STARGATE BROUGHT US TO PARIS...LET'S GO TO THE LOUVRE MUSEUM!


THE PAINTING'S IN THERE!
PUFF PUFF I SHOULD'VE TAKEN OFF THIS SPACE SUIT BEFORE RUNNING.


HOLD ON NOW. WE DARK MINIONS WON'T LET YOU IN SO EASILY!


RATA! BETRAYER! I CAN'T BELIEVE YOU'RE HELPING THE DARK WIZARD NOW!


ALL THE ART IN THE WORLD IS MINE NOW! *HEE HEE*


ART IS MEANT FOR SHARING. YOU CAN'T TAKE IT ALL FOR YOURSELF!


HUMPH! NOW THE PAINTING HOLDING YOUR FRIEND HAS BEEN CUT TO BITS! TAKE ON MY CHALLENGE AND THEN WE'LL TALK!

THE LOUVRE

STAGE 3

Chapter Focus

Let's learn how to control the *flow* of a game. You'll see how to keep score using *variables* and control the order of the game using *broadcasts*.

The Game

This game is actually two games in one. First, you'll face Rata's quiz. Then you'll have to put the *Mona Lisa* back together in a puzzle game. If you get the answer wrong three times, the game ends and you lose!

First, let's import a picture of the Louvre Museum as a background to the Stage.

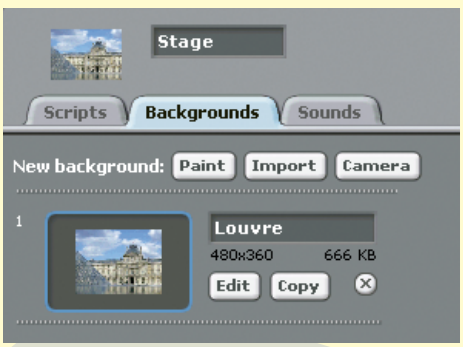


Photo Credit: Raphael Frey

Then we'll add a program that makes the Stage play music. The forever block is a special kind of command we call a *loop*. Any sound effect or music you add here keeps playing again and again, so make sure you like how it sounds!

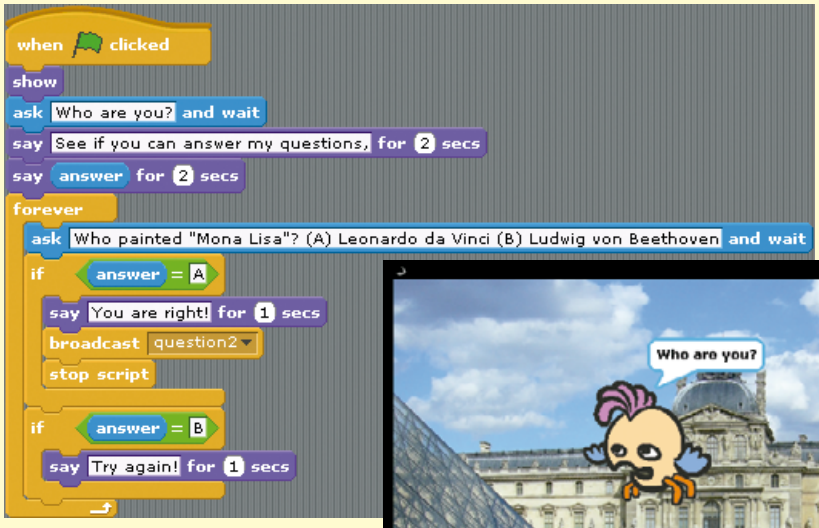
Now we'll add a new sprite ( ) for Rata. He's called *fantasy4* in Scratch's *Fantasy* folder.

1

Write program 1 first. This forever loop makes Rata float up and down.

For program 2, go to the **Sensing** and **Looks** palettes, and use the ask and say blocks. This program asks the first question of Rata's quiz. We've made it a multiple-choice question, so the answer must be *A* or *B*.

2

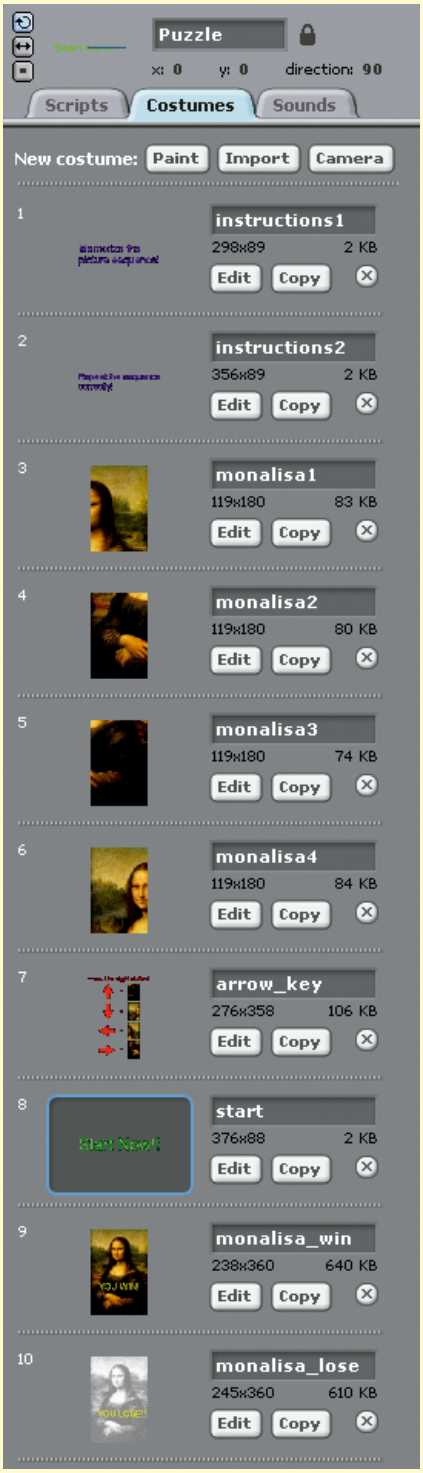


If you noticed back in program 2, there's a command that says `broadcast question2` if you get the right answer. *Broadcasts* are like big announcements to all the programs in your project. They're a great way to connect related parts of a game. So let's try writing two more questions as new programs 3 and 4. These two programs wait for broadcasts `question2` and `question3` to start using the `when I receive` block.

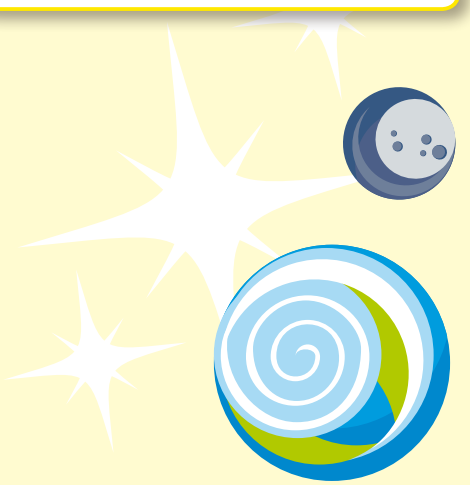
3
```
when I receive question2
forever
  ask Where was it painted? (A) Madrid, Spain (B) Florence, Italy and wait
  if answer = A
    say Try again! for 1 secs
  if answer = B
    say You are right! for 1 secs
    broadcast question3
    stop script
```

4
```
when I receive question3
forever
  ask Where is it now? (A) The Louvre, Paris (B) The Colosseum, Rome and wait
  if answer = A
    say You are right! for 1 secs
    say Now try to solve this puzzle! for 2 secs
    hide
    broadcast puzzle
    stop script
  if answer = B
    say Try again! for 1 secs
```

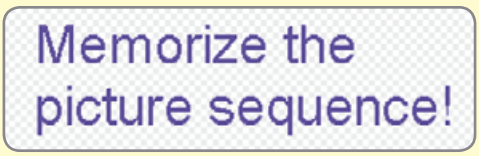
When the player answers all three questions correctly, the `puzzle` broadcast signal in program 4 tells the game that the quiz is over and the puzzle half of our game should now begin.



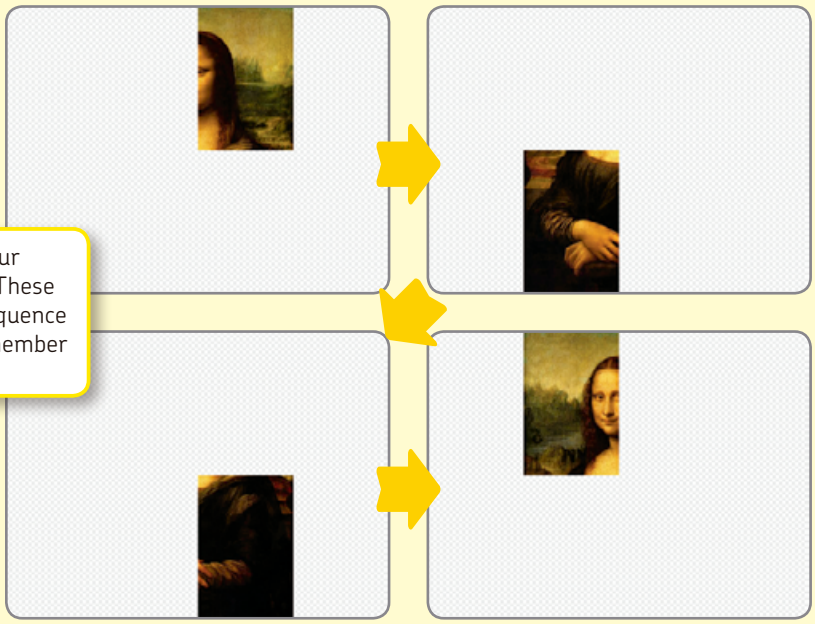
Import the sprite *Puzzle* from the *Super Scratch* folder ( ). This isn't just a single image—it's a sprite with a bunch of different costumes. The sprite's costumes include the instructions for the puzzle game and the puzzle itself. The final two costumes in this sprite display the winning screen and the message that appears when you lose.



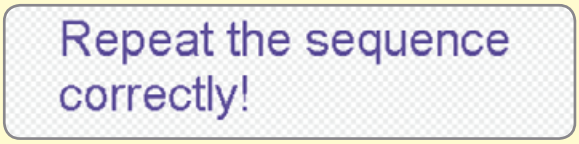
Let's take a closer look. First, we'll display the first instruction costume.

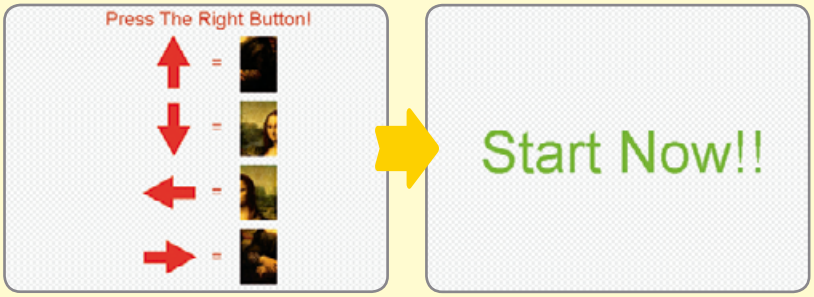


Then we'll display the four parts of the *Mona Lisa*. These costumes display the sequence that players have to remember to win!



The next three costumes display the game instructions and a start screen.



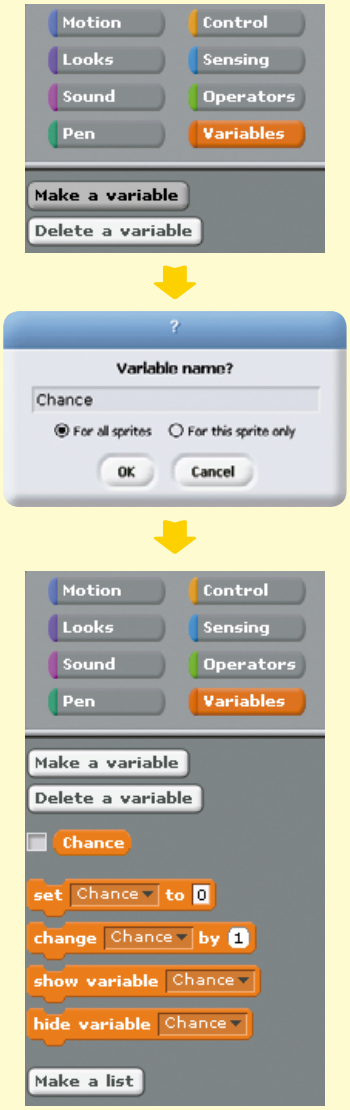


Finally, we have two costumes for the winning and losing screens.

STAGE 3

For this big sprite, we'll need a lot of programs. Let's start by adding a special kind of command called a *variable*. Variables are good for keeping track of numbers that change during a game, like scores, player health, player lives, and more.

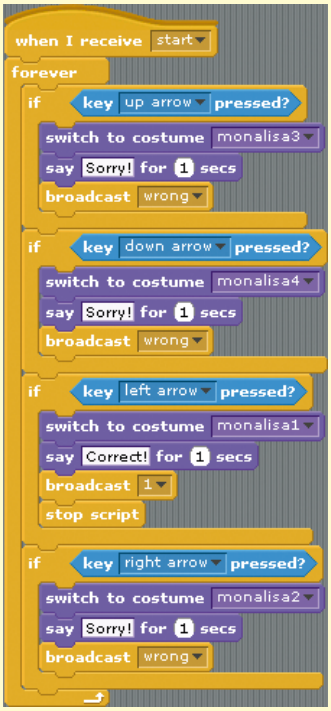
Click **Make a variable** in the **Variables** palette, and call it Chance. The new Chance variable is how the computer knows how many times the player gets another chance to solve the puzzle before losing.



Now for the programs themselves. Add scripts 1 and 2. Script 1 just hides our variable Chance during the quiz part of the game. Next, script 2 determines how the Puzzle sprite should change costumes—just as described on pages 48–49.

Then we'll add four different scripts: one for each right answer to the puzzle. If the player presses the wrong arrow, the sprite changes its costume and a broadcast called `wrong` is broadcast. We'll use this broadcast to control the Chance variable.

3

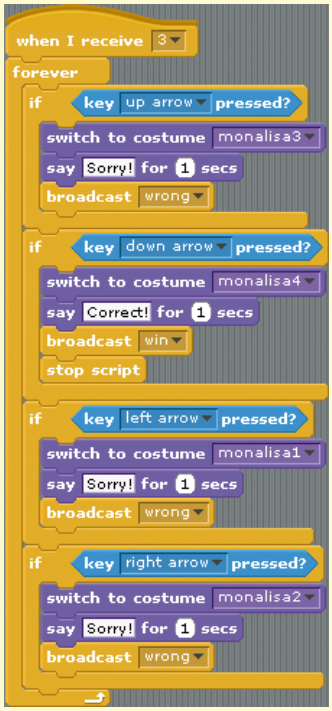


4

Notice how the broadcast named `1` at the end of script 3 starts script 4. Likewise, script 5 starts only `when I receive 3`, which is broadcast by script 4 when the player presses the correct arrow. With all of the correct arrows pressed in script 6, we signal a new broadcast called `win`.

5

6



STAGE 3

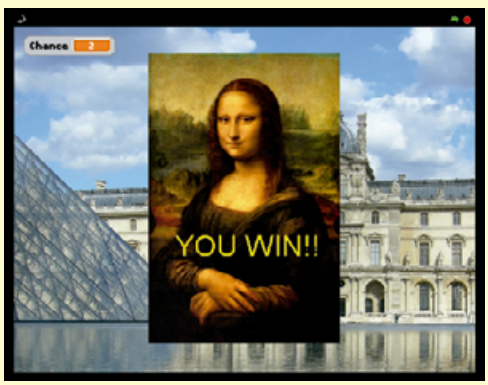
7
```
when I receive wrong
change Chance by -1
wait 1 secs
```

8
```
when clicked
set Chance to 3
forever
  if Chance < 1
    switch to costume monalisa_lose
    stop all
```

9
```
when I receive win
switch to costume monalisa_win
stop all
```

Finally, add three more programs to the Puzzle. Program 7 subtracts 1 from the Chance variable any time it receives the wrong broadcast. Programs 8 and 9 control when the winning and losing screens appear.

That's it! Remember to save your project (**File > Save**) before giving the game a try. Let's see if you can win this!



Scratchy's Challenge!!

Can you use the ask block and broadcasts to create a personality test? How about a flash-card game to learn words in a new language? Give it a try!