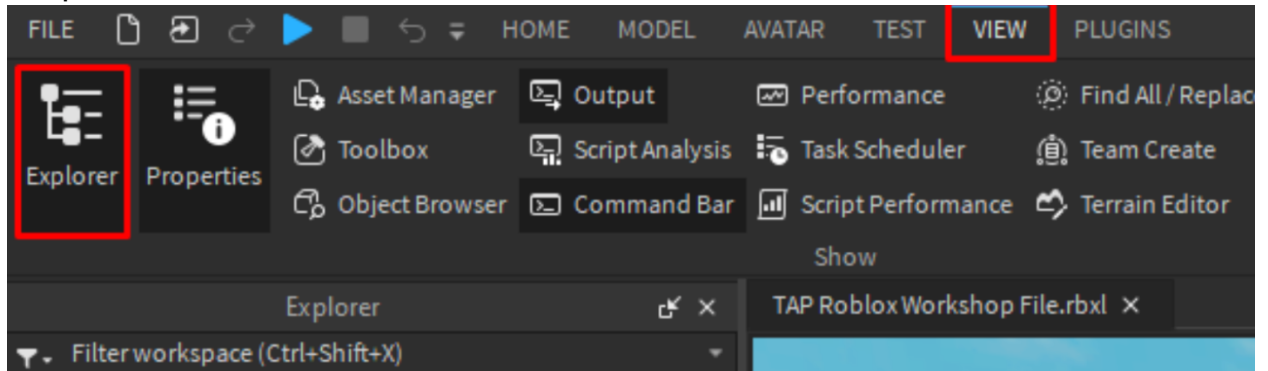


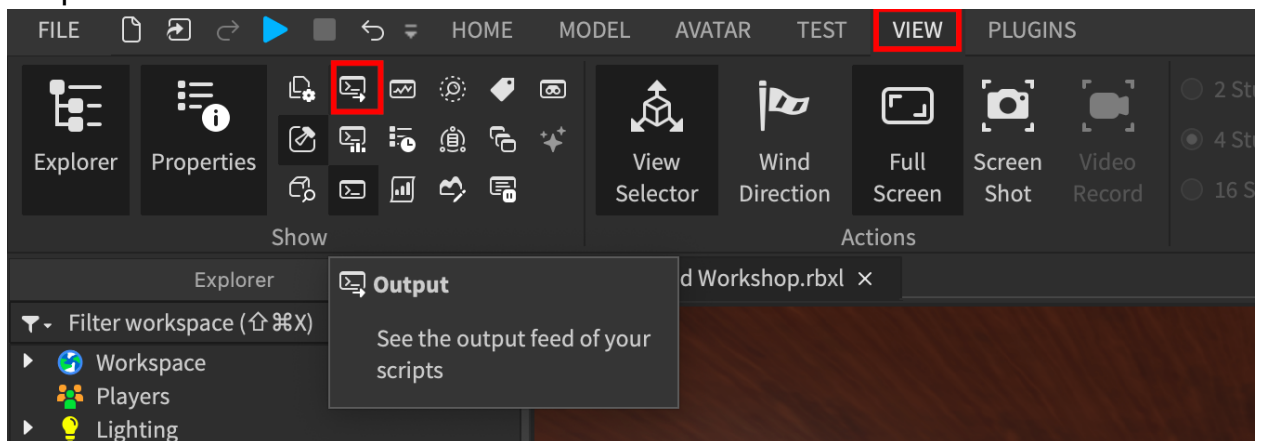
# Virtual Warriors

## Advanced Roblox Workshop Walkthrough

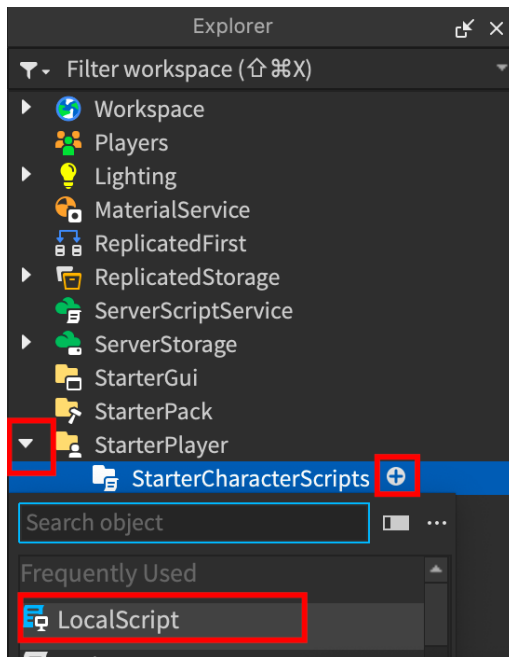
1. Open the “TAP Advanced Workshop.rbxl” file
2. Open the Explorer window by going to View at the top, then click “Explorer”



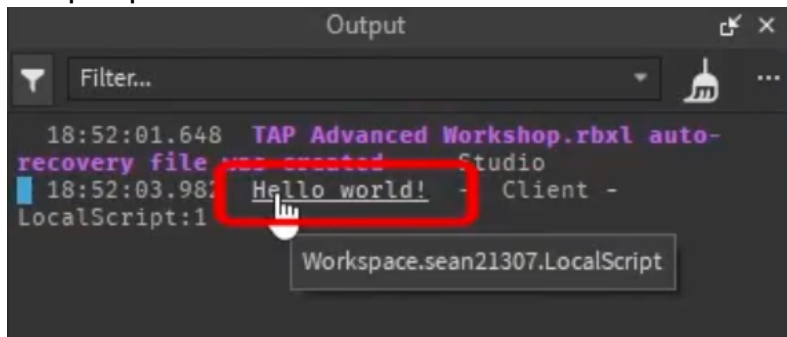
3. Open the Output window by going to View at the top, then click the output icon



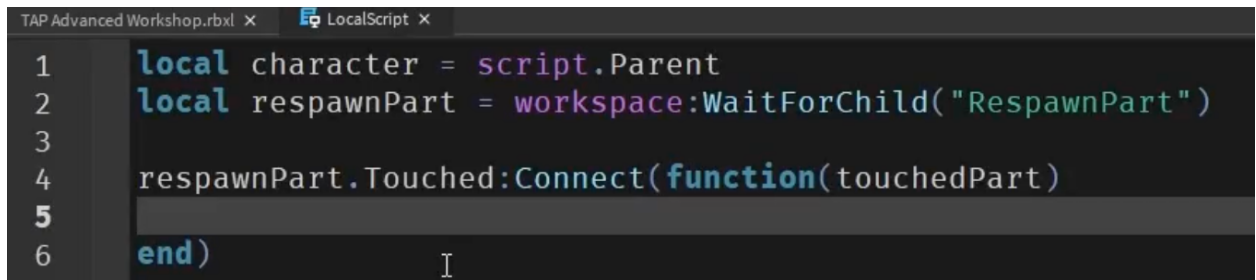
4. In the explorer panel, expand “StarterPlayer”
5. Press the plus icon at the end of StarterCharacterScripts, and click LocalScript



6. The script will have the default code, **print("Hello World!")**. If you press the blue Play button at the top, then "Hello World!" will be printed in the Output panel



7. Press the red "Stop" button and click back on the LocalScript
8. First, we need to create a variable that references the player's character, by typing **local character = script.Parent**, because scripts placed in "StarterCharacterScripts" will be made a child of the character
9. Next, we need to make a variable that references the RespawnPart in the pit, so that we can write code to interact with it. To do this, we use a method provided by Roblox called "WaitForChild()", which will yield the script until that object exists or loads in, so the code is **local respawnPart = workspace.WaitForChild("RespawnPart")**.
10. Now we can use these variables to interact with the world. Roblox provides "events" which you can "connect" to, so that the game engine will tell you when something happens. In this case, we will use the **.Touched** event so we know when something touches the RespawnPart.



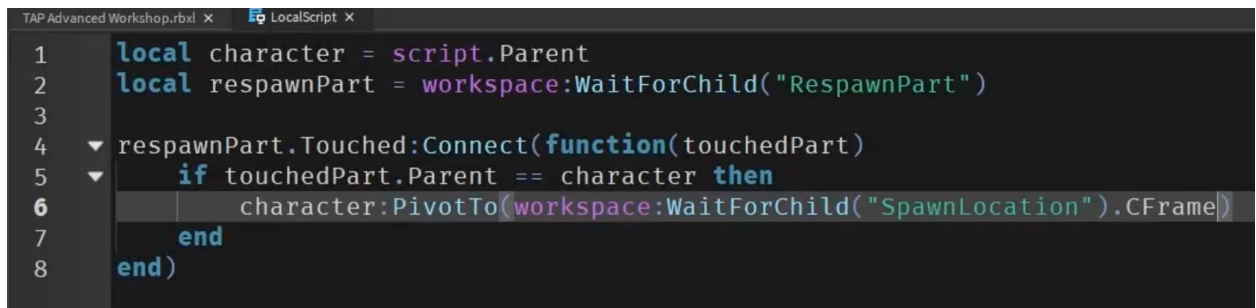
```

1  local character = script.Parent
2  local respawnPart = workspace:WaitForChild("RespawnPart")
3
4  respawnPart.Touched:Connect(function(touchedPart)
5
6  end)

```

11. Since the event gives us the part that touches the respawnPart, we can use it to check if the character fell. To do this, we check if touchedPart.Parent is the character (**if touchedPart.Parent == character then**).

12. If the **if statement** is true, then we will teleport the character back to spawn (the start of the level). For this, we use the **:PivotTo()** method of a model.



```

1  local character = script.Parent
2  local respawnPart = workspace:WaitForChild("RespawnPart")
3
4  ▼ respawnPart.Touched:Connect(function(touchedPart)
5  ▼   if touchedPart.Parent == character then
6      character:PivotTo(workspace:WaitForChild("SpawnLocation").CFrame)
7   end
8  end)

```

13. Now, you can test the game by clicking the blue Play button, and you will see that if you fall in the pit it teleports you back to where you spawned. However, your character bounces around all over. This is because we are teleporting the character directly to the spawn's position, so it ends up in the ground and the engine glitches out.

14. To fix this issue, we can apply an offset to the position we are teleporting the character to. To do this, we need to multiply the existing CFrame by an offset CFrame like **\* CFrame.new(0, 3, 0)**.



```

1  local character = script.Parent
2  local respawnPart = workspace:WaitForChild("RespawnPart")
3
4  ▼ respawnPart.Touched:Connect(function(touchedPart)
5  ▼   if touchedPart.Parent == character then
6      local teleportCFrame = workspace:WaitForChild("SpawnLocation").CFrame
7      character:PivotTo(teleportCFrame * CFrame.new(0, 3, 0))
8   end
9  end)

```

15. If you test the game once again, you will see that your character is smoothly teleported and doesn't act strangely when teleported back to spawn.
16. The next behavior we need to add is something to prevent players from running along the sides of the map to avoid the obstacle course. To do this, we can make it so if they touched the lights on the wall, they are knocked into the pit.
17. First, we need to create a variable to reference the group of lights, so **local lightGroup = workspace.WaitForChild("WallLights")**
18. **lightGroup** is a specific type of object in Roblox called a group, which just acts as a container of objects. Because it is a group, we can call **:GetChildren()** on it, which provides us with an array of the group's children (the lights).
19. We can loop through the children, so that we can add behavior to each light.

```
for _, light in pairs(lightGroup:GetChildren()) do
end
```

20. To keep the code organized, we can create a function and pass the light object as a parameter to add behavior to it.

```
local function OnLightAdded(light)
end

for _, light in pairs(lightGroup:GetChildren()) do
  OnLightAdded(light)
end
```

21. For now, we can add the same teleport behavior from before to the lights.

```
local function OnLightAdded(light)
  light.Touched:Connect(function(touchedPart)
    if touchedPart.Parent == character then
      local teleportCFrame = workspace.WaitForChild("SpawnLocation").CFrame
      character:PivotTo(teleportCFrame * CFrame.new(0, 3, 0))
    end
  end)
end
```

22. If you test the game again, you should see there you are teleported back to spawn once you touched any of the lights.
23. However, we can make it more engaging by programming behavior that makes the lights knock you into the pit. To do this, we have to assign the character's velocity, which includes a direction and a magnitude. The direction will be the way the light is facing, and the magnitude we can modify to get the right number to know the character in the pit.

```
local function OnLightAdded(light)
    light.Touched:Connect(function(touchedPart)
        if touchedPart.Parent == character then
            local flingVelocity = light.CFrame.LookVector
            character.PrimaryPart.Velocity = flingVelocity
        end
    end)
end

for _, light in pairs(lightGroup:GetChildren()) do
    OnLightAdded(light)
end
```

24. If you test the game, you will see that when you touch the lights, you are knocked into the pit, which in turn teleports you back to spawn.
25. One more thing we can add is changing the color of the light to red once it's touched. To do this, we set the **.Color** property of the light once it's touched, and then we reset it to its old color after a split second.

```
local function OnLightAdded(light)
    light.Touched:Connect(function(touchedPart)
        if touchedPart.Parent == character then
            local flingVelocity = light.CFrame.LookVector * 100
            character.PrimaryPart.Velocity = flingVelocity

            local oldColor = light.Color
            light.Color = Color3.fromRGB(255, 0, 0)
            task.wait(0.1)
            light.Color = oldColor
        end
    end)
end
```

26. If you test the game one final time, everything should be working properly. If you complete the obstacle, you will be able to get the ice cream!

27. Congratulations! Now you should be able to make your own behavior with **.Touched** and similar events.