transform.position += new Vector3(0f, 1f, 0f);

----------------------------------

Destroy(this.gameObject)

All you're doing in your code is checking for mousebutton events. You need to tell the engine where the mouse cursor is, what object you're clicking on. You also probably have this script attached to each cube, that's why ALL of them are changing. Look into [Raycast](https://docs.unity3d.com/Documentation/ScriptReference/Physics.Raycast.html), and use [ScreenPointToRay](https://docs.unity3d.com/Documentation/ScriptReference/Camera.ScreenPointToRay.html) to define your [Ray](https://docs.unity3d.com/Documentation/ScriptReference/Ray.html).

Here's an example. Don't forget to either rename the class in the script, or rename the script to match the class name. Attach THIS script to one object only, preferably the main camera.

1. using UnityEngine;
2. using System.Collections;
4. public class RaycastExample : MonoBehaviour
5. {
6. Ray ray;
7. RaycastHit hit;
9. void Update()
10. {
11. //Be sure to have a main camera that is tagged "MainCamera" for this example to work.
12. ray = Camera.main.ScreenPointToRay(Input.mousePosition);
13. if(Physics.Raycast(ray, out hit))
14. {
15. //Left Click, change to red.
16. if(Input.GetMouseButtonDown(0))
17. {
18. hit.collider.renderer.material.color = Color.red;
19. }
21. //Right Click, change to blue.
22. if(Input.GetMouseButtonDown(1))
23. {
24. hit.collider.renderer.material.color = Color.blue;
25. }
26. }
27. }
28. }

You can also use [OnMouseOver](https://docs.unity3d.com/Documentation/ScriptReference/MonoBehaviour.OnMouseOver.html) function for this, which would be a simpler solution but require you to attach the script to every object you'll need to click on. I'd tell you to use [OnMouseDown](https://docs.unity3d.com/Documentation/ScriptReference/30_search.html?q=Onmousedown), but [OnMouseDown](https://docs.unity3d.com/Documentation/ScriptReference/30_search.html?q=Onmousedown) only detects left mouse button. Using [OnMouseOver](https://docs.unity3d.com/Documentation/ScriptReference/MonoBehaviour.OnMouseOver.html) will allow you to detect that you're hovering over the object, then use your Input instruction to determine the button pressed.

1. using UnityEngine;
2. using System.Collections;
4. public class OnMouseOverExample : MonoBehaviour
5. {
6. void OnMouseOver()
7. {
8. if(Input.GetMouseButtonDown(0))
9. renderer.material.color = Color.red;
10. if(Input.GetMouseButtonDown(1))
11. renderer.material.color = Color.blue;
12. }
13. }
15. using UnityEngine;
16. using System.Collections;
17. public class RaycastExample : MonoBehaviour
18. {
19. Ray ray;
20. RaycastHit hit;
21. void Update()
22. {
23. //Be sure to have a main camera that is tagged "MainCamera" for this example to work.
24. ray = Camera.main.ScreenPointToRay(Input.mousePosition);
25. if(Physics.Raycast(ray, out hit))
26. {
27. //Left Click, change to red.
28. if(Input.GetMouseButtonDown(0))
29. {
30. hit.collider.renderer.material.color = Color.red;
31. }
32. //Right Click, change to blue.
33. if(Input.GetMouseButtonDown(1))
34. {
35. hit.collider.renderer.material.color = Color.blue;
36. }
37. }
38. }
39. }
40. using UnityEngine;
41. using System.Collections;
42. public class RaycastExample : MonoBehaviour
43. {
44. Ray ray;
45. RaycastHit hit;
46. void Update()
47. {
48. //Be sure to have a main camera that is tagged "MainCamera" for this example to work.
49. ray = Camera.main.ScreenPointToRay(Input.mousePosition);
50. if(Physics.Raycast(ray, out hit))
51. {
52. //Left Click, change to red.
53. if(Input.GetMouseButtonDown(0))
54. {
55. hit.collider.renderer.material.color = Color.red;
56. }
57. //Right Click, change to blue.
58. if(Input.GetMouseButtonDown(1))
59. {
60. hit.collider.renderer.material.color = Color.blue;
61. }
62. }
63. }
64. }

Another code:-

1. using UnityEngine;
2. using System.Collections;  
     
   public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/ScriptReference/MonoBehaviour.html) {
3. void Example() {
4. transform.Translate(1, 1, 1);
5. }
6. }

# [Transform](https://docs.unity3d.com/ScriptReference/Transform.html).position

[SWITCH TO MANUAL](https://docs.unity3d.com/Manual/class-Transform.html)

public [Vector3](https://docs.unity3d.com/ScriptReference/Vector3.html) **position**;

## Description

The position of the transform in world space.

using UnityEngine;

using System.Collections;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/ScriptReference/MonoBehaviour.html) {

void Example() {

transform.position = new [Vector3](https://docs.unity3d.com/ScriptReference/Vector3.html)(0, 0, 0);

print(transform.position.x);

}

}

# [Transform](https://docs.unity3d.com/ScriptReference/Transform.html).rotation

using UnityEngine;

using System.Collections;  
  
public class ExampleClass : [MonoBehaviour](https://docs.unity3d.com/ScriptReference/MonoBehaviour.html) {

void Example() {

transform.rotation = [Quaternion.identity](https://docs.unity3d.com/ScriptReference/Quaternion-identity.html);

}

}

# Quaternion

struct in UnityEngine

## Description

Quaternions are used to represent rotations.

They are compact, don't suffer from gimbal lock and can easily be interpolated. Unity internally uses Quaternions to represent all rotations.  
  
They are based on complex numbers and are not easy to understand intuitively. You almost never access or modify individual Quaternion components (x,y,z,w); most often you would just take existing rotations (e.g. from the [Transform](https://docs.unity3d.com/ScriptReference/Transform.html)) and use them to construct new rotations (e.g. to smoothly interpolate between two rotations). The Quaternion functions that you use 99% of the time are: [Quaternion.LookRotation](https://docs.unity3d.com/ScriptReference/Quaternion.LookRotation.html), [Quaternion.Angle](https://docs.unity3d.com/ScriptReference/Quaternion.Angle.html), [Quaternion.Euler](https://docs.unity3d.com/ScriptReference/Quaternion.Euler.html), [Quaternion.Slerp](https://docs.unity3d.com/ScriptReference/Quaternion.Slerp.html), [Quaternion.FromToRotation](https://docs.unity3d.com/ScriptReference/Quaternion.FromToRotation.html), and [Quaternion.identity](https://docs.unity3d.com/ScriptReference/Quaternion-identity.html). (The other functions are only for exotic uses.)  
  
You can use the Quaternion.operator \* to rotate one rotation by another, or to rotate a vector by a rotation.