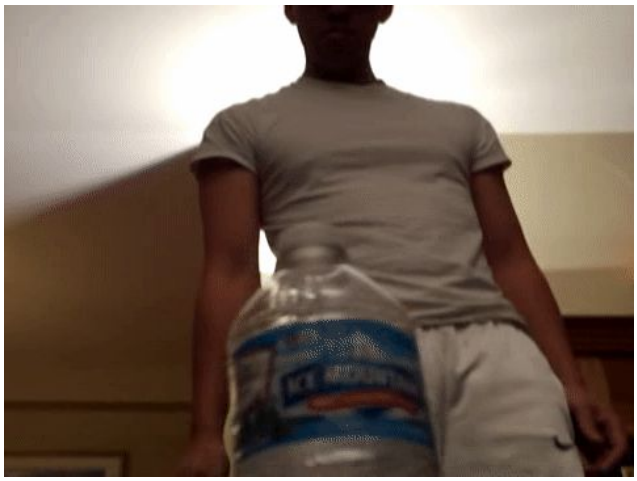
CoderDojo Castleknock

CoderDojo Castleknock

Animation and water bottle flipping

Water bottle flipping *is an activity and a challenge that involves throwing a plastic water bottle, typically full or partially full of liquid, into the air so that it rotates, in an attempt to land it upright on its bottom. In this class, you'll create an app with animated water bottle flipping and links to a youtube video of a fun bottle flipping examples.*



Water bottle flipping.

What You'll Build

With the app shown in Figure 1 you can:

- Show how to animate an image
- Show how can stop and start sound
- Show how you can open a web page from your app.
- Show how you can open a youtube video from your app.

CoderDojo Castleknock

Figure 1. The water bottle flipping app UI

CoderDojo Castleknock

What You'll Learn

This tutorial covers the following concepts:

- Show how to animate an image
- Show how can stop and start sound
- Show how you can open a web page from your app.
- Show you how to open a youtube video via a separate app.

Getting Started

Connect to the App Inventor website. Normally we would start a new project, but we are going to start with a ‘template app’ to make loading all the images we need to use.

Designing the Components

This app has 4 different main components (3 of which compose the buttons), listed in Table 1.

Animation works by changing images very fast so we think the picture is moving. An example is;

https://www.youtube.com/watch?v=YrRDA_IK29k

This is called flipbook animation, an example of creating a flipbook is;

https://www.youtube.com/watch?v=5A0Ro4vj3KM

We do the same thing for our app using the clock to tell us to change the image.

CoderDojo Castleknock

Since we have so many images, it would get pretty boring to load them all of them before starting to write our program, so we are going to load a ‘template’ app with the images already loaded.

Bottle flip app

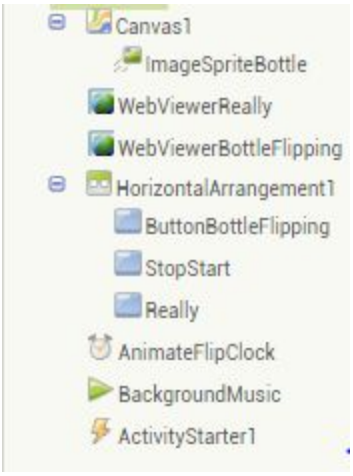
Download this to your computer and then upload it to your app inventor account. To do this ;

- Click on the bottle flip app
- When you run this, the source code shown downloads to your laptop where it can be then uploaded to you App Inventor tab via;

When you run this, the Bottleflip_template shown download to your PC where it can be then uploaded to you App Inventor tab via;

Projects -> Import Project (.aia) from my computer...

We'll break down the app into its functional parts and build them sequentially by going back and forth between the Designer and the Blocks Editor.



CoderDojo Castleknock

Component Type	Pallette Group	What you will name it	Purpose
HorizontalArrangement	Layout	HorizontalArrangement1	Holder for all the buttons below so they all go on the same line
Button	Basic	ButtonBottleFlipping	Go to wikipedia website explaining bottle flipping
Button	Basic	Really	Go to youtube video of World Record.
Button	Basic	StopStart	Either pauses or continues the flipping and music.
Canvas	Drawing and Animation	Canvas1	We have to put our image on a Canvas
ImageSprite	Drawing and Animation	ImageSpriteBottle	Bottle flipping
Clock	Sensors	AnimateFlipClock	Tell us when to flip to a new bottle flipping image
Player	Media	BackgroundMusic	Play music of a drumroll while we flip bottle.

Table 1. All of the components for the BottleFlip app

Creating the front screen

Our user interface will be a image of bottle flipping at the top of the screen and 3 buttons at the bottom of the screen.

Working Buttons

We will have the buttons

CoderDojo Castleknock

- Whats is Bottle Flipping ?
- Start (or Stop), depending if the animation is running.
- Really ?

Which do the following;

- Shows a wikipedia article on a webpage explaining what bottle flipping is.
- Pause the music and animation or restart both.
- Start a youtube video of a bottle flipping.

Start by creating the first two xylophone keys, which we will implement as

The view in the Component Designer should look something like Figure 2.

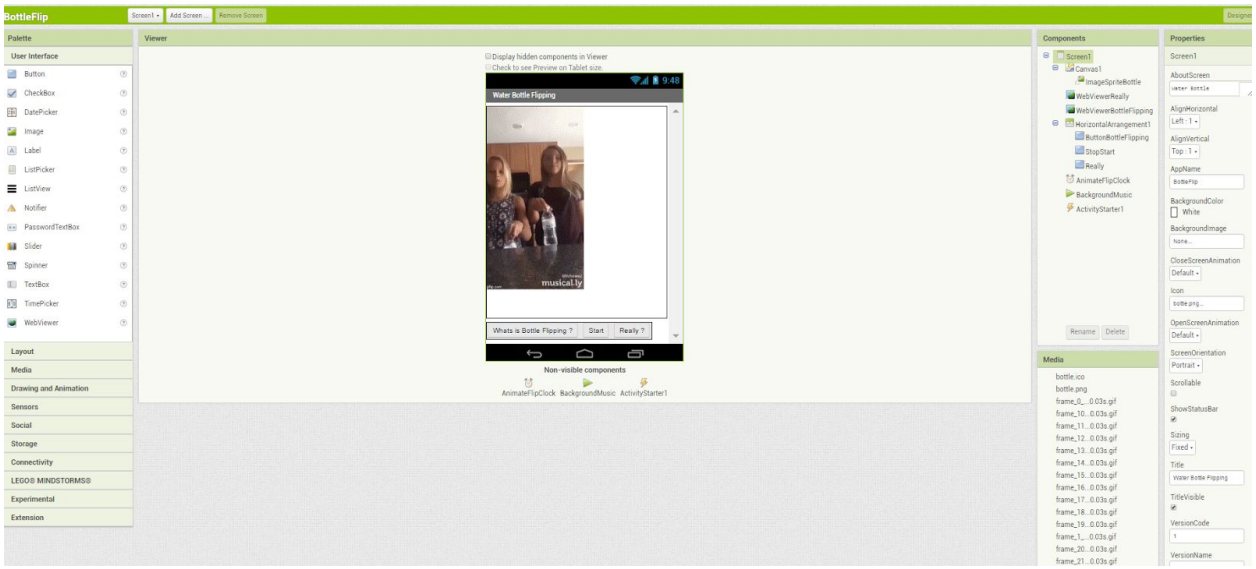


Figure 2. Placing buttons to create screen

The display on your phone should look similar.

CoderDojo Castleknock

Really Button

We are going to;

- Pause the music
- Change the animate variable to false so we know to restart music
- Call a youtube app to run a youtube video

We now switch to the Block view, and from the Really drawer, drag out the call **Really.Click**

Now add to this;

- From the BackgroundMusic drawer the **BackgroundMusic.Pause** call to pause the drum roll music.
- From the AnimateFlipClock drawer the **AnimateFlipClock.TimerEnabled** setting to false dragged out from the logic Drawer to stop the clock running.
- From the Canvas1 drawer the **Canvas1.Visible** setting to false dragged out from the logic Drawer to hide the animation of bottle flipping.
- From the ActivityStarter1 drawer the **ActivityStarter1.Action** setting. Change this “android.inviewtent.action.VIEW” by dragged out from the text Drawer an empty string block like figure 3.

Figure 3 Empty String block.

to saw what we want to do, our “intent”.

- From the ActivityStarter1 drawer the **ActivityStarter1.DataURI** setting. Change this “vnd.youtube:G9P2iUS2oFE” by dragged out from the text Drawer an empty string block like figure 3.

  to tell the youtube app what video we want to play.

- From the ActivityStarter1 drawer the **ActivityStarter1.StartActivity** procedure to call the youtube app.The Really.Click block should now look like figure 4.

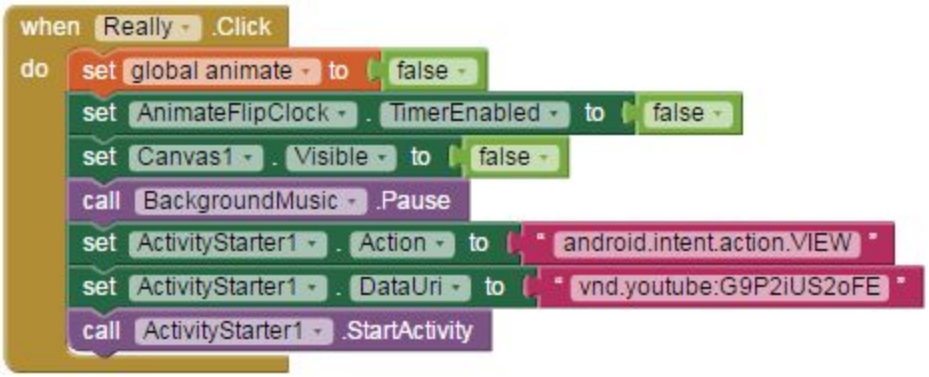


Figure 4. Really.Click

ButtonBottleflipping Button.

We are going to;

- Pause the music
- Change the animate variable to false so we know to restart music
- Stop the clock that tells us to animate the bottle flipping.
- Hide the Canvas that shows the bottle flipping.

- Show the Website that explains what bottle flipping is.

From the ButtonBottleFlipping drawer, drag out the call **ButtonBottleFlipping.Click**

Now add to this;

- From the BackgroundMusic drawer the **BackgroundMusic.Pause** call to pause the drumroll music.
- From the AnimateFlipClock drawer the **AnimateFlipClock.TimerEnabled** setting to false dragged out from the logic Drawer to stop the clock running.
- From the Canvas1 drawer the **Canvas1.Visible** setting to false dragged out from the logic Drawer to hide the bottle flipping.
- From the WebViewBottleFlipping drawer the **WebViewBottleFlipping.Visible** setting to true dragged out from the logic Drawer to show the webpage.
- From the WebViewBottleFlipping drawer the **WebViewBottleFlipping.Width** setting to from the Canvas1 Drawer the **Canvas1.Width** to set the webpage to be as wide as the Canvas that the bottle flipping animation are on.
- From the WebViewBottleFlipping drawer the **WebViewReally.Height** setting to from the Canvas1 Drawer the **Canvas1.Height** to set the webpage to be as tall as the Canvas that the bottle flipping animation are on.

The ButtonBottlrFlipping.Click block should now look like figure 5.

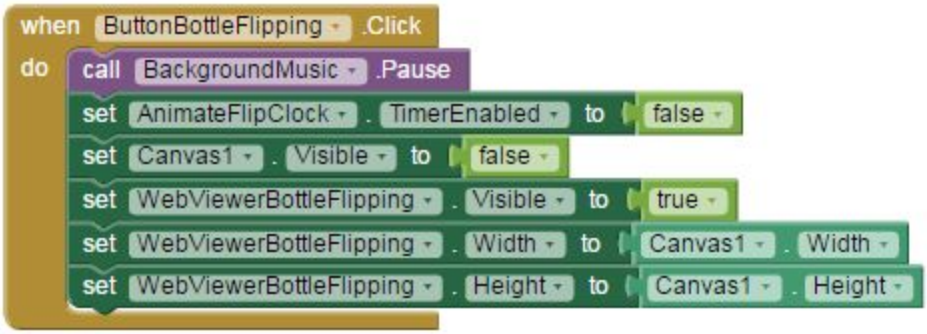
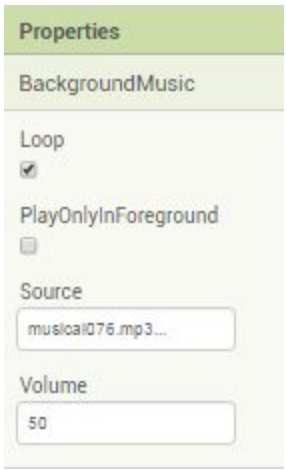


Figure 5. ButtonBottleFlipping.Click

Adding the Sound Component

The bottle flipping deserves a drumroll. So create a **Sound** component called BackgroundMusic.

Its **Source** is set to musical076.mp3, See Figure 6.



CoderDojo Castleknock

Figure 6. BackgroundMusic properties

Playing and pausing the music

The behavior we need to program is for a sound file to play when;

1. Play when the application starts
2. Pause when the pause button is pressed.
3. Play when the play button is pressed.

The play button when pressed to pause button when pressed. This is called a binary action, i.e. it does one of two things.

We can set Play when the application starts by in the Blocks Editor as shown in Figure 7 by doing the following

1. From the My Blocks tab and Screen1 drawer, drag out the **Screen1.Intialize** block.
2. From the BackgroundMusic drawer, drag out the call **BackgroundMusic.Start**.

Figure 7. Playing a sound when application starts.

CoderDojo Castleknock

We can set the play and pause when pressing the Play/Pause when the StopStart button pause button is pressed, by in the Blocks Editor as shown in Figure 5 by doing the following

1. From the My Blocks tab and StopStart drawer, drag out the **StopStart.Click** block.
2. We create a global variable called animate to keep track of if we are playing or not. From the Built-in drawer, drag out the **initialize global(name)** from variables and change name to animate and set it to true like figure 8.
3. From the Built-in drawer, drag out the if-then from controls and put it in the **StartStop.Click** block.
4. Click on the blue button to show the else 'leg' of the if control and drag the else into the if, like figure 9.
5. From the StopStart Button drawer, drag out the set StopStart.Text to either "Start" or "Pause" depending on which 'leg' of the if-then-else statement you are in, like figure 9.

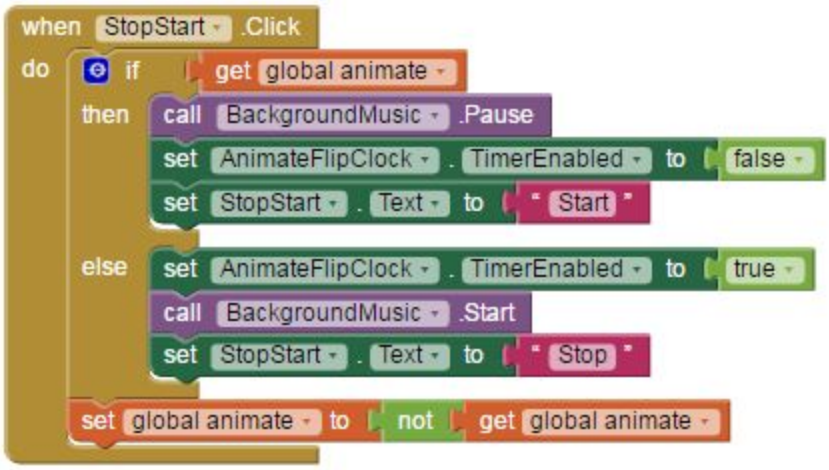


Figure 7. Stopping and starting sound.

Change to

Figure 8. Keeping track of playing using a global variable

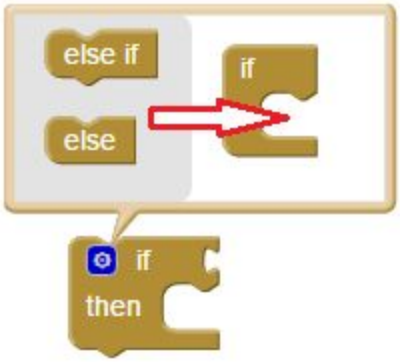


Figure 9. If-then to if-then-else

Let's Animate !

To animate, we need a couple of things;

1. A Clock to tell us when to change images that make up the animation.
2. Some variables to keep track of;
    - The current frame number to be shown
    - The last frame number to be shown
    - The rest of the name of the image, this consists of
        - Frame_start
        - Frame_number

- Frame_end

3. Some logic to tell us when to repeat the images

Clock

We use theAnimateClock, setting it to run when the application starts, with it going off every 50 milli-seconds.

As an aside, some examples of time;

- 1 millisecond (1 ms) – cycle time for frequency 1 kHz; duration of light for typical photo flash strobe; time taken for sound wave to travel ca. 34 cm
- 1.000692286 milliseconds – time taken for light to travel 300 km in a vacuum
- 3 milliseconds – a housefly's wing flap
- 3.3 milliseconds – normal delay time between initiation and detonation of a C4 explosive charge
- 5 milliseconds – a honey bee's wing flap
- 5 milliseconds to 80 milliseconds – a hummingbird's wing flap
- 10 milliseconds (10 ms) – a jiffy, cycle time for frequency 100 Hz
- 50 milliseconds – the time interval between gear changes on a Lamborghini Aventador

Figure 6 Lamborghini Aventador

- 5 to 80 milliseconds – typical latency for a broadband internet connection (important for playing online games)
- 134 milliseconds – time taken by light to travel around the Earth's equator

- 185 milliseconds – the duration of a full rotation of the main rotor on Bell 205, 212 and 412 helicopters (normal rotor speed is 324 RPM)

Figure 7 Bell 412EP of the Los Angeles City Fire Department

- 200 milliseconds – the time it takes the human brain to recognize emotion in facial expressions
- 300 to 400 milliseconds – the time for the human eye to blink
- 495 milliseconds – an approximate average of the round trip time for communications via geosynchronous satellites
- 860 milliseconds – average human resting heart cycle time
- 86,400,000 (24 × 60 × 60 × 1000) milliseconds – one day
- 604,800,000 (24 x 60 x 60 x 1000 x 7) milliseconds – one week
- 31,556,908,800 (86,400,000 × 365.242) milliseconds – one year

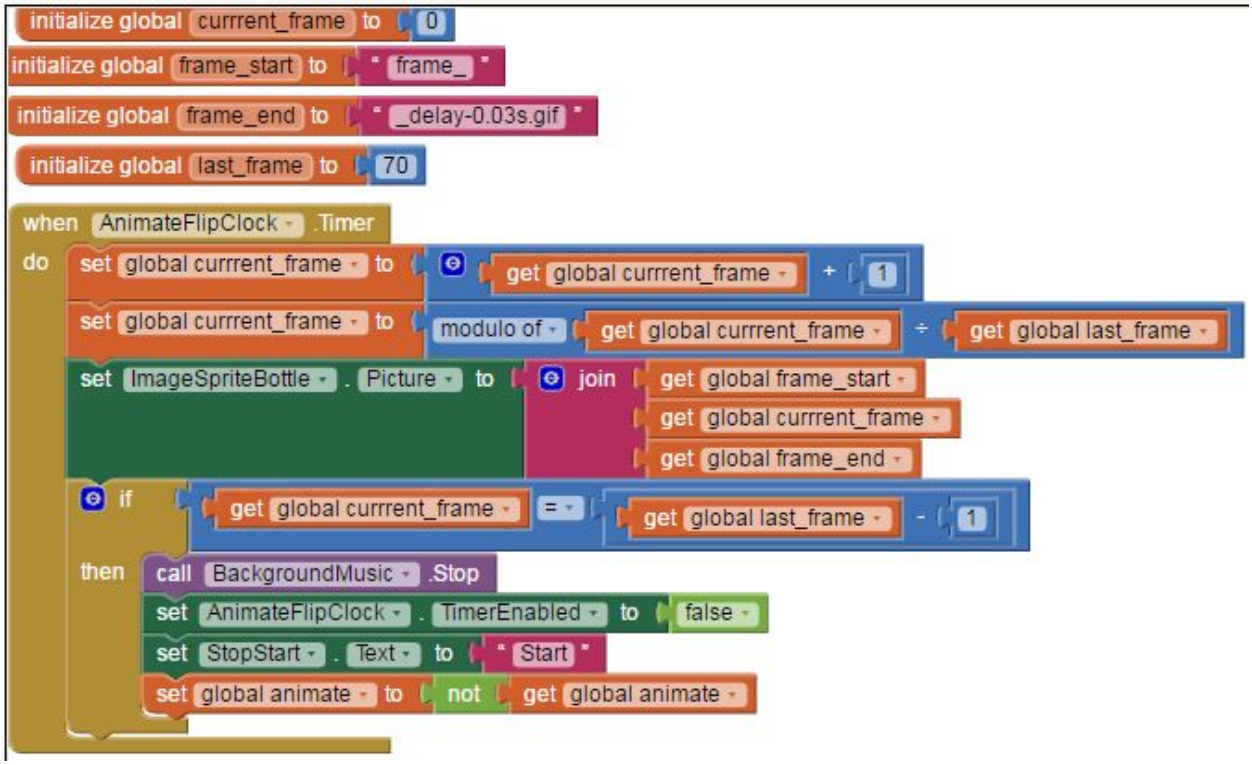


Figure 9. AnimateFlipClock.Timer

We drag **when AnimateFlipClock.Timer** from the AnimateFlipClock Drawer
This procedure is run every time the clock goes off, i.e. we have set it to go off every 50 milliseconds.

Variables

We create 4 global variables like figure 8 by dragging a global variable from the Blocks tab and renaming it and assigning a value as shown.

Logic

We need to keep track of what frame we are currently showing (we use the variable current_frame for this).
We need to use some maths to cycle around the frames. We use the maths function modulus for this.

CoderDojo Castleknock

A familiar use of modular arithmetic is in the 12-hour clock, in which the day is divided into two 12-hour periods. If the time is 9:00 now, then 4 hours later it will be 1:00. Usual addition would suggest that the later time should be 9 + 4 = 13, but this is not the answer because clock time "wraps around" every 12 hours; in 12-hour time, there is no "13 o'clock". Figure 11 shows this addition. Because the hour number starts over after it reaches 12, this is arithmetic modulo 10.

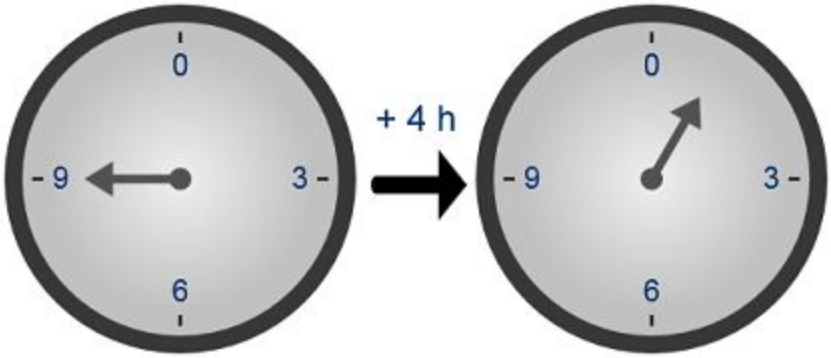


Figure 10. 12 hour clock = arithmetic modulo 12

For our case, we have 70 frame (we use the global variable last_frame to keep track of this information) so we use arithmetic modulo 70 like Figure 11.

CoderDojo Castleknock

Current Frame	Last Frame	module current frame / Last Frame	Frame Name
0	70	0	frame_1_delay-0.03s.gif
1	70	1	frame_2_delay-0.03s.gif
2	70	2	frame_3_delay-0.03s.gif
3	70	3	frame_4_delay-0.03s.gif
4	70	4	frame_5_delay-0.03s.gif
5	70	5	frame_6_delay-0.03s.gif
6	70	6	frame_7_delay-0.03s.gif
7	70	7	frame_8_delay-0.03s.gif
8	70	8	frame_9_delay-0.03s.gif
9	70	9	frame_10_delay-0.03s.gif
10	70	10	frame_11_delay-0.03s.gif
.	.	.	.
.	.	.	.
.	.	.	.
61	70	61	frame_62_delay-0.03s.gif
62	70	62	frame_63_delay-0.03s.gif
63	70	63	frame_64_delay-0.03s.gif
64	70	64	frame_65_delay-0.03s.gif
65	70	65	frame_66_delay-0.03s.gif
66	70	66	frame_67_delay-0.03s.gif
67	70	67	frame_68_delay-0.03s.gif
68	70	68	frame_69_delay-0.03s.gif
69	70	69	frame_70_delay-0.03s.gif
70	70	0	frame_1_delay-0.03s.gif

NB
we add 1 to the module result to get a 1-70 range of frames.

Figure 11. module 70 table

To the AnimateFlipClock.Timer block like figure 9 we add our modulo of operator from the Block Tab in the Math Drawer, we drag the modulo of operator like figure 12.

Figure 12 module of operator

We want it to;

- add one to the current_frame variable

- set the current_frame to the modulo of (or remainder) of when we divide the current_frame by the last frame so the AnimateBearsClock.Timer block should now look like figure 9

Test your work by trying it on the phone.

Ps

We created this tutorial standing on the shoulders of giants. We want to acknowledge the original art work created for musical.ly - "Create beautiful **music** videos with your favorite songs, and share with friends. **Musical**.ly is the world's fastest growing social network around **music** and lifestyle."

and the drum roll in the App which was got from http://zisca.xyz/get/militar-drum-roll-long-free-game-asset-audio/eXQtLS1HdVNLZ1NidVZqYw

CoderDojo Castleknock

Title: **Military Drum Roll [LONG] - Free Game Asset - Audio**

Published: **July 30, 2015**

Uploader: **MobileForge**

Duration: **00:11**

Finally, have fun.

I see a mistake !

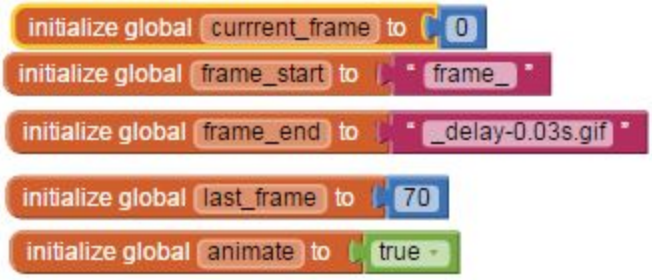
If you see a mistake, email coderdojocastleknock@gmail.com so we can fix this tutorial.

CoderDojo Castleknock

Appendix 1.

Variables

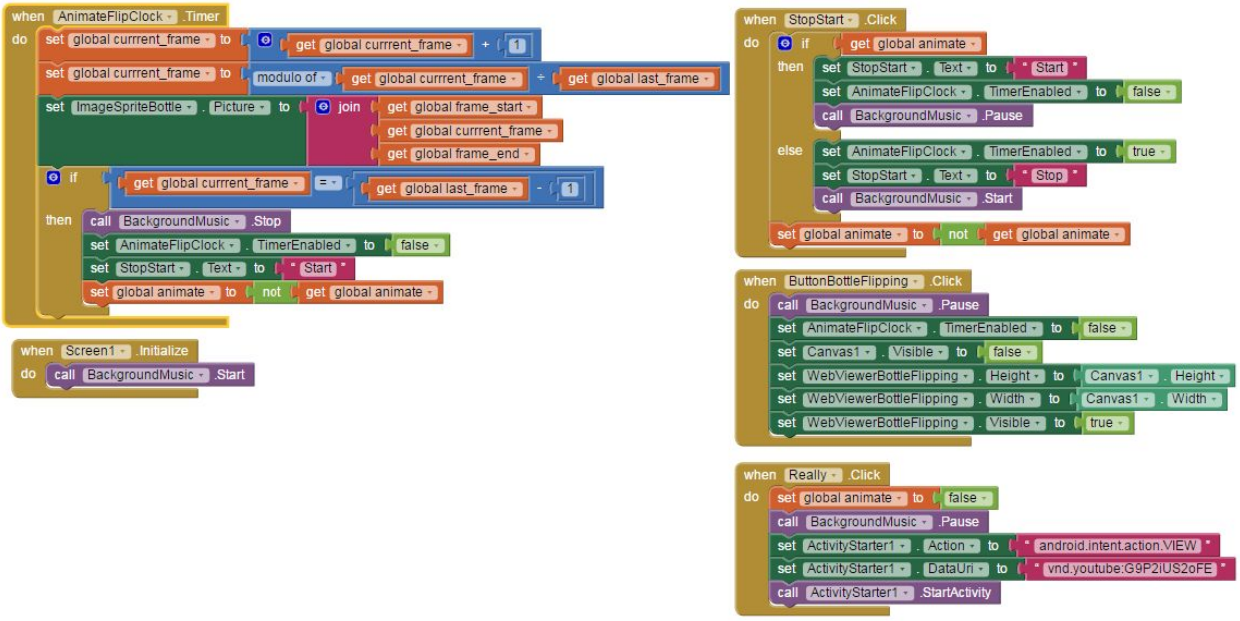
- current_frame
- Frame_start
- Frame_end
- Last_frame
- animate



Built-in blocks

- When Screen1 Initalize
- When AnimateFlipClock.Timer
- When.StopStart.Click
- When ButtonBottleFlipping.Click
- When Really.Click

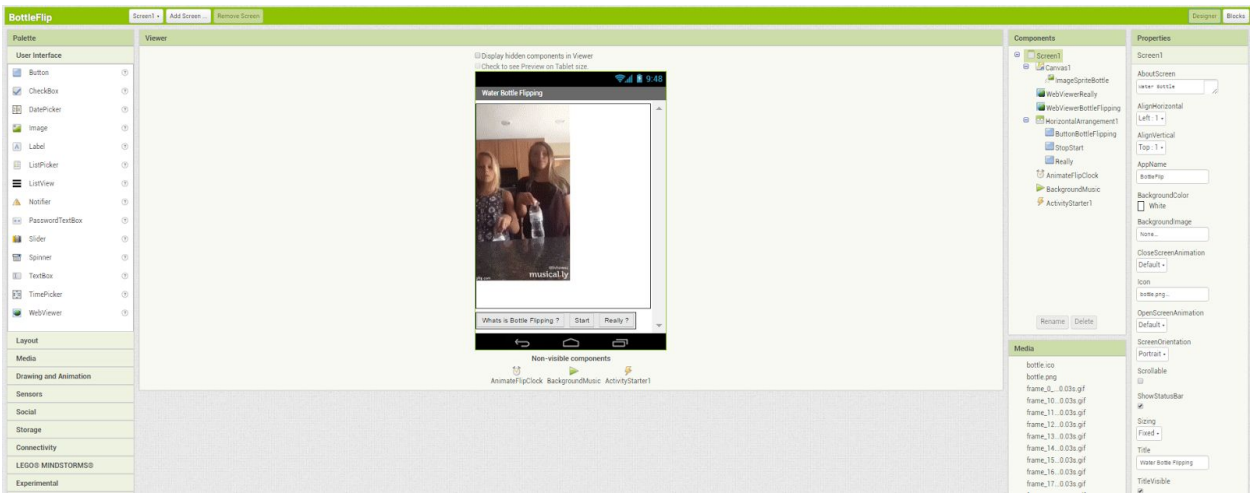
CoderDojo Castleknock



Procedure Blocks

None

Designer



CoderDojo Castleknock

Source

Bottle flip

Download this to your computer and then upload it to your app inventor account. To do this ;

- Click on the bottle flip app
- When you run this, the source code shown downloads to your laptop where it can be then uploaded to you App Inventor tab via;

Projects -> Import Project (.aia) from my computer...