

```

0 references
public class MovePlayer : MonoBehaviour
{
1 reference
    public float thrust;           // variable created which will be used to apply thrust to an object. vector3 times thrust is equal ne
    2 references
    public Rigidbody rigid_body;   // reference of class Rigidbody created. This reference will be used to access it's properties.

    0 references
    void Start()
    {
        rigid_body = GetComponent<Rigidbody>(); // getting the components/properties of the class and stored in rigid_body for manipu
    }
    0 references
    void Update()
    {
        processThrust();
        processRotation();
    }
    1 reference
    void processThrust()
    {
        if(Input.GetKey(KeyCode.Space))
        {
            rigid_body.AddRelativeForce(Vector3.up * thrust); // Adds a force to the rigidbody relative to its coordinate system.
            // rigid_body.AddRelativeForce(0, 1, 0);
        }
    }
}
1 reference

```

For adding force to an object

Vector3.up , Vector3.down, Vector3.left, etc. for vector movements.

Static Properties

back	Shorthand for writing Vector3(0, 0, -1).
down	Shorthand for writing Vector3(0, -1, 0).
forward	Shorthand for writing Vector3(0, 0, 1).
left	Shorthand for writing Vector3(-1, 0, 0).
negativeInfinity	Shorthand for writing Vector3(float.NegativeInfinity, float.NegativeInfinity, float.NegativeInfinity).
one	Shorthand for writing Vector3(1, 1, 1).
positiveInfinity	Shorthand for writing Vector3(float.PositiveInfinity, float.PositiveInfinity, float.PositiveInfinity).
right	Shorthand for writing Vector3(1, 0, 0).
up	Shorthand for writing Vector3(0, 1, 0).
zero	Shorthand for writing Vector3(0, 0, 0).

Rotating an object on press of a button

```
void processRotation()
{
    if(Input.GetKey(KeyCode.A))
    {
        transform.Rotate(Vector3.forward * Torque * Time.deltaTime);
    }
    else if(Input.GetKey(KeyCode.D))
    {
        transform.Rotate(Vector3.back * Torque * Time.deltaTime);
    }
}
```