INTRODUCING OUR LANGUAGE: C#

Topics

- The Features of C#
  - C# is a Compiled Language
  - C# is Managed Code
  - C# is Strongly Typed
  - C# is Function Based
  - C# is Object-Oriented
- Reading and Understanding C# Syntax

Understanding the Features of C#

- **C# is a Compiled Language**
  - **Computer chips only understand *machine language***
    - 000000 00001 00010 00110 00000 100000
    - This is what punch cards were
  - ***Authoring languages* were created to be an intermediary language between humans and computers**
  - **Two kinds of authoring languages:**
    - *Interpreted*
    - *Compiled*

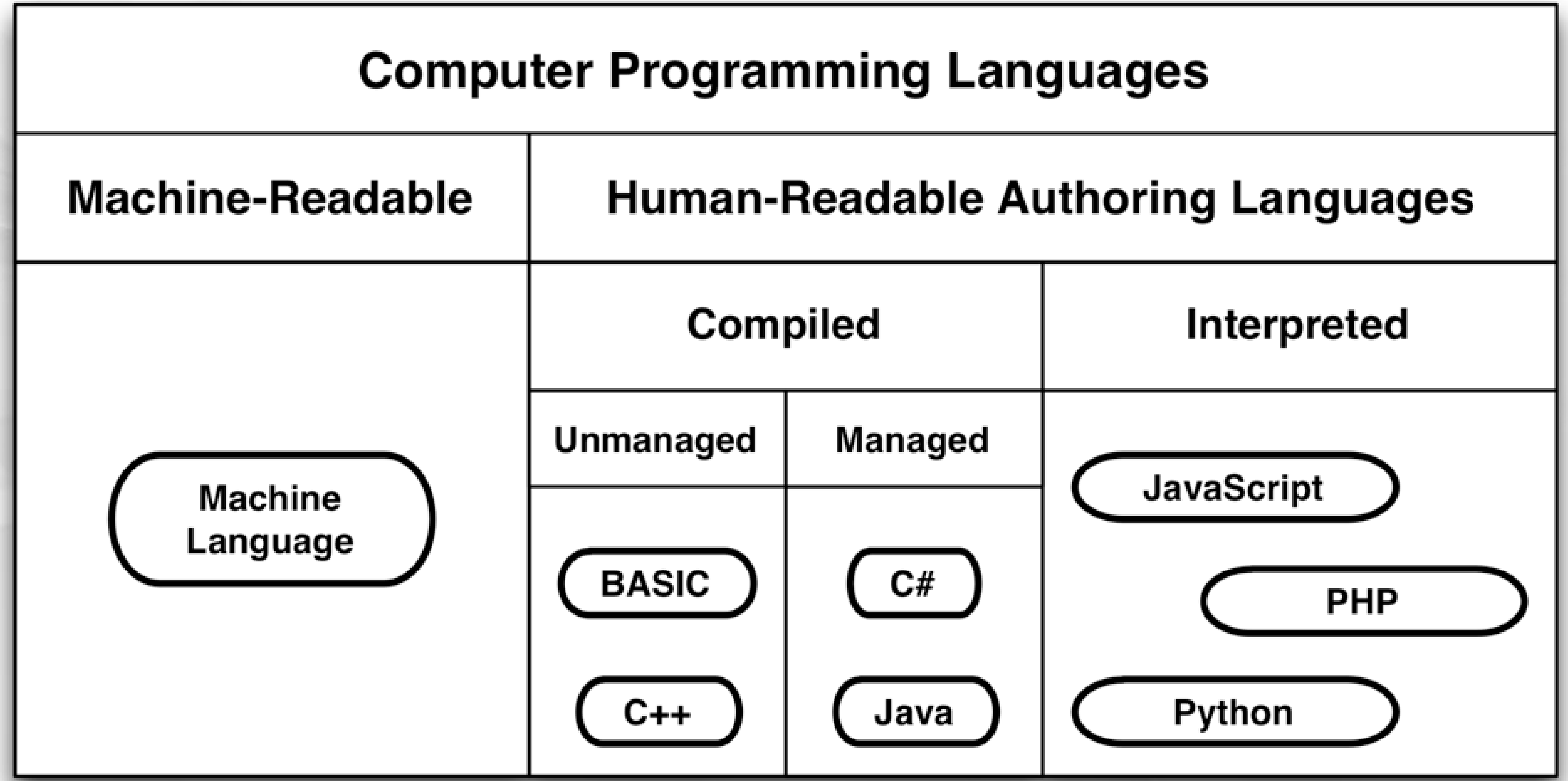
Understanding the Features of C#

- **Interpreted Languages**
  - **e.g., JavaScript, PHP, and Python**
  - **Two-Step Process**
    - Programmer writes the code
    - An *interpreter* converts the code into machine language in real-time
  - **Benefits**
    - **Portability**: Can run on any kind of computer as long as there's an interpreter
  - **Detriments**
    - **Lack of Speed**: The processing power used to interpret the code is not spent on the game itself
    - **Lack of Efficiency**: Because the code can run on any computer, it's not optimized for any specific computer

Understanding the Features of C#

- Compiled Languages
  - e.g., C#, Basic, Java, C++
  - Three-Step Process
    - Programmer writes the code
    - Programmer uses a compiler to convert the code into machine language
    - Computer executes the code
  - Benefits
    - **Speed**: Computer spends more processor power on the game itself
    - **Efficiency**: Code is optimized for a specific processor architecture
  - Detriments
    - **Lack of Portability**: Compiled for only one kind of machine
    - **Extra Compilation Step**

Understanding the Features of C#



Hierarchy of Computer Languages

Understanding the Features of C#

- **C# is *Managed Code***
  - **Computers have a limited amount of Random Access Memory (RAM)**
  - **Older compiled languages like BASIC and C++ require the programmer to manually allocate and deallocate RAM**
  - **In *managed code*, allocation and deallocation are handled automatically**
  - **This makes it less likely that you will accidentally claim all of the memory**
    - Doing so is known as a "memory leak"

VARIABLES IN COMPUTER LANGUAGES

- **A variable is a name that can hold a value**
- **This concept is borrowed from algrbra**
  - **x = 5**
  - **x + 2 = ?**
- **Variables in computer languages can hold much more than just simple numbers**
  - **Numbers**
  - **Words, sentences, paragraphs, novels…**
  - **Images, sounds, 3D models, animations…**
  - **Functions and methods**
  - **Classes and GameObjects**

Understanding the Features of C#

- C# is *Strongly Typed*
  - **In a non-strongly typed language, a variable can hold any kind of data**
    - The same variable could hold a number one moment and an animation the next
  - **A *strongly typed* language restricts the type of data that can be held by any variable**
    - **int x = 5;** – An int x can only hold an integer number
    - **float y = 3.4f;** – A float y can only hold a floating point number
  - **Strong typing allows accurate syntax checking**
    - The compiler can check your code for correctness
  - **Strong typing also allows robust code-completion**
    - The code editor can guess what you want to type and auto-complete

Understanding the Features of C#

- C# is *Function Based*
  - **Computer programs used to be just a series of commands**
  - **This was like giving driving directions to someone**
    1. From school, head north on Vermont
    2. Head west on I-10 for about 7.5 miles (about 12Km)
    3. At the intersection with I-405, take the 405 south for 2mi (3.2Km)
    4. Take the exit for Venice Blvd.
    5. Turn right onto Sawtelle Blvd.
    6. My place is just north of Venice on Sawtelle.
  - **Functional languages allow the encapsulation of commands**
    - "If you see a store on the way, please `BuySomeMilk()`."
    - The `BuySomeMilk()` function encapsulates the many actions involved in finding and purchasing milk.

Understanding the Features of C#

- C# is *Object-Oriented*
  - **Functions and data used to be separate**
  - **Object-oriented programming introduced *classes***
  - **Classes combine functions and data into a single *object***
    - Variables in classes are called *fields*
    - Functions in classes are called *methods*
  - **This enables things like a flock of birds where each bird thinks for itself…**
    - …rather than being controlled by a single, monolithic program
  - **Object-orientation also allows *class inheritance***
    - A subclass can inherit the fields and methods of its superclass
    - e.g., a `Dog` would inherit all the fields and methods of its superclass `Mammal`, which would in turn inherit from `Animal`

Reading and Understanding C# Syntax

- **All languages have *syntax*:**

The dog barked at the squirrel.	[Subject] [verb] [object].
At the squirrel the dog barked.	[Object] [subject] [verb].
The dog at the squirrel. barked	[Subject] [object]. [verb]
barked The dog at the squirrel.	[verb] [Subject] [object].

- **Only one of these sentences is correct**
  - **Only one follows the rules of syntax of the English language**
- **The other sentences have *syntax errors***

Reading and Understanding C# Syntax

- C# statements also have syntax rules
  - **int x = 5;**
    - *Declares* a variable named **x** of the type **int**
      - If a statement starts with a type, the second word of the statement becomes a new variable of that type
    - *Defines* the value of **x** to be **5**
      - The **=** is used to assign values to variables
    - All C# statements end with a semicolon (**;**)
      - A semicolon is used because the period is already used in decimal numbers

Reading and Understanding C# Syntax

- C# statements also have syntax rules
  - **int x = 5;**
  - **int y = x * (3 + x);**
    - Declares a variable named **y** of the type **int**
    - Adds **3 + x** for a value of **8** (because x = 5)
      - Just as in algebra, order of operations follows parentheses first
    - Multiplies **x * 8** for a value of **40** (5 * 8 = 40)
    - Defines the value of **y** to be **40**
    - Ends with a semicolon (**;**)

Reading and Understanding C# Syntax

- **C# statements also have syntax rules**
  - **string greeting = "Hello World!";**
    - Declares a variable named **greeting** of the type **string**
      - *strings* can hold a series of characters like a word or novel
    - Defines the value of **greeting** to be **"Hello World!"**
      - Anything between double quotes is a ***string literal***, a value to be assigned to a string variable
    - Ends with a semicolon (**;**)

Reading and Understanding C# Syntax

- C# statements also have syntax rules
  - **string greeting = "Hello World!";**
  - **print(greeting);**
    - *Calls* the function **print()**
      - When a function is called, it executes its actions
    - Passes the *argument* **greeting** into the function **print()**
      - Some functions take ***arguments***: data that changes how the function acts
      - The **print()** function will print the string **greeting** to the Console pane in Unity
      - The Console pane will display "Hello World!"
    - Ends with a semicolon (**;**)

Chapter 18 – Summary

- You learned important features of C#
  - C# is a Compiled Language
  - C# is Managed Code
  - C# is Strongly Typed
  - C# is Function Based
  - C# is Object-Oriented
- You learned to read and understand some C#
- In the next chapter, you'll write your first Unity C# program