Do It Yourself Academy
D·I·Y·A
Learn | Achieve | Become

Roblox 1 : Mouse and Keyboard Input

Table of Contents

- Inputs
  - Keyboard Input
  - Mouse Input

Inputs

In Roblox, input means how players interact with the game using things like keyboards, mouse clicks, and controllers. It's important for game makers to understand how players use these things so they can make the game fun and easy to play. By paying attention to input, game creators can design games where players can move around easily, do cool actions, and feel like they're really part of the game world.

Keyboard Input

Keyboard input in Roblox refers to how players control their characters and interact with the game environment using keys on their keyboards.Roblox provides developers with tools and functions to detect keyboard input and respond accordingly, allowing for the creation of dynamic and responsive gameplay mechanics.

1. Create a new ***LocalScript*** inside ***StarterPlayer -> StarterPlayerScripts*** in the ***Explorer*** window.

> ⚠ You must choose ***LocalScript*** instead of usual the ***Script***.

2. We need to get references to the ***Players*** service to access the local player and the ***UserInputService*** to detect keyboard input. Additionally, we'll want to wait for the player's character to be added to the game before proceeding.

```
local Players = game:GetService("Players")
local UIS = game:GetService("UserInputService")

local player = Players.LocalPlayer
local character = player.Character or player.CharacterAdded:Wait()
```

3. Use the ***InputBegan*** event of the ***UserInputService*** to detect when a key is pressed. Inside this event, we'll check if the `Shift` key is pressed.

```
local Players = game:GetService("Players")
local UIS = game:GetService("UserInputService")
```

```
local player = Players.LocalPlayer
local character = player.Character or player.CharacterAdded:Wait()

local sprinting = false

UIS.InputBegan:Connect(function(input, GPE)
    if input.KeyCode == Enum.KeyCode.LeftShift then
        sprinting = true
        character.Humanoid.WalkSpeed = 50 -- Set sprinting speed
    end
end)
```

4. When the Shift key is pressed, we'll toggle a boolean variable to indicate that the character is sprinting. We'll also change the character's walk speed to simulate sprinting. When the Shift key is released, we'll toggle the sprinting variable back and reset the character's walk speed to the default value.

```
local Players = game:GetService("Players")
local UIS = game:GetService("UserInputService")

local player = Players.LocalPlayer
local character = player.Character or player.CharacterAdded:Wait()

local sprinting = false

UIS.InputBegan:Connect(function(input, GPE)
    if input.KeyCode == Enum.KeyCode.LeftShift then
        sprinting = true
        character.Humanoid.WalkSpeed = 100 -- Set sprinting speed
    end
end)

UIS.InputEnded:Connect(function(input)
    if input.KeyCode == Enum.KeyCode.LeftShift then
        sprinting = false
        character.Humanoid.WalkSpeed = 16 -- Reset to default speed
    end
end)
```

5. Run the game and playtest it. You can also change the `WalkSpeed` to see how the character will react.

Mouse Input

Mouse click events in Roblox enable players to interact with objects and environments within the game world simply by clicking their mouse buttons. These events are fundamental for

implementing gameplay mechanics such as picking up items, triggering actions, and interacting with interactive elements.

1. Create a new ***LocalScript*** inside ***StarterPlayer -> StarterPlayerScripts*** in the ***Explorer*** window.

2. First, we define a function named `changeColor` that takes a single argument part. Inside this function, we change the color of the given part to a random color using `BrickColor.Random()`.

```
local function changeColor(part)
    part.BrickColor = BrickColor.Random() -- Change to a random color
end
```

3. We then connect this function to the `MouseClick` event of each part without a `ClickDetector` attached. To achieve this, we use the `LocalPlayer.CharacterAdded` event to ensure that we connect the function to parts only after the player's character has been added to the game.

```
local function changeColor(part)
    part.BrickColor = BrickColor.Random() -- Change to a random color
end

-- Connect the function to the ClickDetector's MouseClick event for each part
game.Players.LocalPlayer.CharacterAdded:Connect(function(character)

end)
```

4. For each part in the workspace, we check if it's a ***BasePart*** and doesn't already have a `ClickDetector` attached. If these conditions are met, we proceed to create a new `ClickDetector` instance and parent it to the part.

```
local function changeColor(part)
    part.BrickColor = BrickColor.Random() -- Change to a random color
end

-- Connect the function to the ClickDetector's MouseClick event for each part
game.Players.LocalPlayer.CharacterAdded:Connect(function(character)
    for _, part in pairs(workspace:GetDescendants()) do
        if part:IsA("BasePart") and not part:FindFirstChild("ClickDetector") then



        end
```

```
    end
end)
```

5. Once the `ClickDetector` is created, we connect the `MouseClick` {.inline-code} event of the `ClickDetector` to the `changeColor` function, passing the corresponding part as an argument. This ensures that when the part is clicked, its color changes.

```
-- Function to change the part's color
local function changeColor(part)
    part.BrickColor = BrickColor.Random() -- Change to a random color
end

-- Connect the function to the ClickDetector's MouseClick event for each part
game.Players.LocalPlayer.CharacterAdded:Connect(function(character)
    for _, part in pairs(workspace:GetDescendants()) do
        if part:IsA("BasePart") and not part:FindFirstChild("ClickDetector") then
            local clickDetector = Instance.new("ClickDetector", part)
            clickDetector.MouseClick:Connect(function() changeColor(part) end)
        end
    end
end)
```