Computer Graphics
- Game Programming2

Sung Soo Hwang

Practice: C# Programming

- Enumerations
  - An enumeration type (or enum type) is a value type defined by a set of named integer constants.

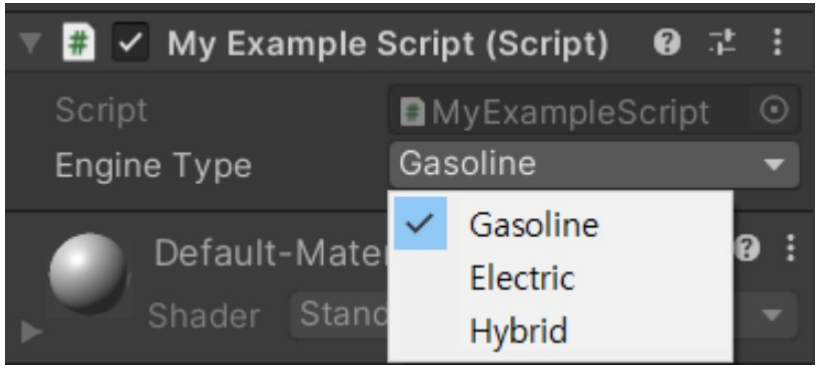
```
enum EngineType
{
    gasoline, electric, hybrid
}

static void Main(string[] args)
{
    EngineType engine = EngineType.electric;
    var type = EngineType.hybrid;
    type = EngineType.gasoline;

    EngineType anotherEngine = EngineType.hybrid;
    anotherEngine = EngineType.gasoline;
}
```

Practice: C# Programming

- Practice
  - In a Unity script, define an enum type as a public type and declare a public object variable in that enum type as follows:

```
public enum EnginType
{
    gasoline, electric, hybrid
}

public EnginType engineType = EnginType.gasoline;
```

  - Then, see the interface corresponding to that variable in the Inspector panel:



Practice: C# Programming

- Arrays
  - An array stores a fixed-size sequential collection of elements of the same type.

```
string[] names = new string[3];

string[] names = new string[3] { "Chris", "Vicki", "Roel" };

        names[0] = "Andrea";

        names[1] // is Vicki
```

Practice: C# Programming

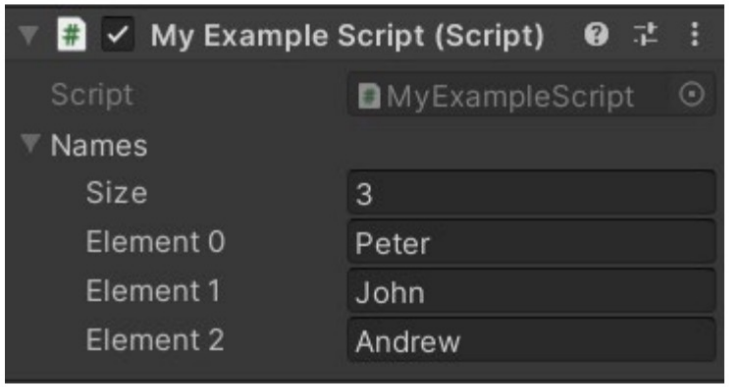
```
static void Main(string[] args)
{
    int[] scores = new int[3];
    scores[0] = 60;
    scores[1] = 32;
    scores[2] = 66;

    var highScores = new int[]
    {
        130, 654, 234
    };

    var moreHighScores = new int[]
    {
        highScores[0], highScores[1], highScores[2], 549, 393, 654
    };

    int[,] playerScores = new int[,]
    {
        {30, 50, 10},
        {42, 12, 60}
    };
    Console.WriteLine(playerScores[0, 1]);
}
```

two-dimensional array

Practice: C# Programming

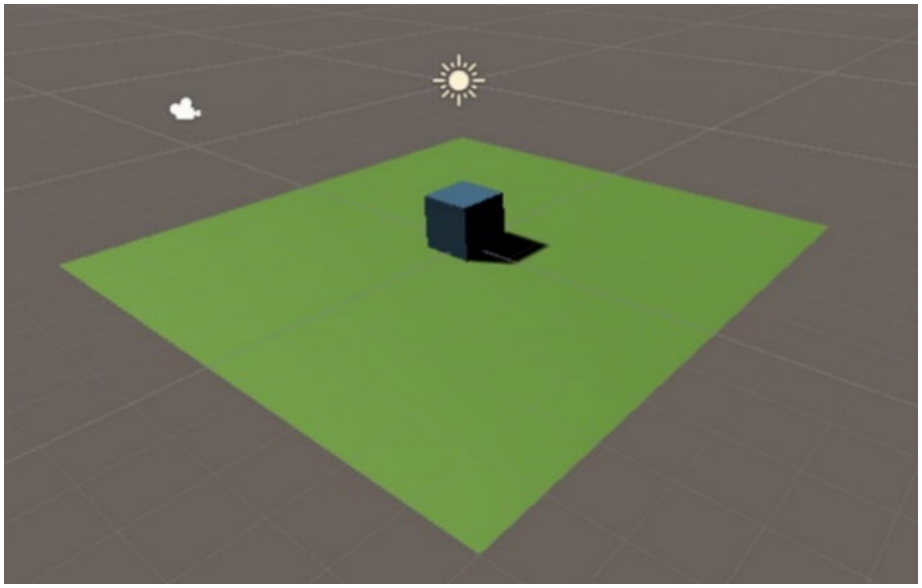
- Practice
  - Make a string array as a public object variable and see the resulting interface in the Inspector panel.

```
public string[] names = new string[3] {"Peter", "John", "Andrew"};
```



Practice: C# Programming

- A Simple Interactive cube
  - Reload the following scene in Unity.

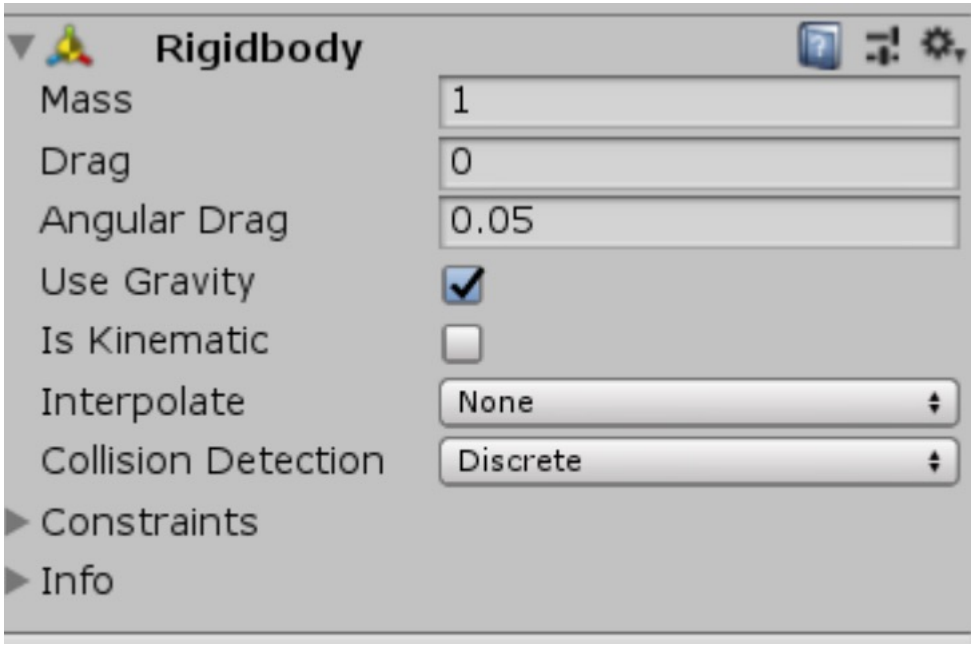


  - Let's add some Physics to the object in the next slide

Practice: C# Programming

- Select the cube and choose Component → Physics → Rigid Body. Then, you will see a new rigid body component in the inspector panel.

*Adding a Rigidbody component to an obj ect will put its motion under the control of Unity's physics engine



- Now modify the existing C# script as given in the next slide.

Practice: C# Programming

```
void Update()
{
    var rigidBody = GetComponent<Rigidbody>();

    if (Input.GetKeyUp(KeyCode.LeftArrow) || Input.GetKeyUp(KeyCode.A))
    {
        rigidBody.AddForce(-100, 0, 0);
    }
    if (Input.GetKeyUp(KeyCode.RightArrow) || Input.GetKeyUp(KeyCode.D))
    {
        rigidBody.AddForce(+100, 0, 0);
    }
    if (Input.GetKeyUp(KeyCode.UpArrow) || Input.GetKeyUp(KeyCode.W))
    {
        rigidBody.AddForce(0, 0, +100);
    }
    if (Input.GetKeyUp(KeyCode.DownArrow) || Input.GetKeyUp(KeyCode.S))
    {
        rigidBody.AddForce(0, 0, -100);
    }
    if (Input.GetKeyUp(KeyCode.Space))
    {
        if(Input.GetKey(KeyCode.LeftShift))
        {
            transform.position = new Vector3(0, 0.5f, 0);
            transform.rotation = Quaternion.Euler(0, 0, 0);
            rigidBody.velocity = Vector3.zero;
            rigidBody.angularVelocity = Vector3.zero;
        }
        else
        {
            rigidBody.AddForce(0, +100, 0);
        }
    }
}
```

The GetComponent generic method returns a component of type T in a game object to which this script is attached.

AddForce(x, y, z) applies a linear force by x, y, and z along the x-axis, y-axis, and z-axis, respectively.

Initialize the state of the game object.