Game Programming

Lecture II

Izzet Fatih Senturk

Shooting

- Create a projectile when ”space key” is pressed

Create a Laser Prefab

- Create a new game object: “Capsule”
  - Set the position 0, 0, 0
  - Scale down to 0.2, 0.2, 0.2
  - Rename the capsule to “Laser”
- Make the capsule prefab and instantiate it at runtime
  - Create a new folder “Prefabs” under “Assets”
  - Drag the “Laser” into the “Prefabs” folder
- Prefabs are shared objects
  - Not unique
  - Duplicate the “Laser” and add “RigidBody” to one of them!
  - Use override to apply the change to all objects
  - Remove “RigidBody” and the duplicate “Laser”

Laser Material

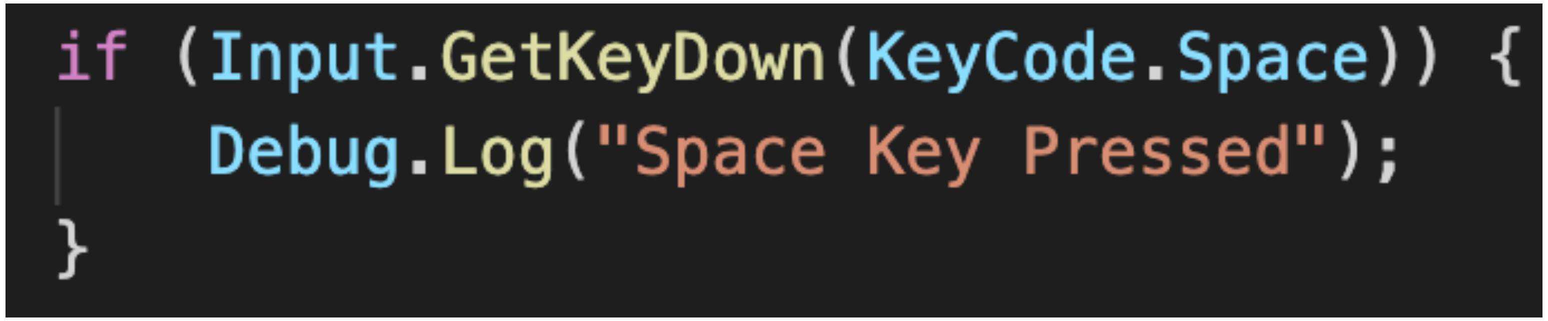
- Create a new material for the laser
  - Create a new material named “Laser_mat” under “Materials” folder
  - Change the color to light blue
- Apply the material to the “Laser”
  - Either drag it on the object and use override
  - Or drag it on the prefab

Instantiate Laser Behavior

- Instantiate the “Laser” object
  - When the user presses the “Space” key
- Then move the object up

Debug Input

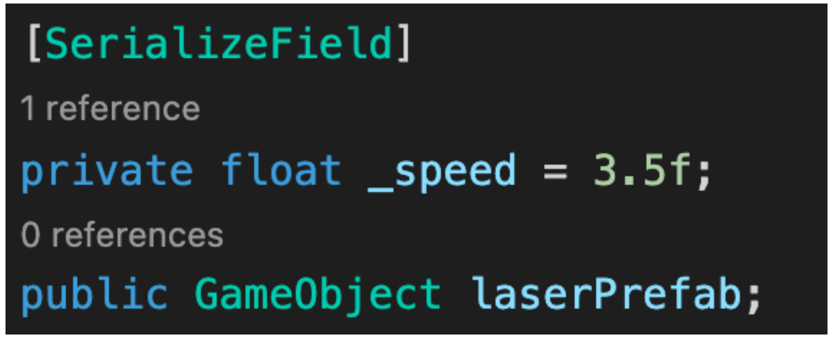
- Check if the “Space Key” is pressed
  - Print a message



```
if (Input.GetKeyDown(KeyCode.Space)) {
    Debug.Log("Space Key Pressed");
}
```

Instantiate Laser

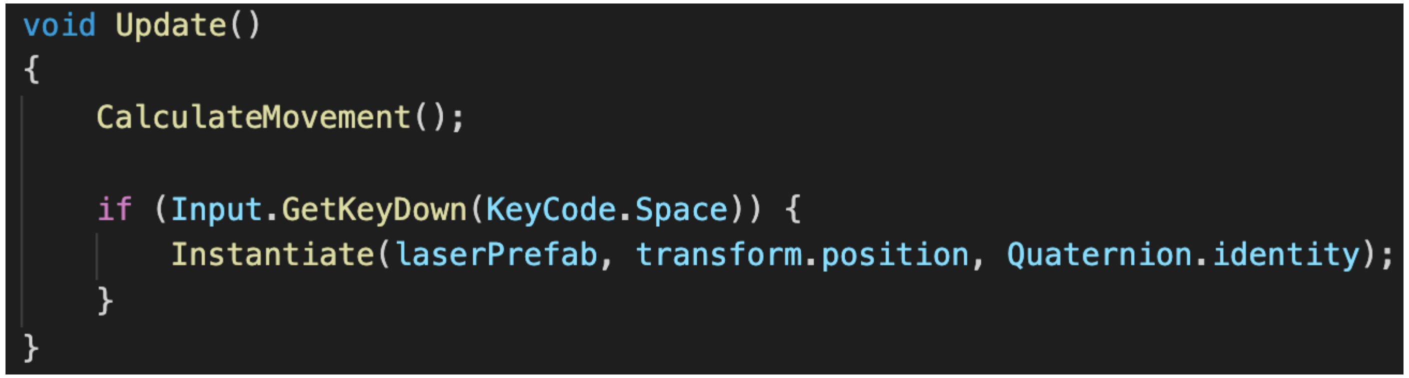
- Define a public variable for the “Laser” prefab in the “Player” script
- Remove the “Laser” object from the Hierarchy window
  - Always use prefabs to instantiate objects!
- Associate the “Laser” prefab with the public variable
- Change the access specifier of the “Laser” prefab variable
  - Make it private, change the name and add “SerializeField”



```
[SerializeField]
1 reference
private float _speed = 3.5f;
0 references
public GameObject laserPrefab;
```

Instantiate Laser

- Instantiate the "Laser" object from the prefab
    - Specify the prefab to instantiate: "Laser" prefab
    - Specify its position: current position
    - Specify the rotation: default rotation

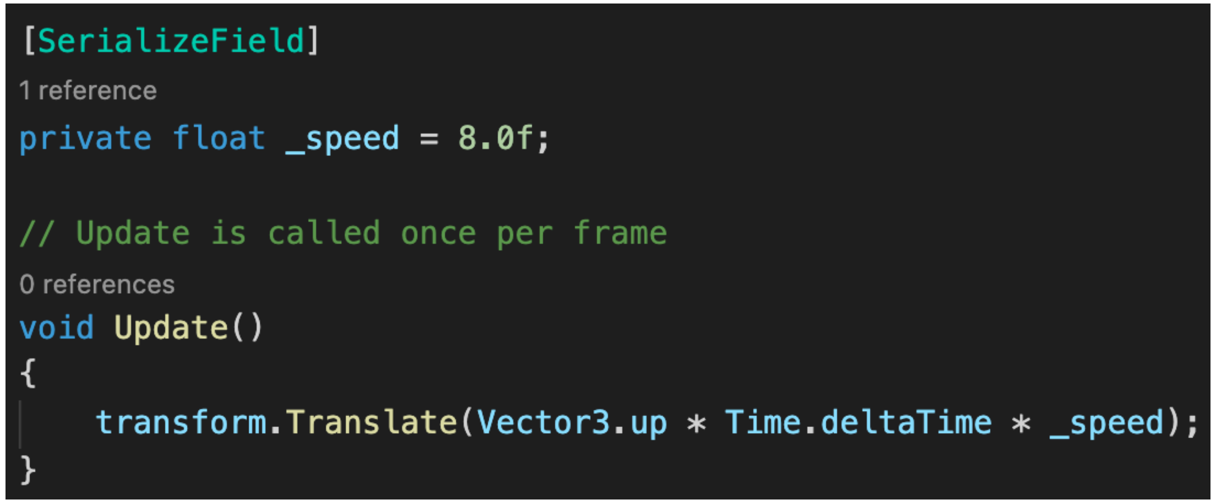


```
void Update()
{
    CalculateMovement();

    if (Input.GetKeyDown(KeyCode.Space)) {
        Instantiate(laserPrefab, transform.position, Quaternion.identity);
    }
}
```

Laser Behavior

- As soon as it is instantiated, ”Laser” should move up
- Create a new C# script: “Laser_sc”
  - Apply it to the “Laser” prefab
- Move up infinitely
  - Define a speed variable
  - Translate the object

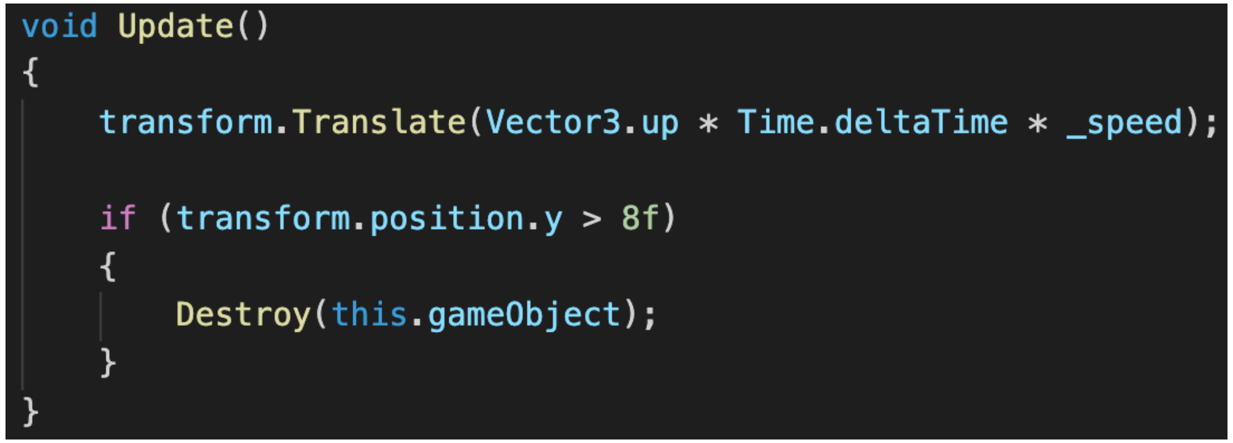


```
[SerializeField]
1 reference
private float _speed = 8.0f;

// Update is called once per frame
0 references
void Update()
{
    transform.Translate(Vector3.up * Time.deltaTime * _speed);
}
```

Destroy Laser

- Notice the populated “Laser” objects in the Hierarch window
- Destroy the “Laser” when it goes out of the screen
  - Check the y position when it goes off the screen

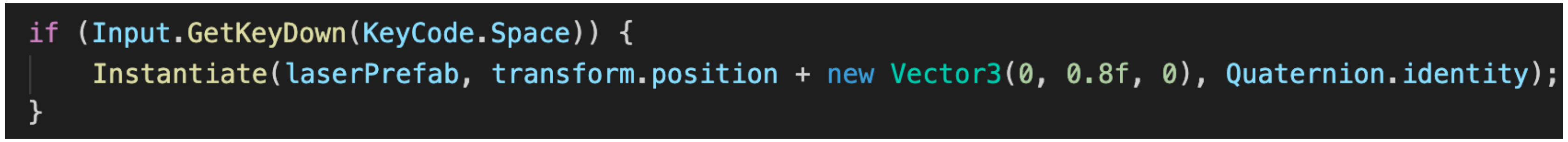


```
void Update()
{
    transform.Translate(Vector3.up * Time.deltaTime * _speed);

    if (transform.position.y > 8f)
    {
        Destroy(this.gameObject);
    }
}
```

Laser Position Offset

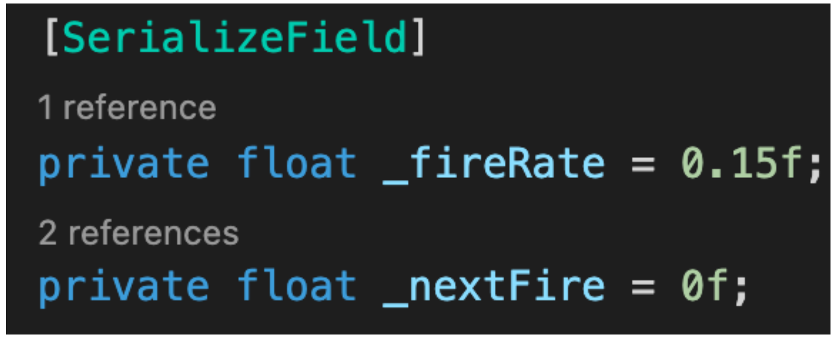
- Change the spawn position of the "Laser"
  - 0.8 units on the y



```
if (Input.GetKeyDown(KeyCode.Space)) {
    Instantiate(laserPrefab, transform.position + new Vector3(0, 0.8f, 0), Quaternion.identity);
}
```

Cool Down System

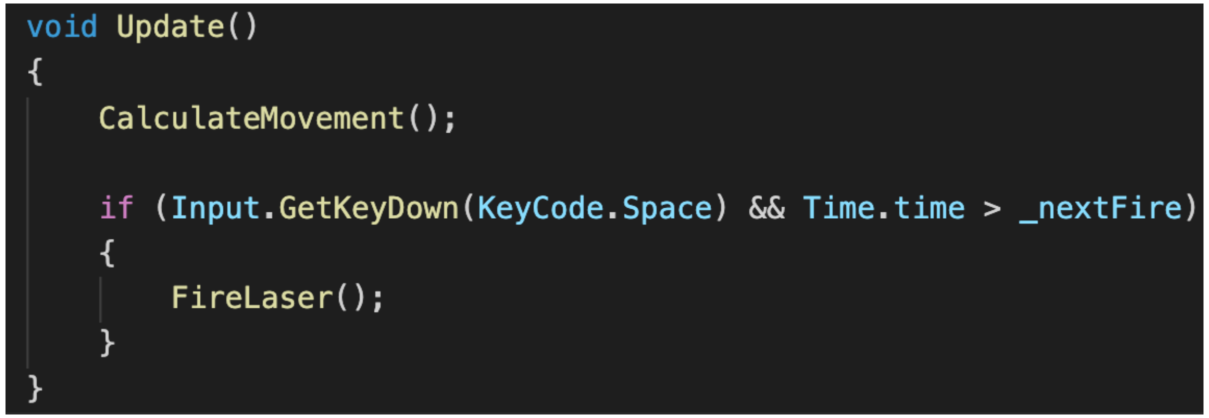
- Currently, we can fire infinitely
- Add a fire delay
  - Define a cool down delay
  - During the delay, we should not be able to fire



```
[SerializeField]
1 reference
private float _fireRate = 0.15f;
2 references
private float _nextFire = 0f;
```

Code Improvement

- Create a new method for laser fire: “FireLaser”



```
void Update()
{
    CalculateMovement();

    if (Input.GetKeyDown(KeyCode.Space) && Time.time > _nextFire)
    {
        FireLaser();
    }
}
```