**Unity Certification Preparation:**

**C# Programming**

**Orlando Unity3d Development Meetup**

**Date: 4-Sep-18**

**All Copyright PompaSetting Studios © 2018**

**Contents**

[**0.1** **Glossary** 3](#_Toc523808918)

# **0.1 Glossary**

|  |  |  |
| --- | --- | --- |
| NUM | TERM/GLOSSARY | DEFINTION 1 /EXAMPLE 1 |
| 0001 | A POINT | A point is a location in space.  There is only one point in space that has its coordinates.  The x,y,z of the point is a single location in space. A point plus a vector will result in a point |
| 0002 | A\* SEARCH | In computer science, A\* (pronounced as "A star") is a computer algorithm that is widely used in pathfinding and graph traversal, the process of plotting an efficiently directed path between multiple points, called "nodes". It enjoys widespread use due to its performance and accuracy. However, in practical travel-routing systems, it is generally outperformed by algorithms which can pre-process the graph to attain better performance,[1] although other work has found A\* to be superior to other approaches.[2] A\* achieves better performance by using heuristics to guide its search. |
| 0003 | API | API = Is the (programmable) interface between at least two or more software programs |
| 0004 | Artificial Neural Network | A computer program that tries to mimic the structure of the human brain |
| 0005 | ASSET PACKAGES | Collections of code that you can add to a project to help give you additional functionality, more artwork, or even existing game logic that you can build upon. |
| 0006 | ASYNCHRONOUS MULTIPLAY | Asynchronous gameplay is a popular phrase for describing various forms of online games that connect players but don't require simultaneous play. |
| 0007 | AWAKE function | The AWAKE function is used to declare and initialize the variables or game state before the game gets started. |
| 0008 | Awake Function | The Awake function is called on all objects in the scene before any object's Start function is called. |
| 0009 | BAKING | The process of creating a NAV MESH |
| 0010 | Box Collider | A box collider is an invisible shape around a gameobject. (They are denoted in the scene as having a green outline). Box colliders trigger an event to the console, once the perimeter or volume of the collider has been breached by another gameobject. They are used to detect collisions between gameobjects. |
| 0011 | BOX COLLIDER | Defines a cube area where collisions will be detected. |
| 0012 | ` | Breadth-first search (BFS) is an (Uninformed) algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'[1]) and explores the neighbor nodes first, before moving to the next level neighbors. |
| 0013 | BREADTH-FIRST SEARCH (BFS) | Starts from the Root Node. An algorithm that searches by visiting every state on a given layer, before progressing to the next layer down. Breadth first is both complete and optimal. However, because the search is very thorough. This type of search requires a lot of memory and processing time. FIRST IN-FIRST OUT QUEUE AGENDA. Meaning Adds newly discovered successors to the end. |
| 0014 | C# | C# (pronounced C Sharp) is a programming language designed for building a wide range of applications. C# is simple, powerful, type-safe, and object-oriented. With its many innovations, C# enables rapid protyping (application development) while retaining the expressiveness and elegance of C-style languages. |
| 0015 | CAMERA FRUSTUM | "Conical Pyramidal field of view in front of the notional camera |
| 0016 | Classes | Classes = are the blueprints for objects. They are used to create or instantiate objects |
| 0017 | COROUTINES | A COROUTINE is a "step-through" function. Where you pause execution of the program, to play out an event over 2 or more frames or a period of time, instead of executing the function in a 1 single step or 1 single frame update or in 1 s. |
| 0018 | Debug.Log | Debug.Log = Logs, echo outputs or prints a message to the Unity Console Window. |
| 0019 | DEPTH-FIRST SEARCH (DFS) | Depth-first search (DFS) is an (Uninformed) algorithm for traversing or searching tree or graph data structures. One starts at the root (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before backtracking. |
| 0020 | DEPTH-FIRST SEARCH (DFS) | DRILLSDOWN into one path of the tree and will back track if it does not find the goal state. STACK AGENDA. Adds newly discovered successors to the front (LAST-IN, LAST OUT) |
| 0021 | DEV | In Start Menu > Type Dev > Select Visual Studio Developer Command Prompt 2017 |
| 0022 | Dot notation | Dot Notation is used to call methods and variables contained within the specified object to the left |
| 0023 | DRAW CALL | Represents a single draw from a texture to a display |
| 0024 | Euler Angles | The Euler angles are three angles introduced by Leonhard Euler to describe the orientation of a rigid body with respect to a fixed coordinate system. |
| 0025 | Exception | An exception is an event on (post-compilation) execution (RUNTIME) that a program was not designed to handle, E.g. if the user inputs a negative height. |
| 0026 | FIELD | A field is a variable that exists inside of a class outside of the scope of a method. |
| 0027 | FIXED UPDATE | FixedUpdate called before any physics calculations. Runs several times a frame |
| 0028 | FINITE STATE MACHINE (FSM) | A decision-making tool for Artificial Intelligence. A finite-state machine is a model used to represent and control execution flow. It is perfect for implementing AI in games, producing great results without a complex code. Only a single state can be active at the same time, so the machine must transition from one state to another in order to perform different actions. |
| 0029 | FLOAT v DOUBLE | The DOUBLE, and FLOAT variable types are different in the way that they store the values. Precision is the main difference where FLOAT is a single precision (32 bit) floating point data type. Whereas DOUBLE is a double precision (64 bit) floating point data type. |
| 0030 | FLOCKING | Computer generated algorithm. Where NPC's move in cohesive groups rather than independently E.g. Birds, Fish Shoals. (3 main rules: alignment, cohesion, and separation) |
| 0031 | FRAME | A FRAME is the ( rectangular ) image sent to the MONITOR by your computers processor |
| 0032 | FRAME RATE (Or Frame Per Second) | The FRAME RATE (FPS) is HOW MANY TIMES (in 1 second) your computer (the CPU) can RENDER or DRAW an IMAGE to your monitor in 1 second. The higher the frame rate, the smoother, less jerky the visuals you will see. |
| 0033 | gameObject | gameObject = the object to which this script is attached |
| 0034 | gameobjects | GameObjects=Act as components for your scene |
| 0035 | GARBABGE COLLECTION | the automatic process of freeing up space in a computer's memory by removing data that is no longer required or in use. |
| 0036 | GetComponent<> | GetComponent returns the component you want to access from the inspector or script. |
| 0037 | GetKey | input.GetKey = returns the keycode for the key pressed on input |
| 0038 | GetKeyDown | Means whenever the key has been pressed DOWN. We then check the input |
| 0039 | GetkKeyUP | Means whenever key has been pressed down, then released. We check the event for the key input. |
| 0040 | GROUND PLANE | Ground Plane allows you to attached content to horizontal surfaces such as floors and tables. |
| 0041 | HideInInspector | Use this command to hide a public variable from the console. But still want it visible and accessible by other classes in the script |
| 0042 | HIERARCHY PANEL | The Hierarchy panel lists the game objects that are inside of the Scene Window |
| 0043 | IENumerator | Is the return type for Coroutines |
| 0044 | INSPECTOR | The INSPECTOR=Displays the properties for the selected object |
| 0045 | INTELLIGENT AGENT | pieces of Software that make decisions. INTELLIGENT AGENTS have 4 rules: 1)Agent should have the ability to perceive their environment (i.e a robot will have sensors, an NPC character will have eyes to see where the PC is) 2)AGENT will use information gathered from OBSERVATIONS to make a decision 3)THE DECISION will result in action being taken 4)Any decision made by an AI agent will be rationale |
| 0046 | IS KINEMATIC | When you want object to have PHYSICS properties, but be STATIONARY or STATIC in the scene |
| 0047 | Lerp | Linearly interpolating is finding a value that is some percentage between two given values. |
| 0048 | LocalScale |  |
| 0049 | Main | main = entry point of your program to be compiled, before it is run (or executed) |
| 0050 | Mesh | A mesh = collection of triangles in 3D space to create the impression of a solid object. |
| 0051 | Mesh Collider | The Mesh Collider takes a Mesh Asset and builds its Collider based on that Mesh. It is far more accurate for collision detection than using primitives for complicated Meshes. Mesh Colliders that are marked as Convex can collide with other Mesh Colliders. |
| 0052 | MESH RENDERER | The Mesh Renderer takes the geometry from the Mesh Filter and renders it at the position defined by the object’s Transform component. |
| 0053 | Mono develop | Is the default IDE that comes with unity |
| 0054 | MonoBehaviour | MonoBehaviour is the parent or base class from where all the Commands you use in your script are derived |
| 0055 | MonoBehaviour | Monobehaviour is the base class where you inherit or access all the methods, keywords, and commands you use in your code, i.e. Monobehaviour allows you to add scripts to gameObjects in your scene |
| 0056 | MonoBehaviour | Using Javascript every script automatically derives from MonoBehaviour. When using C# or Boo you have to explicitly inherit from MonoBehaviour. |
| 0057 | Namespaces | Namespaces = used to differentiate classes that have identical names |
| 0058 | Namespaces | Namespaces = are used to avoid naming conflicts. When you start to add libraries from other programmers its highly likely we are going to use the same names. Therefore, we create a namespace as a unique group name identifier for the collection of classes that it contains |
| 0059 | Namespaces | A namespace is a collection of classes |
| 0060 | NAVIGATION MESH | A navigation mesh, or navmesh=is an abstract data structure used in artificial intelligence applications to aid agents in pathfinding through complicated spaces. A Nav Mesh=rspons |
| 0061 | NAVIGATION MESH | An invisible flat plane to move characters around in a scene |
| 0062 | NON-PRIMITIVE DATA TYPES | Non-primitive data types are not defined by the programming language but are instead created by the programmer. They don't store the value, they store the reference to the value ( a bit like pointers). And are sometimes called "reference variables (class, interface, array variable)," or "object references," since they reference a memory location, which stores the data. |
| 0063 | NPC | Non-Player Characters in a Game |
| 0064 | NPC | Non-Player Characters in a Game |
| 0065 | OFF-MESH LINKS | Off-Mesh Links are used to create paths crossing outside the walkable navigation mesh surface. For example, jumping over a ditch or a fence, or opening a door before walking through it, can be all described as Off-mesh links. |
| 0066 | OnCollisionEnter2D | Used For 3D OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider. |
| 0067 | OnCollsionEnter | Used For 2D OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider. |
| 0068 | OOP | AT THIS POINT. One thing to understand about C#. As well as being an Object Orientated Programming Language (where the program is structured into classes and objects) |
| 0069 | PARALLEX SCROLLING | PARRALLEX SCROLLING=Is when the background moves slower than the foreground to give the sense of depth |
| 0070 | PARRALLAX SCROLLING | Which is when the background moves slower than the foreground to create the illusion of depth and scale |
| 0071 | PIXEL PERFECT | PIXEL PERFECT=UI adjusted to the nearest PIXEL when rendered. Sharpens look of UI element |
| 0072 | PREFABS | PREFAB = pre-made gameObject which you can use again, again & again. |
| 0073 | PREFABS | Prefab is a type of asset -- a reusable GameObject stored in Project View |
| 0074 | Primitive Types | They are called primitive data types , because they are the main built-in types, that come with the compiler...as the programmer you don't need to do anything to use them. The most famous primitive data types are: int, short, char, float, double, char, bool. In C#, primitive data types are actually objects, It means when you write the following code, variable foo is actually an Object. String and Object are classes |
| 0075 | private | PRIVATE before a method or a variable means the value or data is hidden from other classes in your program. The method or variable can only be called by the SAME class that it is in. |
| 0076 | PROJECT PANEL | The Project window, in this view is where you ACCESS, IMPORT, STORE & EDIT the ASSET files for your project |
| 0077 | Property | A property = is a special sort of class member, intermediate in functionality between a field (or data member) and a method. |
| 0078 | PUBLIC | PUBLIC before a method or a variable means the value or data is VISIBLE, can be ACCESSED by other classes in your program. |
| 0079 | Quaternion | Complex data structure that holds a rotation |
| 0080 | Quaternion.identity | Means the object has no rotation |
| 0081 | REALTIME MULTIPLAYER | connect multiple players together in a single game session and transfer data messages between connected players |
| 0082 | RECT TRANSFORM |  |
| 0083 | Rendering | Is the process of taking the assets of an image and turning it into a video game |
| 0084 | Rigidbody |  |
| 0085 | Rotation |  |
| 0086 | RUNTIME ERROR | The code throws an exception and crashes (after compilation ) when you run the program |
| 0087 | SCENE | SCENE=Simply means Level or environment |
| 0088 | Scene View / Window | SCENE VIEW = DESIGN WINDOW the space inside the Unity IDE where you can edit, change & modify your assets |
| 0089 | Scenes | Scenes = Scenes are where your GameObjects are placed to make a game level. Games are made of one or more scenes (aka, levels) linked together. |
| 0090 | Screen Match Mode | Screen Match Mode=used to scale the canvas area if the aspect ratio of the current resolution doesn't fit the reference resolution. E.g. If the screen resolution is larger, the UI will be scaled up, and if it's smaller, the UI will be scaled down. |
| 0091 | SCREEN SPACE OVERLAY | UI elements will be displayed in front of any objects at all times. Default Setting for a CANVAS |
| 0092 | SCRIPTABLE OBJECT | A Scriptable Object=A Class derived from Unity's Object Class, whose references and fields can be serialized. |
| 0093 | SCRIPTABLE OBJECT | A Scriptable Object=1)A Script 2)It doesn't receive (most) callbacks from Unity 3)At runtime not attached to any specific GameObject 4)Each different instance can be saved to its own file. |
| 0094 | Scripted Behaviour | Objects that do not respond to events IS NOT INTELLIGENT |
| 0095 | SERIALIZATION | Saves the object state |
| 0096 | SerializeField | modifier to access the field from the inspector |
| 0097 | Solution | A solution = just a collection of projects |
| 0098 | SPRITE | Sprites=A 2D Graphic. An individual graphics that can be static or animated in a 2D game |
| 0099 | SPRITE SHEETS | Sprite sheets are multiple images put together on a single image. |
| 0100 | static = | static means that the method cannot be re-created i.e. there will only be one instance of this method generated -> you cannot create another main method in your program |
| 0101 | static void main(String args[]) | Aka THE MAIN METHOD. It’s the start and end of the program instructions. |
| 0102 | static void main(String args[]) | static void Main(string[] args) |
| 0103 | String args[] | The Main method can be declared with or without the parameter string args[]. An array of Strings (args is just the name of the array. You can name it something else if you want) |
| 0104 | String args[] | The parameter of the Main method |
| 0105 | Tags | Keywords attached to gameObjects |
| 0106 | TAGS | TAGS are KEYWORDS attached to gameObjects, so we can identify the gameObjects. TAGS are useful for 1)Managing Collisions 2)Getting Information on GAMEOBJECTS |
| 0107 | TEXTURE COMPRESSION | By default, it tries to optimize the textures to work in a 3D environment |
| 0108 | The GAME view | The GAME VIEW is rendered from the Camera(s) in your game. It is representative of your final, published game. |
| 0109 | Time.deltaTime | Time.deltaTime = is the time passed since the last frame. Or the time between frames. |
| 0110 | Time.deltaTime | DeltaTime is a static variable inside of the TimeClass. DeltaTime = the time taken to render the last frame. |
| 0111 | Transform | The transform is the container to store and update the position, rotation and scale of the gameobject in the scene. Every Transform can have a parent, which allows you to apply position, rotation and localScale hierarchically. |
| 0112 | Translate | transform.Translate = In effect the instruction to move an object from Point A to Point B in the scene. |
| 0113 | UNITY ASSET STORE | The UNITY ASSET STORE is a Market Place. The UNITY SHOP inside of the Unity Editor where you can purchase free and paid ASSETS Like plugins, artwork, models, textures and other components to use in your unity project. |
| 0114 | Update v FixedUpdate | Update runs once per frame. FixedUpdate can run once, zero, or several times per frame, depending on how many physics frames per second are set in the time settings, and how fast/slow the framerate is |
| 0115 | using System.Collections | Using System.Collections = is called a namespace, they give's you access to the Microsoft C# library |
| 0116 | using UnityEngine | Using UnityEngine = is a called a namespace, they give you access to the UnityEngine library or API |
| 0117 | VARIABLE TYPE | A VARIABLE TYPE determines what kind of data is allowed to be assigned to the variable. |
| 0118 | Vector | A Vector (Vector2, Vector3 or Vector4) is a ??? that allows you to assign more than one value to a variable |
| 0119 | VECTOR | A vector is a direction and length.  A vector has no predetermined starting point. Vectors with the same values can be anywhere. the x,y,z of a vector is the length of the vector in each of those dimensions. A vector plus a vector will result in a vector |
| 0120 | Vector2 | In 2D mode. Vector2 allows you to store the position (or Co-Ordinates) 2 values of the gameobject in your Scene (x,y) |
| 0121 | Vector3 | In 3D mode. Vector3 allows you to store the position (or Co-Ordinates) 3 values of the gameobject in your Scene (x,y,z) |
| 0122 | Vector4 |  |
| 0123 | void = | void means that there is no data (or return value) to be returned by this method. E.g. to a variable or another method. Hence the return 0; (for the sake of completeness) |
| 0124 | void FixedUpdate | FixedUpdate should be used instead of Update when dealing with PHYSICS and Rigidbody. Update runs once per frame. Whereas FixedUpdate can run once, zero, or several times per frame. |
| 0125 | VS2015/2017 | Microsoft VISUAL STUDIO 2015/2017 IDE Code Editor |
| 0126 | VUFORIA START RATING | VUFORIA START RATING=Based on Contrast, detail and features. Places pattern on marker, which camera recognizes then software places 3D model on top of the surface |
| 0127 | Vuforia WIDTH | width of your target in scene units. The size of the target should be on the same scale as your augmented virtual content. Vuforia uses meters as the default unit scale. The target's height will be calculated when you upload your image. |
| 0128 | VuMark | Similar to QR Code |
| 0129 | WAYPOINTS | (Sequenced) POSITIONS on a MAP |
| 0130 | Why are we using Visual Studio 2015 instead of 2017 |  |
| 0131 | Why code in C# for unity ? | Why code in C# for unity ? 1)More organized way of coding. E.g. if you declare your data types upfront, it makes it easier to review and debug your code later on...because the variables have already been explicitly declared 2)The number 2 reason is because C# uses Visual Studio. Personally, I have not come across a better code editor on the market. With Intellisense and inline help, they provide more tools and support to debug your code, 3)Most documentation/tutorials/examples are written in C# |
| 0132 | WORLD SPACE | Is the x,y,z static gizmo directions |
| 0133 | CRLF | Carriage Return LF |
| 0134 | LAYER | Restricts interactions between objects |
| 0135 | VOIP | Voice over Internet Protocol. VoIP allows you to make free, or very low cost, telephone calls over the Internet. |
| 0136 | PHYSIC MATERIAL | The Physic Material is used to adjust friction and bouncing effects of colliding objects. |
| 0137 | MESH | A mesh is made up of Polygons or Triangles. |

/End