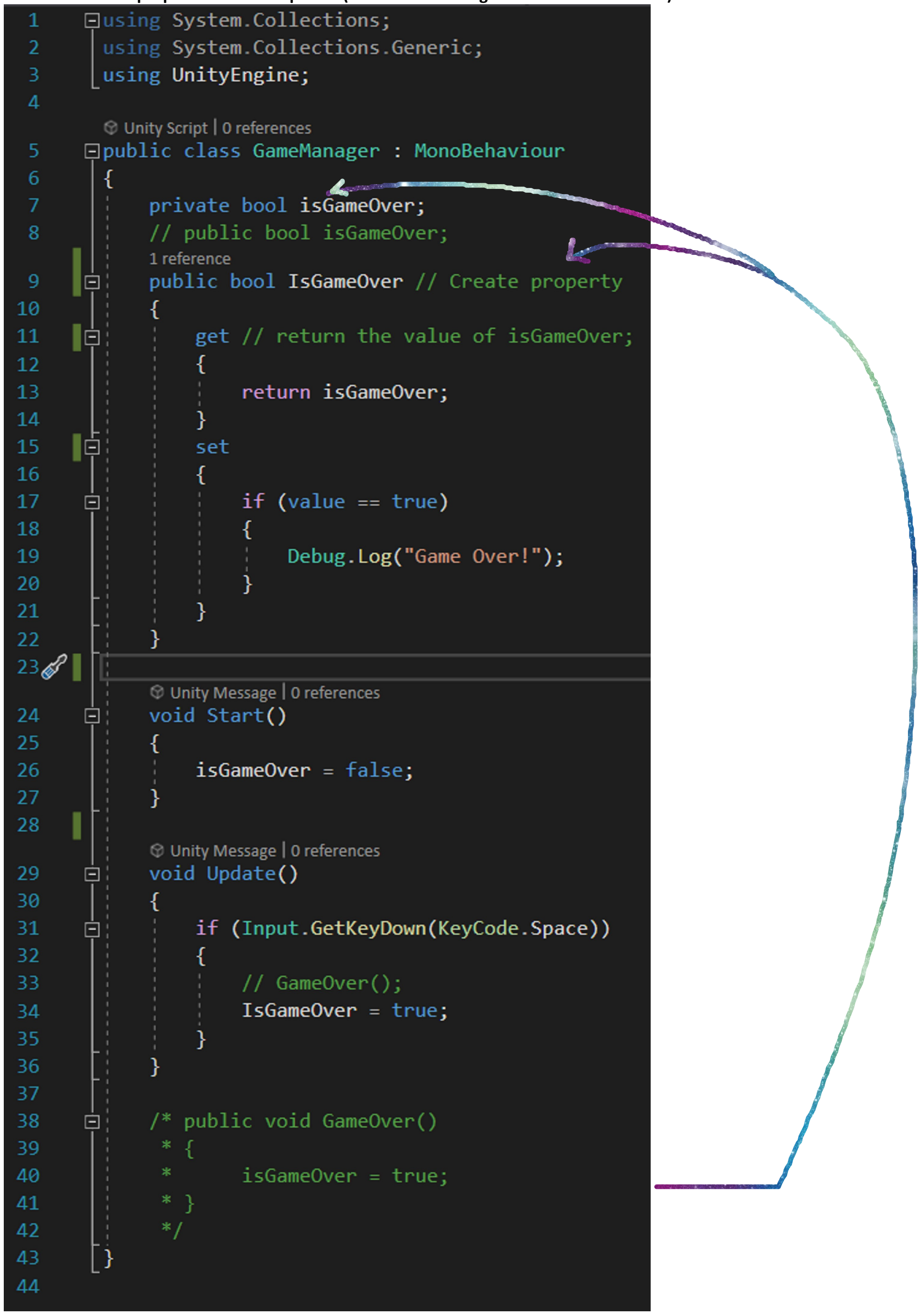
Properties

Tuesday, January 5, 2021 12:47 AM

Safer way to access variables rather than making them public.

- Think of them as smart variables.
- Not only can they retrieve information from them, but you can also run functionality through them.
- "**getters**" and "**setters**" **{get; set;)**
- '**Setter**' - '=' to assign a value to it
- '**Getter**' - retrieve the value
- '**Value**' - automatically interprets value assigned to it.
- **Cannot see properties in the Inspector (visible with Debug Mode or custom Editor)**

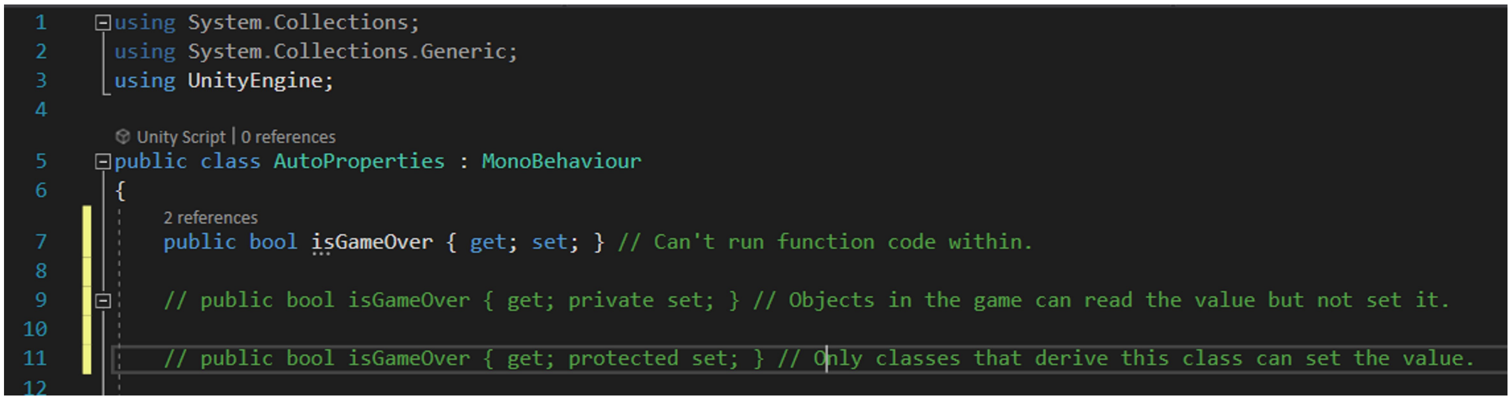


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class GameManager : MonoBehaviour
{
    private bool isGameOver;
    // public bool isGameOver;
    1 reference
    public bool IsGameOver // Create property
    {
        get // return the value of isGameOver;
        {
            return isGameOver;
        }
        set
        {
            if (value == true)
            {
                Debug.Log("Game Over!");
            }
        }
    }

    Unity Message | 0 references
    void Start()
    {
        isGameOver = false;
    }

    Unity Message | 0 references
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            // GameOver();
            IsGameOver = true;
        }
    }

    /* public void GameOver()
     * {
     *      isGameOver = true;
     * }
     */
}
```

Auto Properties

When working with properties where you set the get & set manually, have a variable to work with it.

An auto property is declared like a normal variable.



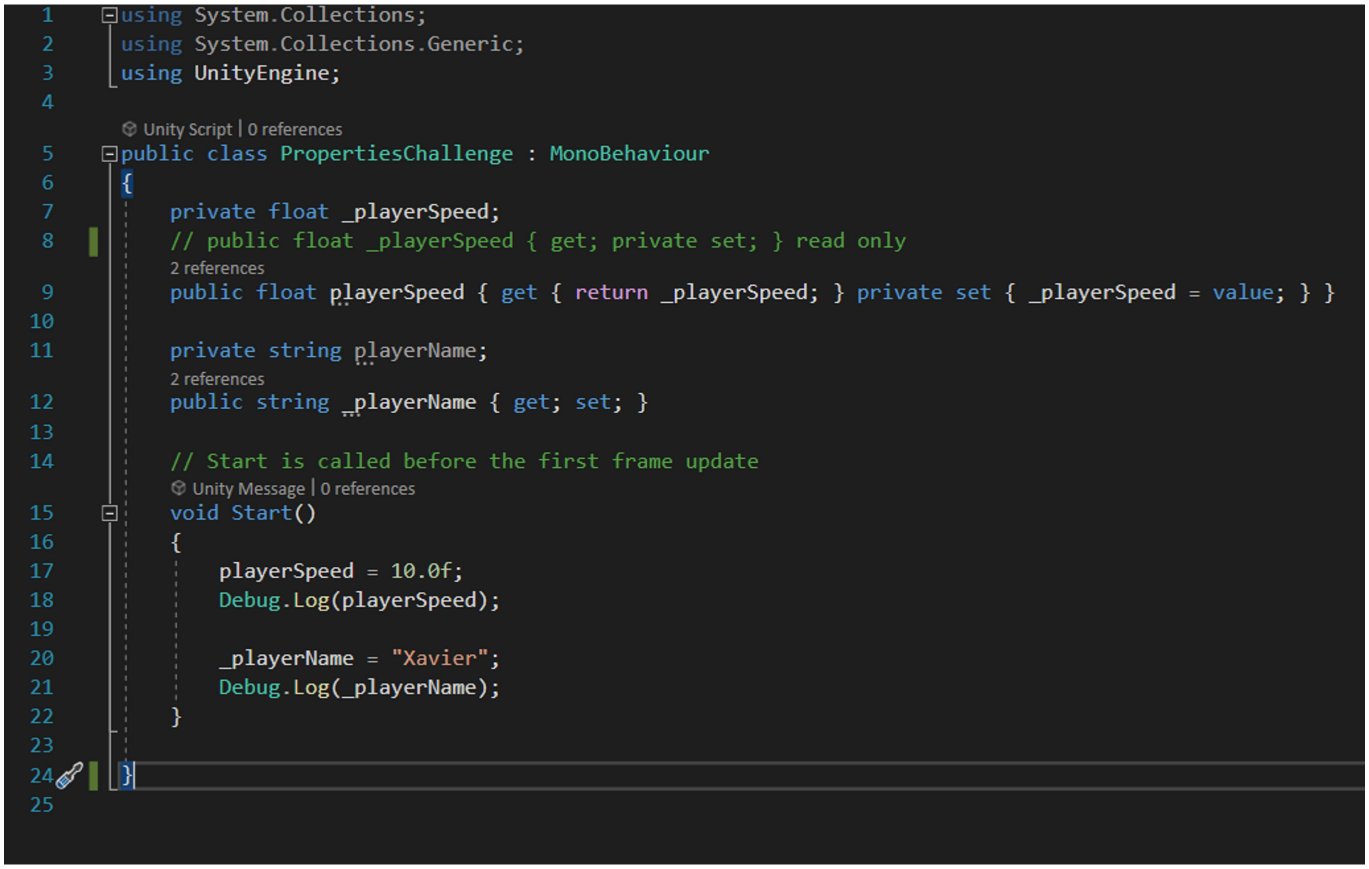
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class AutoProperties : MonoBehaviour
{
    2 references
    public bool isGameOver { get; set; } // Can't run function code within.

    // public bool isGameOver { get; private set; } // Objects in the game can read the value but not set it.

    // public bool isGameOver { get; protected set; } // Only classes that derive this class can set the value.
```

When to Use

Typically used in manager classes.
Doesn't allow initial assigning. public
int myAge = 26; vs public int **MyAge** {get;set;} void Start { **MyAge** = 26; }

Create the following Properties Example

Speed (read only) & Name (public)



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class PropertiesChallenge : MonoBehaviour
{
    private float _playerSpeed;
    // public float _playerSpeed { get; private set; } read only
    2 references
    public float playerSpeed { get { return _playerSpeed; } private set { _playerSpeed = value; } }

    private string playerName;
    2 references
    public string _playerName { get; set; }

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        playerSpeed = 10.0f;
        Debug.Log(playerSpeed);

        _playerName = "Xavier";
        Debug.Log(_playerName);
    }

}
```