Solutions to Coin Flip Modeling Creative Projects

Projects Demo (English)	Mini Projects (English)	Mini Projects (Spanish)	Printable

Coin Flip is an app that simulates the flipping of a two-sided coin. This app uses App Inventor's random number generator and two images to simulate the coin flip.

Objectives: In this lesson you will learn to:

- add additional features to an existing mobile app;
- modify some of the code of an existing app;
- improve coding skills by solving simple and challenging programming problems;
- use *Math* random number blocks to generate a random value.

Click the image to watch video

Getting Ready

Open App Inventor with the Coin Flip Projects template. This will open the project that we completed in the Coin Flip Tutorials. Or, if you already have App Inventor open, you can use your project from the Coin Flip Tutorial. Use the *Save As* option to rename your project to *CoinFlipV2* or something to indicate that it is version 2 of that app.

If using the template linke, be patient. It sometimes takes a moment to retrieve and open the project.

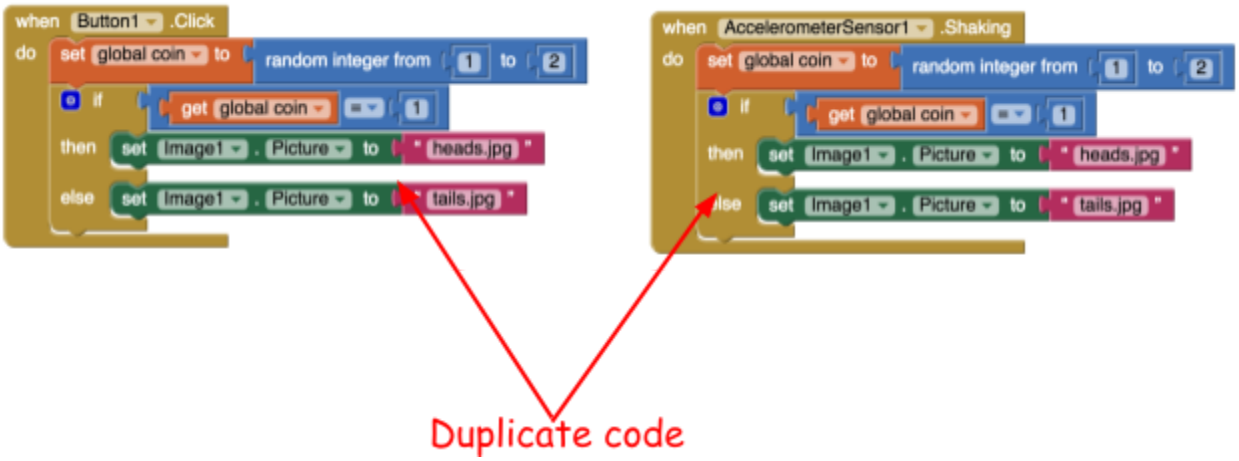
Coin Flip Mini Projects

Solutions to Mini Projects

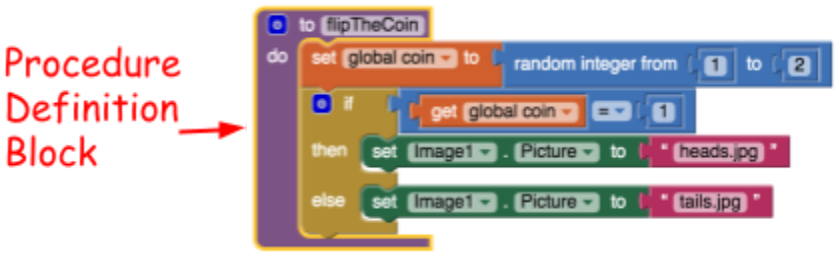
Here are some sample solutions.

1. Modify your app so that the user can also shake the phone to flip the coin. (HINT: Use the Accelerometer Sensor.) NOTE: Instead of copying and pasting the coin-flip algorithm, you'll want to use a ***procedure*** to reduce complexity in your code.

One way to do this is to incorporate an *Accelerometer* component (sensor drawer) into your app and then duplicate the code in the *When Button.Clicked* block in the Accelerometer's *When Acclerometer.Shaking* block:



Procedural Abstraction. But that's probably not how a computer scientist would do it. They would want to avoid the duplicate code. So rather than copying and pasting, we can use a procedure here. Let's organize the blocks in the *do* slot into a named procedure, *flipTheCoin:*



Then we can simply call this procedure in each of the event handlers.

This is another example of ***procedural abstraction***. Using a procedure in this way helps us ***reduce complexity*** in our code -- it makes the code more readable and easier to modify, as we will see in some of the subsequent solutions.

The procedure we created is like defining a new word in a language. It represents the process of flipping a coin and rather than describing the details of the process every time we want to flip a coin, we can just use the word (call the procedure).

2. Modify your app so that “heads” or “tails” is spoken when the coin is flipped. (HINT: Use the TextToSpeech component.)

Now that we have defined the *flipTheCoin* procedure, we need only make a change to that procedure -- instead of to the two event handlers. This is **another benefit of procedural abstraction** -- i.e., modifying and maintaining a program is easier because the code for flipping a coin is ***encapsulated*** in a single named procedure.

3. Modify the event handler in the Coin Flip app to use random fraction instead of random integer. (HINT: App Inventor’s ***random fraction block*** returns a decimal number between 0 and 1, not including 1. Some examples: 0, 0.25, 0.33, 0.5, 0.66, 0,75, 0.99.)

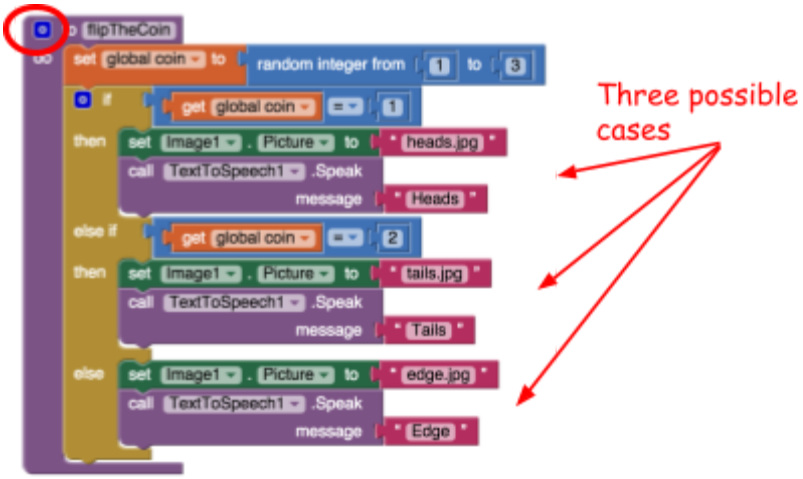
To model the 50:50 behavior of a coin flip, we need to test whether our random fraction is less

than or equal to 0.5. That should happen 50% of the time if App Inventor's random fraction block works well.

Procedural abstraction: Here again, we can implement this change by modifying the *flipTheCoin* procedure because it completely encapsulates the coin-flipping algorithm. If we didn't use a procedure, it would be much more difficult and error-prone to make this kind of change.

4. **If/else Algorithm:** You now have an app that can flip a two-sided coin. Modify your app that so that it can flip a three-sided coin. (Hint: You will need an if/else block with three conditions. Also, it might be better to use the *random integer block* for this problem. You'll need a third image for this problem; here's one that is openly licensed: coin on edge.)

For this problem you need to use the ***mutator widget*** on the ***if block*** (the little blue gear) to create an if/else block with three possible cases.



5. According to this report, if you stand a bunch of Lincoln pennies on their edge and then bang the table, they have a strong bias toward coming up heads. Let's suppose the coin has a 70% chance of coming up heads (30% tails) in this experiment. Create a model to simulate this biased coin. (HINT: Use *SaveAs* to create a new project for this problem.)

Here we want to change our condition to <= 0.7.

6. Real-World Simulations: you can use the random fraction block to simulate real world statistics.