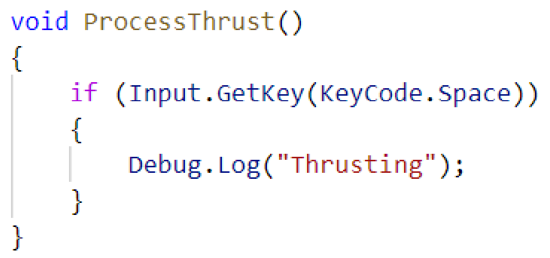
Practicing Methods

Step 1: create a method for moving the rocket vertically after update function

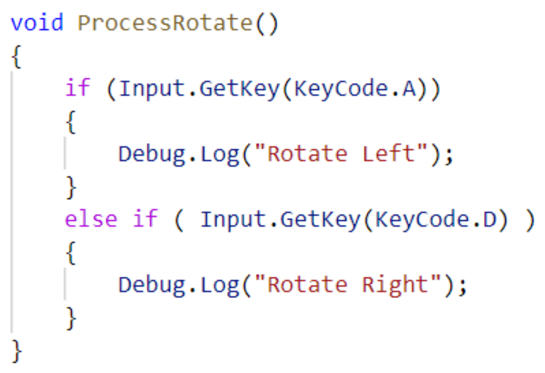
- Open Movement script
- Create the method after update method as the following



```
void ProcessThrust()
{
    if (Input.GetKey(KeyCode.Space))
    {
        Debug.Log("Thrusting");
    }
}
```

Step2 : create a method for rotating the rocket to the left or right after update function

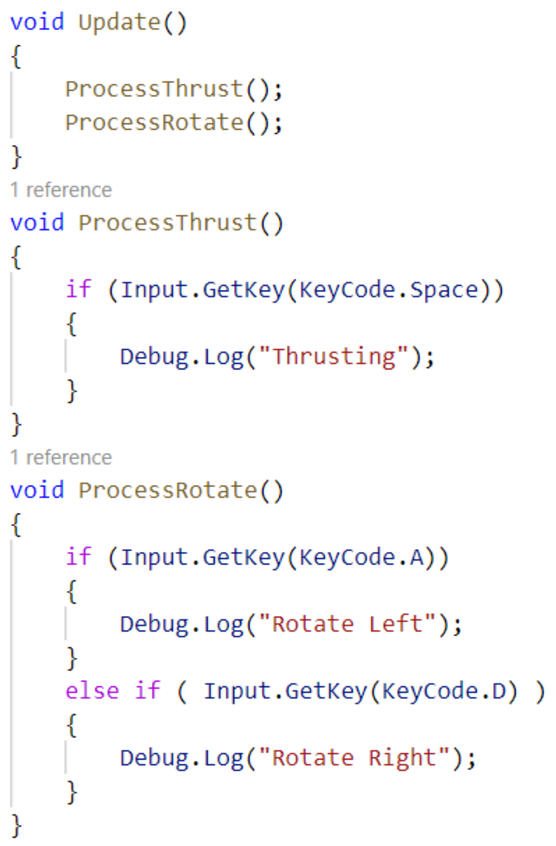
- Create the method after ProcessThrust method as the following



```
void ProcessRotate()
{
    if (Input.GetKey(KeyCode.A))
    {
        Debug.Log("Rotate Left");
    }
    else if ( Input.GetKey(KeyCode.D) )
    {
        Debug.Log("Rotate Right");
    }
}
```

Step3 : call the ProcessThrust and ProcessRotate methods inside update method

- The whole code is as the following

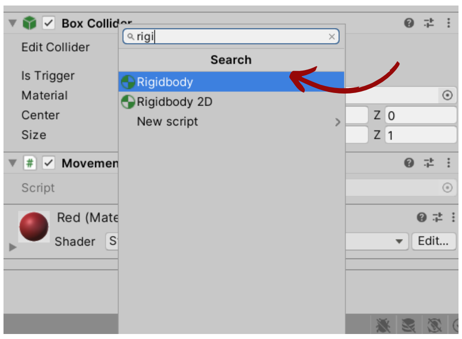


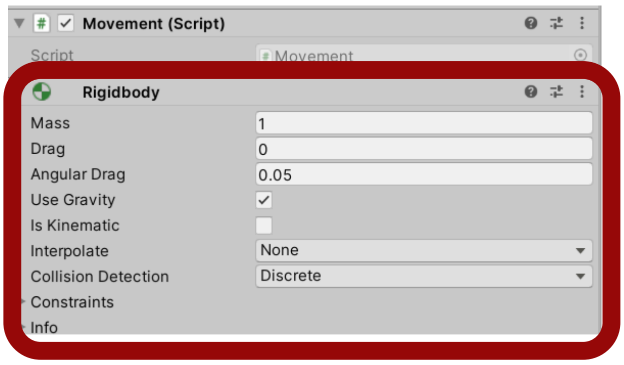
```
void Update()
{
    ProcessThrust();
    ProcessRotate();
}
1 reference
void ProcessThrust()
{
    if (Input.GetKey(KeyCode.Space))
    {
        Debug.Log("Thrusting");
    }
}
1 reference
void ProcessRotate()
{
    if (Input.GetKey(KeyCode.A))
    {
        Debug.Log("Rotate Left");
    }
    else if ( Input.GetKey(KeyCode.D) )
    {
        Debug.Log("Rotate Right");
    }
}
```

Physics Laws

Step 1: Add "Rigidbody" Component to the "Rocket" Object

- Click on the "Rocket" object
- Go to the Inspector
- Scroll down to the end of the Inspector
- Click on "Add Component"
- Search for "Rigidbody" and click on it
- Then in the inspector, the "Rigidbody" component will appear





Movement (Script)

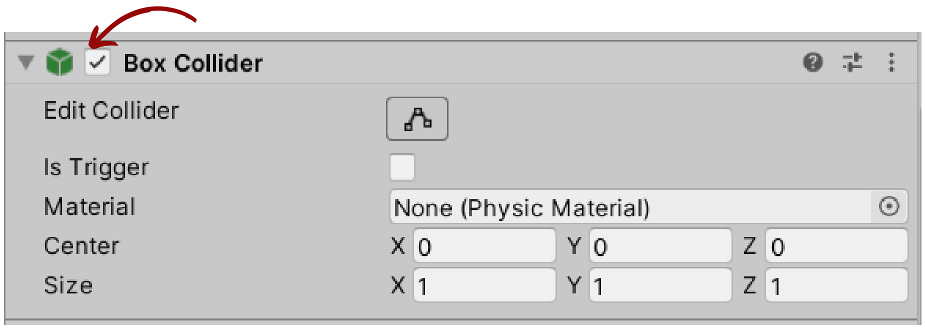
Script	Movement

Rigidbody

Mass	1
Drag	0
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	
Interpolate	None
Collision Detection	Discrete
Constraints	
Info	

Step2 : Make sure that your objects have colliders, and the "collider" component is checked. If not, then add the "collider" component using the following instructions :

- Click on the object
- Go to the Inspector
- Scroll down to the end of the Inspector
- Click on "Add Component"
- Search for "collider" and click on it
- Then in the inspector, the "collider" component will appear



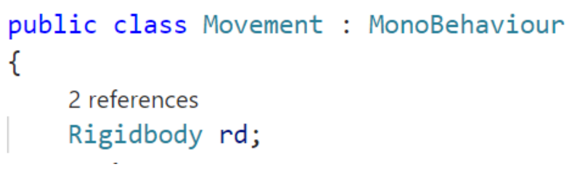
✓ Box Collider

Edit Collider						
Is Trigger						
Material	None (Physic Material)					
Center	X	0	Y	0	Z	0
Size	X	1	Y	1	Z	1

Rocket Vertical Movement

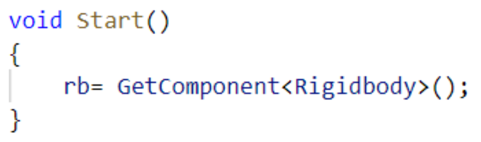
Step 1: Access the rigidbody component of the rocket

- Open Movement script
- declare the rigidbody variable of the rocket as rd



```
public class Movement : MonoBehaviour
{
    2 references
    Rigidbody rd;
```

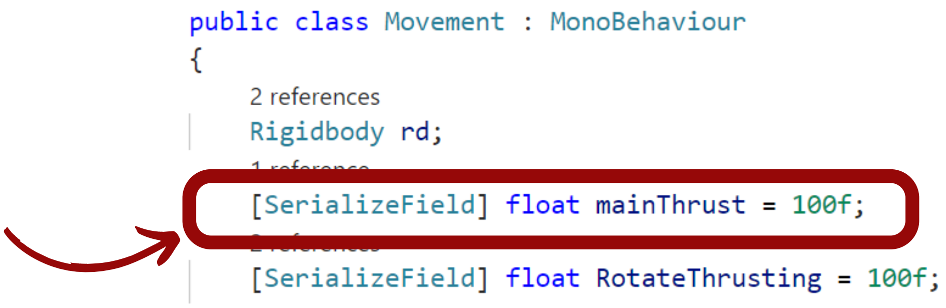
- access and save the rigid component of the rocket in the "rd" variable in start function



```
void Start()
{
    rb= GetComponent<Rigidbody>();
}
```

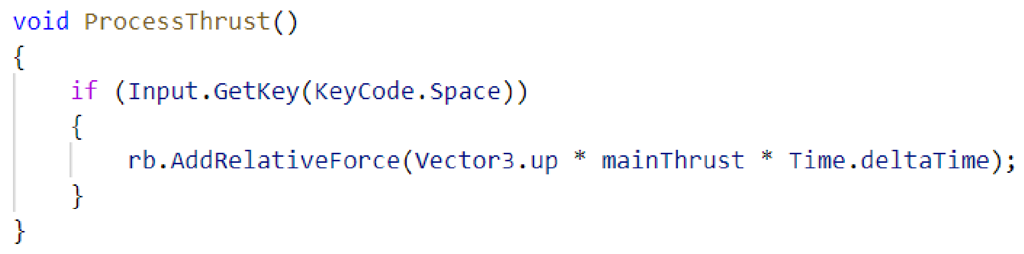
Step2 :Create a function for the rocket movement named "ProcessThrust"

- Create a variable to control the vertical force as the following, before start method



```
public class Movement : MonoBehaviour
{
    2 references
    Rigidbody rd;
    1 reference
    [SerializeField] float mainThrust = 100f;
    2 references
    [SerializeField] float RotateThrusting = 100f;
```

- After update function within the movement script class write the definition of the function as follows



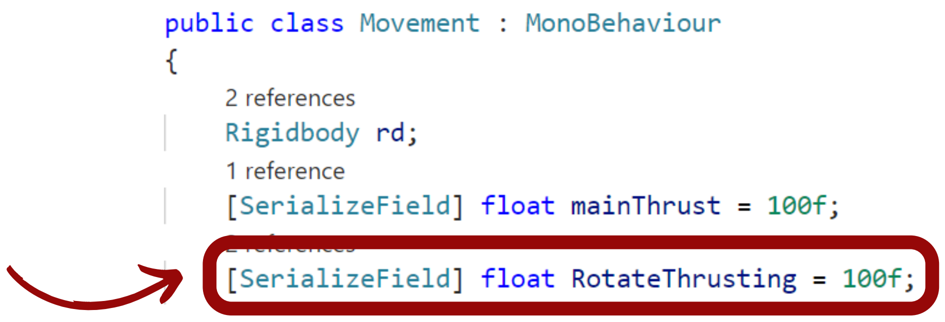
```
void ProcessThrust()
{
    if (Input.GetKey(KeyCode.Space))
    {
        rb.AddRelativeForce(Vector3.up * mainThrust * Time.deltaTime);
    }
}
```

Note: The movement is made time-frame independent by multiplying it with Time.deltaTime

Rocket Rotation

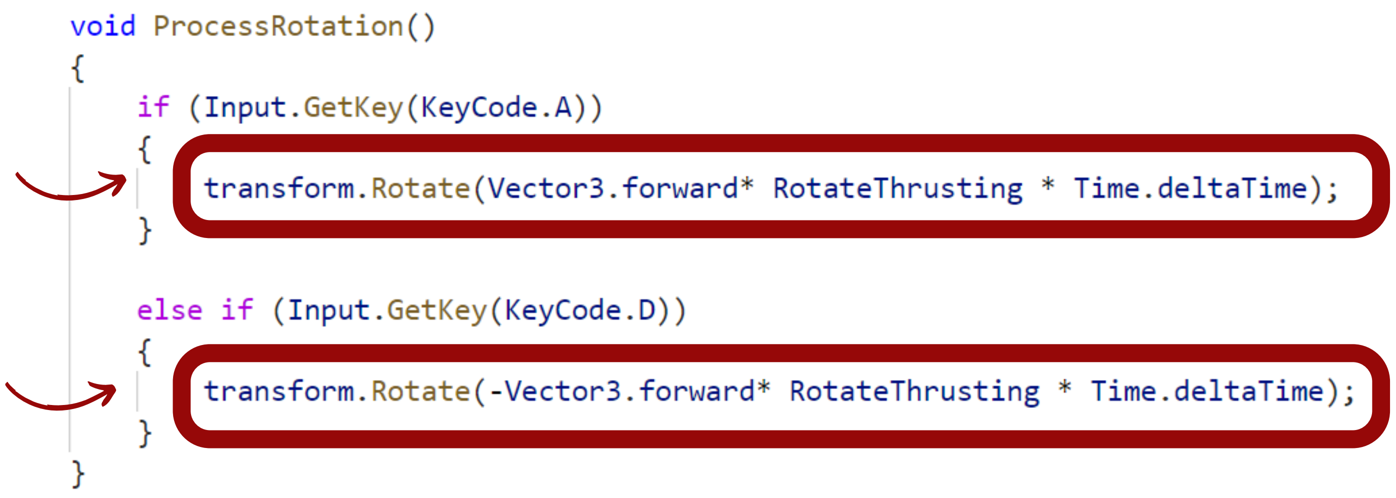
Step 1: Create a serialized "RotationThrusting" variable, before the "Start" function

- Open "Movement" Script
- Write the following code before the "Start" fucntion



```
public class Movement : MonoBehaviour
{
    2 references
    Rigidbody rd;
    1 reference
    [SerializeField] float mainThrust = 100f;
    2 references
    [SerializeField] float RotateThrusting = 100f;
```

Step 2: Update the "ProcessRotation" function as follows



```
void ProcessRotation()
{
    if (Input.GetKey(KeyCode.A))
    {
        transform.Rotate(Vector3.forward* RotateThrusting * Time.deltaTime);
    }

    else if (Input.GetKey(KeyCode.D))
    {
        transform.Rotate(-Vector3.forward* RotateThrusting * Time.deltaTime);
    }
}
```