Junior Computer Academy

ROBLOX STUDIO

STEP computer ACADEMY

Junior Computer Academy

Roblox Studio

Lesson 3. Fading Trap

STEP computer ACADEMY

Attention!

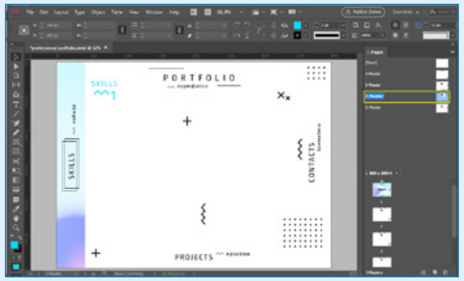
This lesson contains interactive options of the PDF format: buttons, video and audio materials, hyperlinks, bookmarks, and page navigation.

Browsers and applications for viewing PDF files built into operating systems and mobile devices may not be able to display such content correctly.

We recommend opening the lesson in free programs: Adobe Reader or Foxit PDF Reader for correct viewing.

THIS LESSON CONTAINS IMAGE CAROUSELS

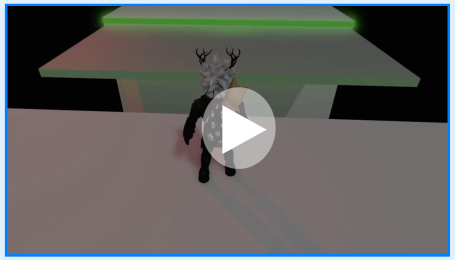
You can switch between images by clicking the gray buttons. A button of the currently active Figure differs in color.



Figures: 1 2

THIS LESSON CONTAINS VIDEOS

This lesson contains videos marked with PLAY icon. Click on it to watch the video, and it will open in a separate window.



Contents

Fading Trap

In the previous lesson, you learned how to trigger code based on player behavior. This lesson will show you how to make a platform which fades away when a player steps on it. Just like in Video 1.

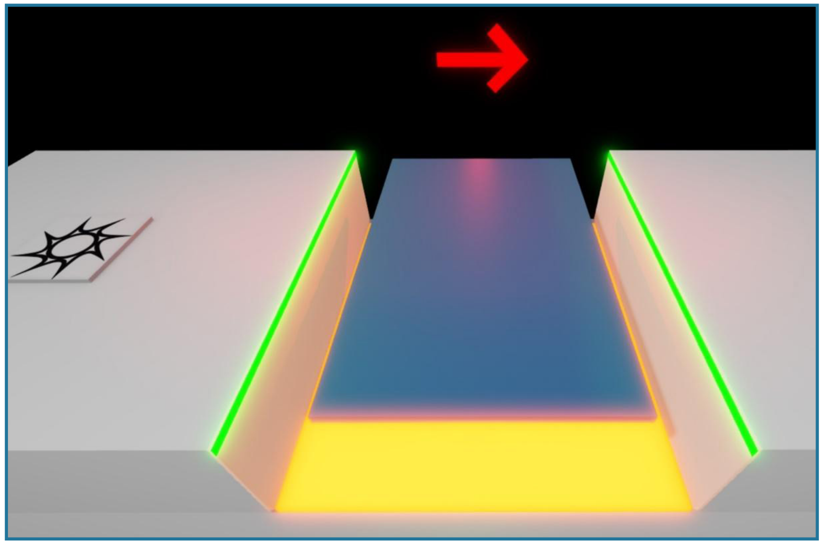
Video 1

CREATING A PLATFORM

Setting Up

After the previous lesson, you can place your fading platform above the lava floor — either way, it should go over a space that players can't jump across.

1. Insert a part and move it into place in your game world — call it FadingPlatform.
2. Resize it so a player can jump on it (Fig. 1).
3. Make sure it's anchored.
4. Insert a Script into the part, rename it to FadeOnTouch, and remove the default code (Fig. 2).



Figures: 1 2

5. Create a variable for the platform and an empty function connected to the platform's Touched event.

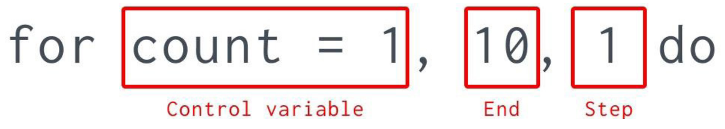
```lua
local platform = script.Parent

local function fade()

end

platform.Touched:Connect(fade)
```

Fading the Platform

Having the platform vanish in an instant would be no fun at all — players would find it impossible to get across the gap. It would be better if the platform could fade away before players could fall through it to give them a chance to jump off.

You could change the Transparency property and wait a very short time over and over again to get this effect, but a gradual fade would require at least 10 changes between 0 and 1. That's 20 lines of very repetitive code.

This can be achieved much more effectively using a for loop which repeats code a specific number of times. Each loop of the code is known as an iteration. A for loop is defined with three things, separated by commas (Fig. 3):



```lua
for count = 1, 10, 1 do
```

Figure 3

- **Control variable** — The variable created and used to count the loops. In this example, it's count and the starting value is 1.
- **End value** — The value it has to get to for the loop to stop. In this example, it's 10.
- **Step increment** (optional) — Determines what to add to the control variable each loop. If left out, it defaults to 1, so in this example it's unnecessary.

1. In the function, create a for loop starting from 1 which iterates 10 times.
2. Inside the for loop, set the Transparency property to the control variable divided by 10.
3. Call the wait function with a time of 0.1.

```lua
local platform = script.Parent

local function fade()
for count = 1, 10 do
platform.Transparency = count / 10
wait(0.1)
end
end

platform.Touched:Connect(fade)
```

When the loop runs, count increases by 1 with each iteration. This means that the platform's Transparency will increase by 0.1 every 0.1 seconds, reaching full transparency after 1 second.

Reappearing

After the platform has vanished, players should fall through it. The platform should also come back a few seconds after it fades — otherwise, players would never get to try the jump again if they failed. The CanCollide property controls whether players can fall through a part.

1. Set the CanCollide property of the platform to false after the for loop.
2. Wait for a few seconds using the wait function.
3. Set the CanCollide property back to true.
4. Set the Transparency property back to 0.

```
local platform = script.Parent

local function fade()
for count = 1, 10 do
platform.Transparency = count / 10
wait(0.1)
end
platform.CanCollide = false
wait(3)
platform.CanCollide = true
platform.Transparency = 0
end

platform.Touched:Connect(fade)
```

DEBOUNCING

In the previous lessons, you learned that the Touched event runs each time a player's body part comes into contact with the part. This behavior causes issues when a player runs across the fading platform: the function will run multiple times, resetting the loop each time.

For the code to work properly, the function should only run once when the player touches the platform for the first time. Ensuring that an action is only triggered once when it would otherwise be triggered multiple times is known as debouncing.

Debouncing Variable

To debounce a function, a boolean variable can be used to keep track of when the platform has already been touched. Boolean just means the only values it can contain are true and false.

Create a variable called isTouched and set it to false.

```
local platform = script.Parent

local isTouched = false

local function fade()
for count = 1, 10 do
platform.Transparency = count / 10
wait(0.1)
end
platform.CanCollide = false
wait(3)
```

```
platform.CanCollide = true
platform.Transparency = 0
end

platform.Touched:Connect(fade)
```

Checking the Variable

An if statement can be used to only run the code in the fade function if the isTouched debouncing variable is false.

Wrap the body of the fade function in an if statement with the condition not isTouched.

```
local platform = script.Parent

local isTouched = false

local function fade()
if not isTouched then
for count = 1, 10 do
platform.Transparency = count / 10
wait(0.1)
end
platform.CanCollide = false
wait(3)
platform.CanCollide = true
platform.Transparency = 0
end
end

platform.Touched:Connect(fade)
```

Negate Operator

The Lua not operator reverses the value of whatever follows it. In conditional terms, this means that the if statement on the left behaves the same as either statement on the right.

```
if not isTouched then

end




```

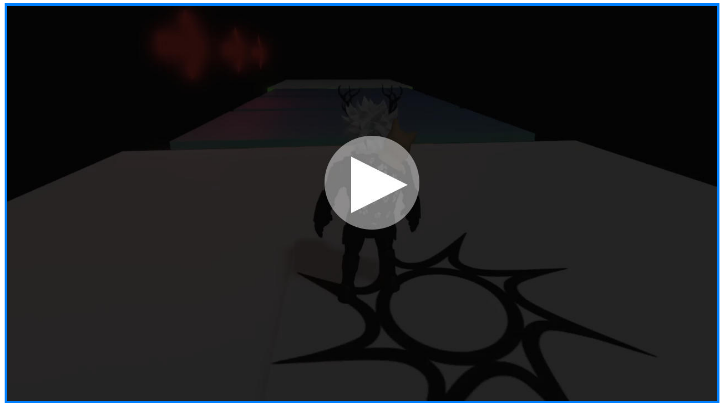
```
if isTouched == false then

end

if isTouched == nil then

end
```

Toggling the Debounce

Currently, the code in the fade function will always run because isTouched is false and not isTouched evaluates to true. To complete the debounce routine, you'll need to toggle the variable's value in two places.

1. Set isTouched to true inside the if statement, before the platform begins to fade.
2. Set it back to false once the platform has reappeared.

```
local function fade()
if not isTouched then
isTouched = true
for count = 1, 10 do
platform.Transparency = count / 10
wait(0.1)
end
platform.CanCollide = false
wait(3)
platform.CanCollide = true
platform.Transparency = 0
isTouched = false
end
end

platform.Touched:Connect(fade)
```



Video 2

And that's it! Test your code now, and you should find that the platform fades away when the player jumps on it and comes back a few seconds later.

You can duplicate this platform across a wider gap to create a challenging obstacle and change the speed at which they fade to balance the difficulty. The example is in Video 2.

Final Code:

```
local platform = script.Parent

local isTouched = false

local function fade()
if not isTouched then
isTouched = true
for count = 1, 10 do
platform.Transparency = count / 10
wait(0.1)
end
platform.CanCollide = false
wait(3)
platform.CanCollide = true
platform.Transparency = 0
isTouched = false
end
end

platform.Touched:Connect(fade)
```

Roblox Studio
Lesson 3. Fading Trap