# Make dice images dynamic

With our widgets created and linked with our Java code, we can now make the images dynamic and change them based on the values we get from our random number generator. We need to do two things to make that work. The first thing is alter how we are working with our images and then we will create a loop to sort through the die values and the die images and update which images they display. In Android, there are two main ways to store images. One, is saving them as a drawable. Drawables are good options for images that you use once in your app and don't change dynamically.

The other is to save them as a project asset. Assets are better solutions for dynamic images. If you try and use drawables, it is common to run out of memory when you run your app. So, Assets are the better way to do this. When you create a new Android project, it sometimes creates the drawable folder for you. You have to create the assets folder on your own and it requires some unique steps. Open the project for this movie, and right-click on the Res folder in your project. Select New and then go down to the bottom where it says Android Folder with a little Android icon then select assets Folder.

A new window will pop up. Go ahead and select Finish. Now, you'll have a new folder called assets in your project. We will take the images that we have in our drawable folder and we'll go ahead and select them, right-click and say Copy then select the assets folder, right-click and say Paste. Then click OK to confirm. Since we are only using the die\_1.png file as a drawable resource when we start the project, we can delete two through six from the drawable folder.

With our images migrated over as project assets now, we can create our code. If you go into our Java file and you go down to our roll dice method, we are going to create a loop that we will use to go through all the dice and images in our project and set everything up. Let's go ahead and create the structure for our loop first. After we've set our array list for our die values, we'll create a for loop. I'm going to set up our for loop with an integer called dieOfSet and we're going to initially set that to zero. dieOfSet needs to always be less than three for the loop to continue and we'll increment dieOfSet by one each time the loop runs.

With each loop, we will work with a unique die value and a unique image view. The die value will be an integer and we can use that integer to build the file name reference for the image asset that we want to display. That is why the files have the numbers one through six inside of them. Then we can tell the image view that corresponds with that die to display that image. To start, we need to build the file name of the asset that we want to display. We can do that by creating a new string and use the value of the die in the array list to create the file name.

Start with String and we'll set this as imageName Our image file names start with a die and then an underscore then we'll concatenate this with our array list dice and then we'll use get to get a specific item out of the array list. This index is going to be dieOfSet Then we'll concatenate the end of the file name which is a period png This will give us a string that will be die1, die2, die3, and so on for each side of the dice.

The next step is to get the asset based on the file name, create a drawable object based on that asset and set the drawable as the image for the image view in the array list of image views. We need to wrap this in a try catch block to handle any I/O errors we might get when we run the code. So we can do that first. Go ahead and start with try and then at the end we'll add catch. We're going to look for an I/O exception we'll call that e during the catch block.

If we get an exception, we can print the stack trace of the error e print stack trace. Now, in the try block we can write our code. First we need to access the image from the assets of the project and we'll save that as an input stream. Input stream equals stream equals get assets this will get from the specific assets folder that we created earlier. It's then going to use an open method and we pass in the image name that's the file name for the asset.

You'll need to import the input stream method so press alt enter. Next, we will create a new temporary drawable object and link it to the image stream that contains our asset reference. So we'll create a drawable. We'll just call it d for now and let's say that drawable is going to create from stream and then we'll pass in a reference to our stream which in the last line we called stream. There's a second attribute that we don't need in this method so we'll just pass in null.

Remember to input the drawable class by pressing alt enter. Last, we will use the array list of image views we created earlier and set the drawable using the temporary object we just created. So we'll access dice image views and we're going to use the get method and we're going to use dieOfSet so we can select that specific image view within the array list collection. Then, that particular object we're going to then use set image drawable and we'll set that to d which is the drawable that we created on the previous line.

With that, the loop is finished. Let's go ahead and run our app and see how it works. When you click the button, you'll see that the numbers in the text view match the images that are displayed underneath. Our code is using the number to build the file name of the asset it needs. We then use that file name to find the image in the project assets and then configure it as a drawable to dynamically change the reference for each image view within the collection and the display of the matching face on the die.

Because we're using an image view collection for both the die values and the dice images, we can loop through each one to reduce the size of our code.