SE2250 – Software Construction

Lecture #4

C# and Unity

January 28, 2019

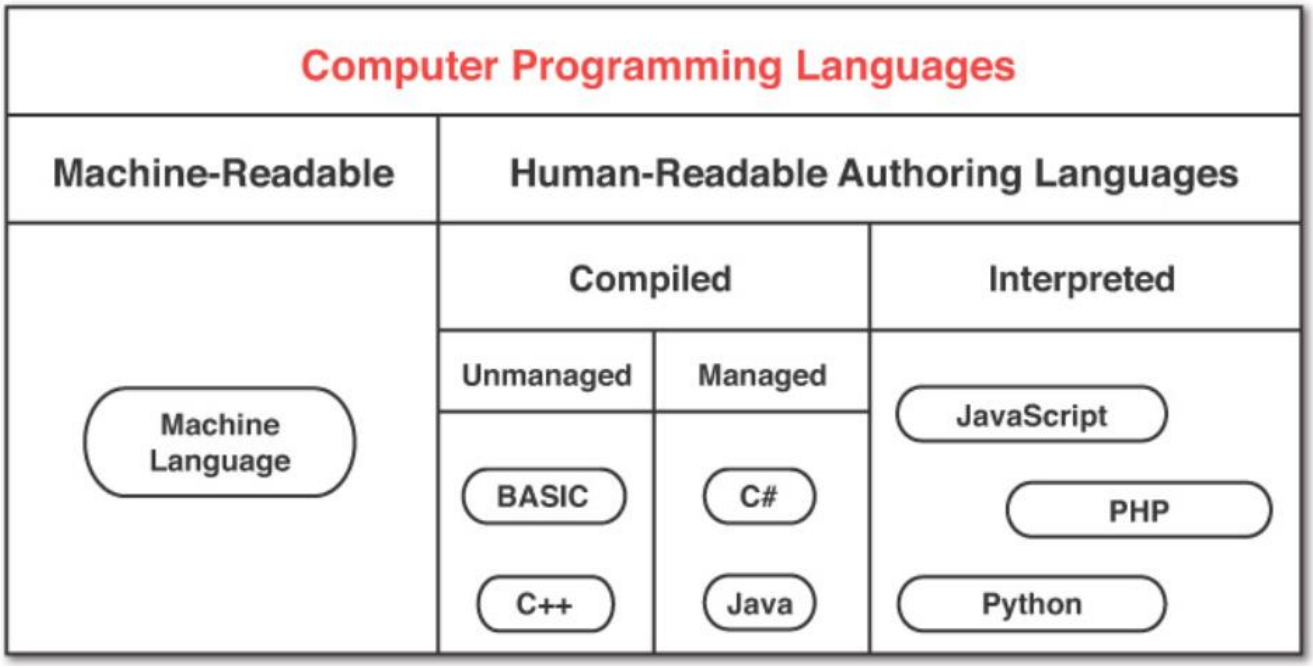
Outline

- C# features
- C# in Unity
- C# in Unity – Collections

C#

- Compiled language (in contrast to interpreted)
  - *An interpreted language* - most of its implementations execute instructions directly, without previously compiling a program into machine-language instructions
  - *A compiled language* - most of its implementations convert the code from the authoring language to a compiled application
  - Compiled languages – often suited for specific execution platforms (objective-C)

C#


Computer Programming Languages
Machine-Readable
Human-Readable Authoring Languages
Compiled
Interpreted
Unmanaged
Managed
Machine Language
BASIC
C++
C#
Java
JavaScript
PHP
Python

C#

- Managed code
  - allocation and de-allocation of memory is handled automatically
- Strongly Typed (in contrast weekly typed)
  - When a variable is declared, type must be specified

```
int x= 5;
```

  - enables the compiler to make optimizations
  - makes possible real-time syntax checking
  - enhances code-completion

C# in Unity

- Variables

- Unity uses floats
- Always append **f** to a decimal number

- Char ' '
- String " "

- Some come with a default, but always initialize

```
bool   bravo;
int    count;
float  val;
string str = "test"
char   c = 'c';
…
val=1.3f;
```

C# in Unity

- Naming Conventions

- CamelCase:
  - `aVeryLongNameThatIsEasierToReadBecauseOfCamelCase`
  - `variableNamesStartWithALowerCaseLetter`
  - `ClassNamesStartWithACapitalLetter`

C# in Unity

- Naming Conventions:
  - Variable names start with a lowercase letter
    - `someVariableName`
  - Function names start with an uppercase letter
    - `Start(), Update()`
  - Class names start with an uppercase letter
    - `GameObject, ScopeExample`
  - Private variable names often start with an underscore
    - `_hiddenVariable`
  - Static variable names are often all caps with snake_case
    - `NUM_INSTANCES`

C# in Unity

- Instance variables and functions
  - Tied directly to a single instance of the variable type
  - Each instance can have different value
  - Instance variables referred to as ***fields***
  - Instance functions are referred to as ***methods***.

```
Vector3 position = new Vector3 ( 0.0f ,3.0f ,4.0f );
```

C# in Unity

- Static class variables and functions
  - Tied to the class definition itself rather than being tied to an individual instance
  - Store information that is the same across all instances of the class

C# in Unity

- Example from Unity lecture - Move the cube from the script

```
12      // Update is called once per frame
13      void Update () {
14          Vector3 move =Vector3.zero;
15          float speed = 0.01f;
16          if (Input.GetKey (KeyCode.RightArrow)) {
17              move = Vector3.right;
18          } else if (Input.GetKey (KeyCode.LeftArrow)) {
19              move = Vector3.left;
20          }
21          gameObject.transform.position = move*speed
22              + gameObject.transform.position;
23      }
24  }
25
```

C# in Unity

- Class instances are referred to by reference, not value
- If you are comparing two class instances to see whether they are the same, the thing that is compared is their location in memory, not their values

```
public class Human {
    public string  name;
    public Human partner;
}
```

C# in Unity

- Vector3 – collection of 3 floats

```
Vector3 position = new Vector3 (0.0f, 3.0f, 4.0f);
position.x
position.magnitude
Vector3.zero
Vector3.up
Vector3.Cross(v3a, v3b)
…
```

C# in Unity

- Colour with transparency

```
Color darkRedTranslucent = new Color(0.25f, 0f, 0f, 0.5f);
Color.red;
Color.gray;
```

- Rotation

```
transform.Rotate(Vector3.right * Time.deltaTime);
(public void Rotate(Vector3 eulers, Space relativeTo = Space.Self))

Quaternion lookUp45Deg = Quaternion.Euler(-45f, 0f, 0f);
transform.rotation = Quaternion.Euler(-45f,0,0);
```

C# in Unity

- Math functions
  - Mathf library

```
Mathf.Sin(x)
Math.PI
Mathf.Min(2, 3, 1)
```

- Random

```
Random.value
Random.insideUnitCircle
Random.onUnitSphere
Random.insideUnitSphere
```

C# in Unity

- Comparison
  - Simple data types by value (bool, int, float, char, string, Vector3, Color, Quaternion)
  - Others by reference

C# in Unity

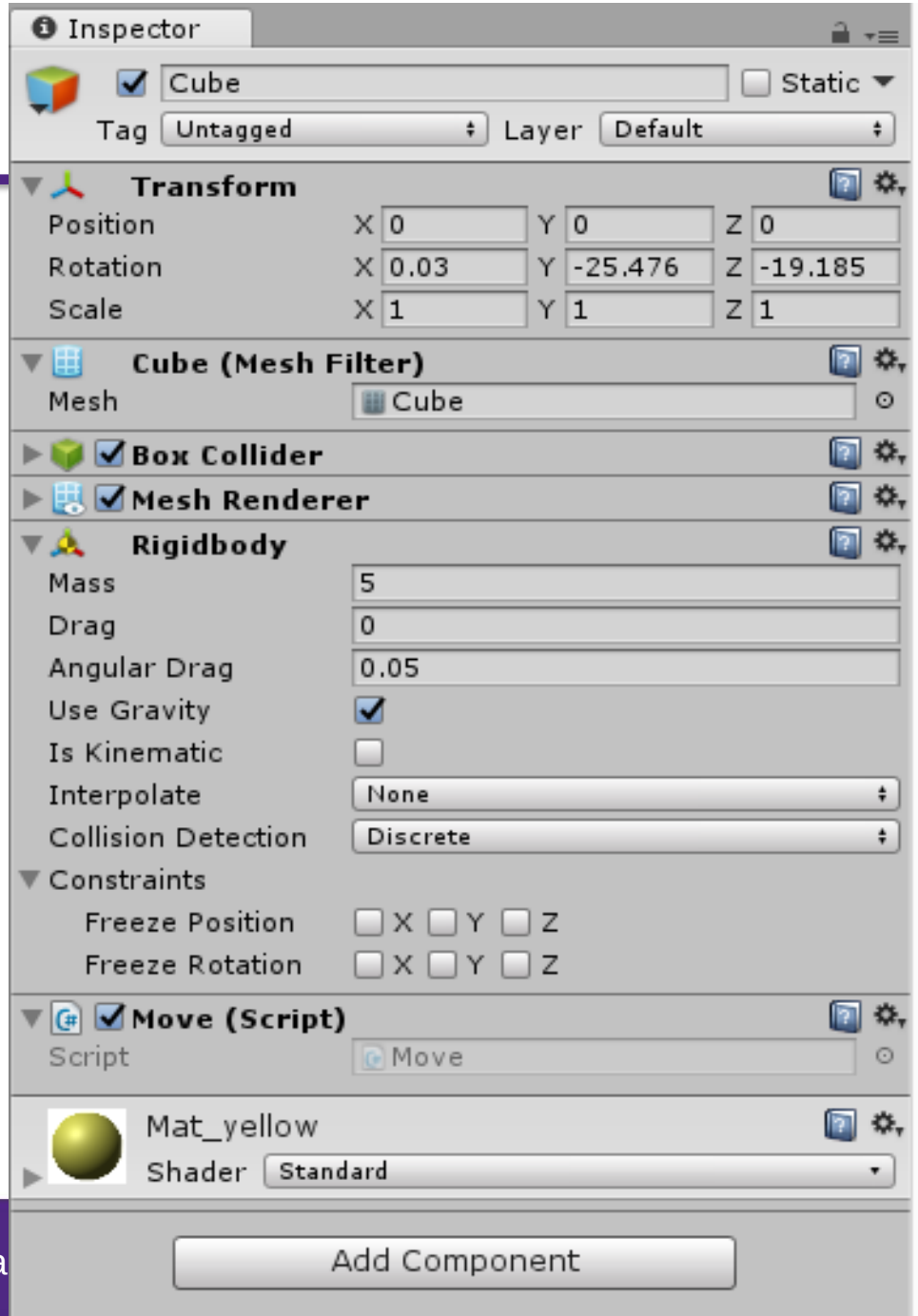
- GameObject – the base for all entities in Unity screen
- GemeObjects contain components:
  - Transform – position, rotation, scale
  - Collider – physical presence, collide with other objects
  - Rigid body – physical simulation
    - Gravity, drag, forces
    - isKinematic = true - Sets the position without using the physics
    - isTrigger = true - A trigger doesn't collide with rigid bodies (OnTrigger events)

C# in Unity

- GameObject
  - Components

```
gameObject.tag;
gameObject.transform.position.x;

…
Rigidbody rb
  = gameObject.GetComponent
   <Rigidbody>();
print (rb.mass);
```



C# in Unity

- Boolean Operators

- ! Not operator
- && AND operator
- || OR operator

- shorting operators - after the operator has determined the return value, it returns that value without executing the rest of the code.

C# in Unity

- Boolean Operators

```
bool printAndReturnTrue () {
    print( "--true" );
    return( true );
}

bool printAndReturnFalse() {
    print( "--false" );
    return( false );
}

void ShortingOperatorTest() {
    bool andTF = ( printAndReturnTrue() && printAndReturnFalse() );
    print( "andTF: "+andTF );
    bool andFT = ( printAndReturnFalse() && printAndReturnTrue() );
    print( "andFT: "+andFT );
}
```

C# in Unity

- If …else if…else

```
if (!night) {                    //Condition 1 evaluates to false
    print("It's daytime.");      //Condition 2 evaluates to true
} else if (fullMoon) {
    print("Beware werewolves!!!");
} else {                         //Condition 3 not evaluated
   print("It's night, but the moon is not full.");
}
```

C# in Unity

- Switch
  - Does not support fall-through from one case label to another one (unless first one is empty)

```
int num = 3;
switch (num)
{ // The variable in parentheses (num) is the one being compared
    case (0): // Each case is a literal number that is compared against num
        print("The number is zero.");
        break; // Each case must end with a break statement.
    case (1):
        print("The number is one.");
        break;
    default: // If none of the other cases are true, default will happen
        print("The number is more than a couple.");
        break;
} // The switch statement ends with a closing brace.
```

C# in Unity

- Loops
  - `while, do…while, for`

```
int i = 0;
while (i<3)
{
    print("Loop: " + i);
    i++;
}
```

```
int i = 0;
do
{
    print("Loop: " + i);
    i++;
} while (i < 3);
```

```
for (int i=0;i<3; i++)
{
    print("Loop: " + i);
}
```

C# in Unity

- Loops
  - `while, do…while, for`
  - `foreach`
    - iterate through collections
  - `continue` and `break`

C# in Unity - Collections

- Arrays
- Lists
- Dictionaries
- Queues
- Stacks

C# in Unity - Collections

- Arrays
  - Indexed, ordered list of objects
  - Fixed length (can have empty)

```
string[] sArray;
sArray = new string[10];
sArray[0] = "These";
sArray[1] = "are";
sArray[2] = "words";
print( "The length of sArray is: "+sArray.Length );
string str = "";
foreach (string sTemp in sArray) {
    str += "|"+sTemp;
}
print(str);
```

C# in Unity - Collections

- Arrays
  - Indexed, ordered list of objects
  - Fixed length (can have empty)

…

```
foreach (string sTemp in sArray) {
    str += "|"+sTemp;
}
print (str);
```

With for loop instead?

C# in Unity - Collections

- Arrays

```
string[] myArray = new string[] { "A", "B", "B" };
//...
System.Array.IndexOf(myArray, "A");
System.Array.Resize(ref myArray, 7);
```

C# in Unity - Collections

- Multidimensional Arrays

```
string[,] sArray2d;
sArray2d = new string[4, 4];
sArray2d[0, 0] = "A";
sArray2d[0, 3] = "B";
sArray2d[1, 2] = "C";
sArray2d[3, 1] = "D";
print( "The Length of sArray2d is: "+sArray2d.Length );
```

- What will this print?

C# in Unity - Collections

- Lists (used a lot in Unity)
  - Variable length

```
List<string> sList;
sList = new List<string>();
sList.Add( "What" );
sList.Add( "is" );
sList.Add( "This?" );
print( "sList Count = "+sList.Count );
print( "The 0th element is: "+sList[0] );
print( "The 1st element is: "+sList[1] );
string str = "";
foreach (string sTemp in sList) {
    str += sTemp+" ";
}
```

C# in Unity - Collections

- Lists with game objects
- Example from Cube Spawner demo

```
public GameObject          cubePrefabVar;
public List<GameObject> gameObjectList;
public float               scalingFactor = 0.95f;
public int                 numCubes = 0;

void Start()
{
    gameObjectList = new List<GameObject>();
}
```

```
void Update()
{
    numCubes++;
    GameObject gObj = Instantiate<GameObject>(cubePrefabVar);
    gObj.name = "Cube " + numCubes;
    Color c = new Color(Random.value, Random.value, Random.value);
    gObj.GetComponent<Renderer>().material.color = c;
    gObj.transform.position = Random.insideUnitSphere;
    gameObjectList.Add(gObj);
    List<GameObject> removeList = new List<GameObject> ();
    foreach (GameObject goTemp in gameObjectList)  {
        float scale = goTemp.transform.localScale.x * scalingFactor;
        goTemp.transform.localScale = Vector3.one * scale;
        if (scale <= 0.1f) {
            removeList.Add(goTemp);
        }
    }
    foreach (GameObject goTemp in removeList)  {
        gameObjectList.Remove(goTemp);
        Destroy(goTemp);
    }
}
```

C# in Unity - Collections

- Lists with game objects
  - Demo – Cube spawner
  - What else to do before running?
  - How to fix the error?
  - Change the scaling factor without changing the script?

C# in Unity - Collections

- Array or List?
  - List has flexible length, whereas array length is more difficult to change.
  - Array is very slightly faster.
  - Array allows multidimensional indices.
  - Array allows empty elements in the middle of the collection.
  - Lists are a bit simpler to implement (especially for for prototyping)

C# in Unity - Collections

- Dictionaries
  - Key/value pairs, Constant access time

```
Dictionary<string, string> provDict;
provDict = new Dictionary<string, string>();
provDict.Add("ON", "Ontario");
provDict.Add("MB", "Manitoba");
print("There are " + provDict.Count + " elements in provDict.");
foreach (KeyValuePair<string, string> kvp in provDict)
{
    print(kvp.Key + ": " + kvp.Value);
}
print("ON is " + provDict["ON"]);
provDict["BC"] = "British Columbia";
foreach (string k in provDict.Keys)
{
    print(k + " is " + provDict[k]);
}
```

C# in Unity - Collections

- Queues
  - FIFO collection
  - `Deque()`
  - `Enqueue()`
- Stacks
  - FILO collection
  - `Pop()`
  - `Push()`