We will make this game a bit more challenging for the user to play, and we are also going to make this game mobile compatible so that we can host this game on GitHub and wrap it in an APK, so you can play it on your mobile phone and share it with your friends.

We move onto the first challenge, which is to **add multiple ropes**. But, here we will not be using the balloon to push the fruit. Rather, we are going to create an arrangement that the user has to think about which rope should be cut so that fruit lands on the bunny. The user can do that in many ways by applying your creativity.

We are going to **create three ropes**, after which we need to create buttons for each rope and call the function to drop the fruit. We already have a button and rope on the canvas, but we need to change its position. We have changed the position of the bunny in such a way that the user has to think in what order the ropes have to be cut so that fruit lands on the bunny. Let's place all the buttons on the canvas, that will make it easy for us to place the ropes.

We will **create three buttons**, one is going to be at the **far left-hand side** of the canvas, the next two buttons will be on the **right-hand side**, but one button is close to the top and the other is at the **center** of the canvas.

All the buttons will have the same background image. But we need to attach them with different functions. One function **drop** is already defined, **drop2** and **drop3** functions we will define later

We got the buttons on the screen, now we need to **add two more ropes**. Creating the additional ropes is very easy, first define two more variables such as var rope2 and var rope3; Next, in the setup() function we can assign the rope object to these variables. While creating the rope object we need to tell how many sections(elements) each rope will have, this will control the length of the rope. The second thing is we need an x and y position where the rope will be connected on the canvas.

Calling these functions will display the ropes, but when we run the program we can see that only one rope is attached with fruit.

To **attach the fruit** with the other two ropes, we need to **create the constraint** between them. By using the **Link** class

We have made a Link class, we just need to create the object of that class and while creating the object we need to pass the two bodies. Here bodies will be rope and fruit, we already have a constraint between rope and fruit, now we just need to create this constraint between rope2 and rope3.

Now, we need to write the code to drop the fruit from the other two ropes.

We have already attached the function with the buttons, now we will define two more functions to drop the fruit.

In the functions, we are calling the **break**() function from the rope class and the **detach**() function from the Link class. After that, we are setting the constraint as null so that it does not interfere with our fruit body. Now if we press on any button, the rope will detach from there and fruit falls down. We have to cut the rope carefully so that fruit lands on the bunny.

Now the task is to **make this game compatible with mobile phones** and to **host it online** so that we can make an **installable app** out of it.

if you want to share your game with your friends, you can do it By hosting it on **GitHub**

We have done this with the T-rex game.

So we are going to follow the same process here, which is:

1. Make the game compatible with different screen sizes.

2. Host on GitHub

3. Make an installable file with Thunkable

To make the game adapt to any screen size, we already have inbuilt variables in p5js.

But if the screen size is changing, this will create problems. To overcome that, we will use the **windowWidth** and **windowHeight** variables instead of having a fixed screen size.

This will automatically change the screen size based on the size of the screen of the user. But here is the problem, **this will only work for a computer screen**. This won't work with the mobile screen. For **mobile screens**, we need to **use** **displayWidth** and **displayHeight**, and we are going to use both of these

We will create a variable var isMobile, which will store the type of mobile such as iphone, ipad or android. Then we will write a condition that will check the device we are currently on, if we are on a mobile device it will use diplayWidth and displayHeight, if we are on a computer screen it will use windowWidth and windowHeight. Now, create two variables as var canW and canH, these will store the width and height of the screen.

**Note**: We are adding 80 to the displayWidth in order to properly fit the screen width. As in some devices, game width is lesser than the screen width. If you feel the game is stretched too much in the mobile, you can remove the 80 or reduce the value to an appropriate number

This will create a **dynamic screen size**. But it will not work. Because we need to add a very important line of code in our HTML file that will make the webpage screen responsive.

Now, we need to change a few more things, such as **the position of the bunny and the ground.**

we are placing the ground and bunny with **respect to the new screen size**. So the ground y-position will be canH which is the height of the screen. And the y-position of the bunny will be canH-80. So the bunny will be placed on the ground. One more important change we need to make in the background image, the **background image** is placed at 0,0 which is the top left corner of the canvas. But the image size will be fixed as 500 x 700px. This will not cover our entire screen if the screen size is bigger. But we can specify the image size as well, and here we will do that using displayWidth and displayHeight. We will also add some value in displayWidth to make sure the image covers the entire screen size.

**Host the game on GitHub.**

You can set any **name** for the repo but at the **end of it, it should be .github.io.**

the final step for this class create an APK file using Thunkable